Contents

1 تعریف پایگاه داده

این درس درباره پایگاه داده است. واژه پایگاه داده از دو واژه تشکیل شده است. برای تعریف، اول داده را تعریف می کنیم.

تعریف داده Data: هر نوع واقعیت در مورد افراد، اشیا و رخدادها و به طور کلی هر نوع اطلاعات خام به عنوان داده شناخته می شود.

مثال: واقعیت درباره یک فرد

1.1 پایگاه داده دوره های آنلاین

برای آشنایی با مفاهیم پایگاه داده اجازه بدهید که کار خود را با یک مثال شروع کنیم. این پایگاه داده مربوط به دوره‌های آنلاینی است که توسط یک موسسه برگزار می گردد. موسسه دوره‌های مختلفی دارد. فرضیات زیر را در نظر می گیریم.

- هیچ دو دوره متفاوتی با نام یکسان نداریم. یعنی مثلا فقط یک دوره پایگاه داده داریم.
- نام کوچک دانشجویان یکتاست. هیچ دو فردی با نام یکسان وجود ندارد.
- هر دانشجو می تواند بدون محدویت در دوره‌های شرکت کند، یعنی حتی بدون خرید! این دانشجویان اجازه دونلود فیلم یا جزوه را ندارند.

سوال) در دوره آنلاین ما چه افراد، اشیا، رخدادهای داریم؟
پاسخ) افراد= دانشجویان، اساتید رخدادها (کاری که افراد انجام می‌دهند) = حضور در دوره، ثبت‌نام در دوره

سوال) داده های مربوط با دانشجویان چیست؟
پاسخ) دانشجو (نام، شهر محل سکونت، نام دانشگاه لیسانس)
سوال) داده های مربوط به یک دوره چیست؟
پاسخ) دوره (نام دوره، نام مدرس)
سوال) داده های مربوط به یک ثبت نام چیست؟
پاسخ) ثبت‌نام(نام دانشجو، نام دوره، شماره مرجع بانکی، مبلغ واریز)
سوال) داده های مربوط به یک حضور چیست؟
پاسخ) حضور (نام دانشجو، نام دوره، شماره جلسه، ساعت حضور)

حالا داده ها را در یک تعداد جدول سازماندهی می کنیم.

Student

sname	city	uni
Reza	Mashad	Qom
Ali	Najaf	Tehran
Negin	Brojerd	Sharif
Sara	Rasht	Ferdowsi

Student(sname, city, uni)

course

cname	tname
DB	Hadi
AI	Ramin

course(cname, tname)

buy

sname	cname	Trno	Amount
Reza	DB	1388	600
Ali	AI	1370	650
Negin	DB	1365	550

buy(sname, cname, Trno, Amount)

att

sname	cname	meeting	hour
Negin	DB	1	4
Negin	DB	2	2
Reza	AI	1	4
Negin	AI	1	3

Att (sname, cname, meeting, hour)

حالا که این داده ها را سازماندهی کردیم. یک نام جدید پیدا می کنند. اطلاعات!

تعریف اطلاعات Information: هر مجموعه از داده های سازماندهی شده را اطلاعات می گویند.

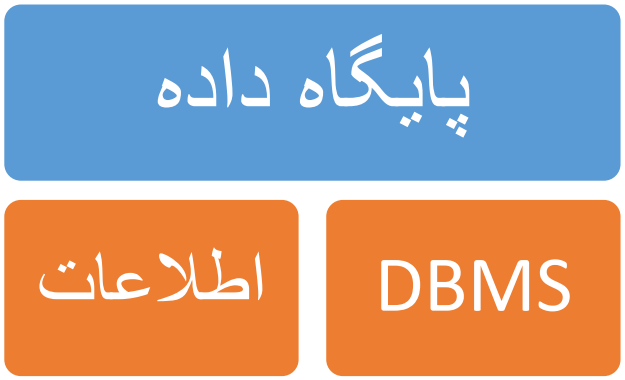
نکته: تنها راه سازماندهی جدول بندی داده ها نیست!

تعریف پایگاه داده Database:

یک پایگاه داده مشتمل بر دو مولفه اصلی است. مجموعه از داده های سازماندهی شده دایمی و یک سامانه مدیریتی به نام DBMS

Database = DB

DBMS = DB Management System



نکته: DBMS یک نرم افزار است. مثل برنامه Microsoft word که شما با آن مقاله می‌نویسید. با استفاده از نرم‌افزار DBMS می‌توانید پایگاه داده را ایجاد کنید، داده وارد کنید یا داده‌ها را از DB بخوانید. از مشهورترین DBMS ها می‌توان Microsoft SQL را نام برد.

سوال) چه کسانی از پایگاه‌داده استفاده می‌کنند؟ کاربران پایگاه داده چه کسانی هستند؟
پاسخ) کاربران پایگاه‌داده دو دسته هستند.
۱. کاربر عادی. تخصصی در زمینه کامپیوتر ندارد. مثلا هر کس که با اپلیکشن همراه بانک کارت به کارت می‌کند از طریق موبایل به پایگاه‌داده بانک دسترسی دارد.
۲. کاربر متخصص. دانش تخصصی کامپیوتر و پایگاه‌داده دارد. مثل مدیر پایگاه‌داده که پایگاه داده را مدیریت و طراحی می‌کند. یا برنامه‌نویسان که بری نوشتن برنامه‌های کاربردی از پایگاه‌داده استفاده می‌کنند.

سوال) کاربرد اصلی پایگاه داده چیست؟
پاسخ) بازیابی اطلاعات یا گزارش گیری از داخل پایگاه. مثلا به سوالات زیر پاسخ بدهیم.
لیست افرادی که در دوره ها را خریده اند چیست؟
چند نفر دوره های را خریده اند؟
چند نفر در دوره های شرکت کرده اند؟
کسانی که دوره ها را خریده اند، از کدام دانشگاه ها بوده اند؟

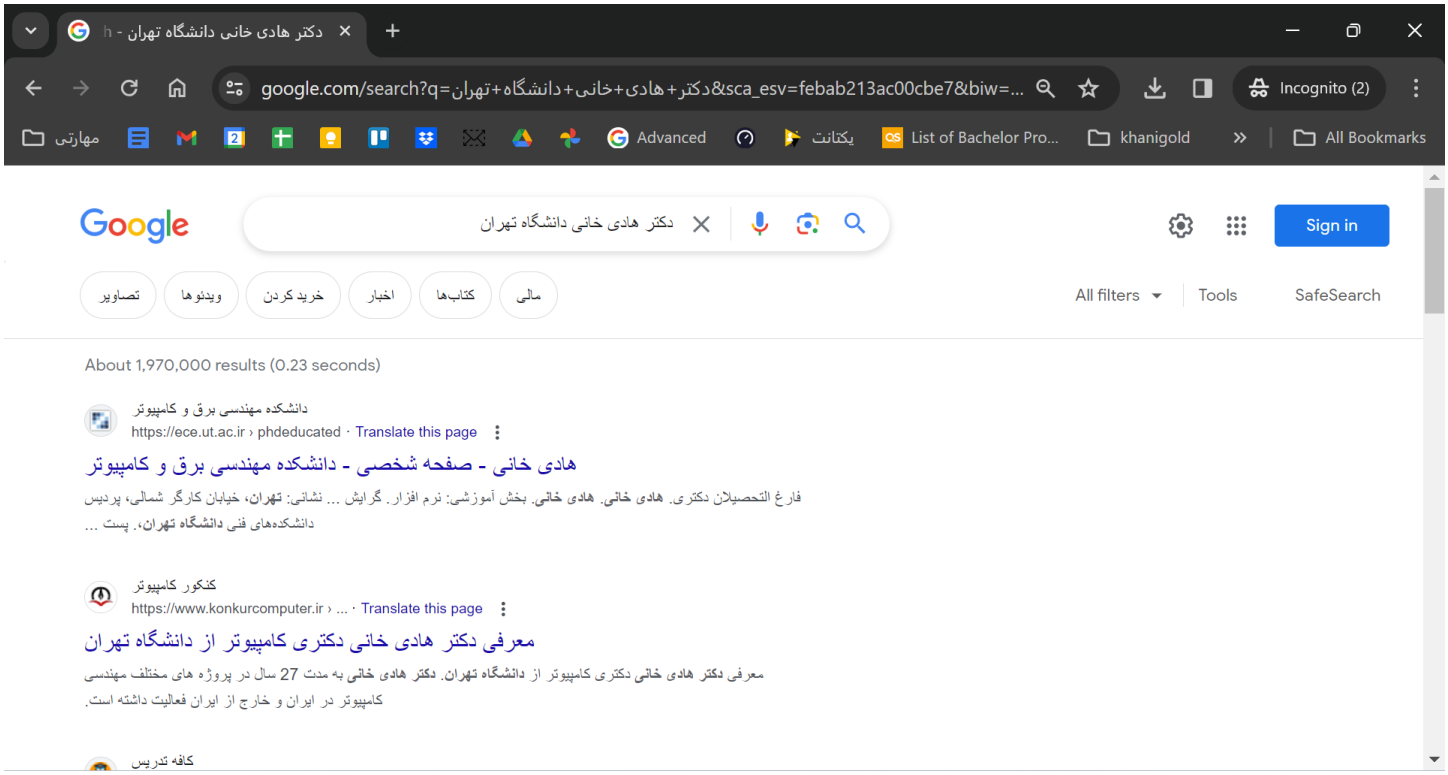
کاربران متخصص برای ارتباط با پایگاه‌داده از زبانی به نام SQL استفاده می‌کنند. در ادامه دستور SELECT که برای بازیابی اطلاعات از پایگاه‌داده استفاده می‌شود با مثال آشنا می‌شویم. SQL مخفف Sequential Query Language است. به عبارتهایی که با زبان SQL نوشته می‌شود، کوئری یا پرس‌وجو گفته می‌شود.

2 دستورات پایگاه داده

2.1 دستور Select

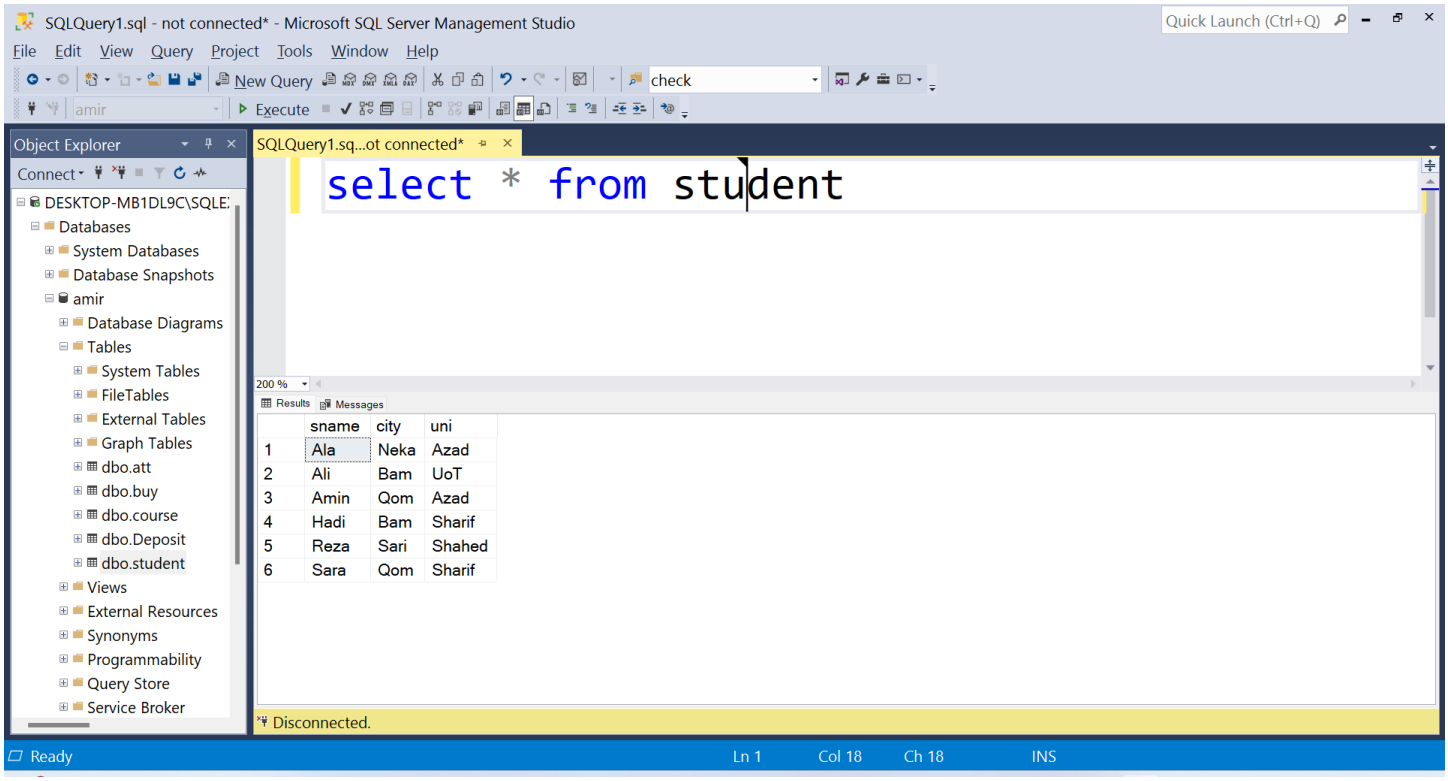
وقتی می خواهیم گزارش گیری کنیم یا اطلاعات را از داخل پایگاه داده بخوانیم، از دستور Select استفاده می کنیم.

سوال) قبل از شرکت در کلاس امروز، از پرس و جو کجا استفاده کرده اید؟
هر کسی که در گوگل جستجو می کند از کوئری یا پرس و جو استفاده کرده است. در واقع آنچه که شما در باکس جستجو گوگل تایپ می کنید. کوئری است. البته کوئری‌های گوگل برای افراد عادی طراحی شده‌است و ساده‌تر است. ولی کوئری‌های که ما در زبان پایگاه‌داده می‌نویسیم به زبان SQL است و کمی دقیق‌تر و پیچیده‌تر است. که در طول این دوره آن را یاد می‌گیرید.



2.1.1 پرس و جوی۱: انتخاب همه ستونها

مشخصات همه دانشجویان



شکل 1 Microsoft SQL Server Management Studio

نکته: به جای نوشتن اسم همه ستونهای جدول از کاراکتر * استفاده می کنیم.
نکته: کلماتی که با رنگ آبی نوشته شده است. کلمات کلیدی زبان است. باید عینا به همین شکل نوشته شود.

2.1.2 پرس و جوی ۲: انتخاب یک ستون خاص

نام دانشجویانی که دوره ها را خریداری کرده اند.

```
select sname from buy
```

	sname
1	Reza
2	Reza
3	Ala
4	Hadi
5	Amin
6	Hadi
7	Ala

2.1.3 پرس و جوی ۳: انتخاب سطر

نام کلیه افرادی که در دوره AI شرکت کرده اند.

```
Select sname from att where cname = 'AI'
```

	sname
1	Reza
2	Reza
3	Sara

نکته: دستور select به شکل خودکار، سطرهای تکراری را حذف نمی کند. و مثلا دو بار reza را گزارش می دهد. برای حذف سطرهای تکراری، از distinct باید استفاده شود.

```
Select distinct sname from att where cname = 'AI'
```

	sname
1	Reza
2	Sara

2.1.4 پرس و جوی ۴: ترکیب شرطها به and

کلیه دانشجویانی که دوره DB را زیر 600 هزار تومان خریده اند.

```
select sname from buy where cname = 'DB'  and amount < 600
```

	sname
1	Hadi

```
select * from buy
```

200 %

Results Messages

	sname	cname	amount	trno
1	Reza	GS	300	3322
2	Reza	AI	520	3448
3	Ala	AW	320	4423
4	Hadi	AI	550	5654
5	Amin	DB	600	7074
6	Hadi	DB	500	8997
7	Ala	AI	430	9097

نکته: برای شرط های پیچیده باید از **and** و **or** استفاده کنیم.

2.1.5 مراحل نوشتن پرس‌وجو

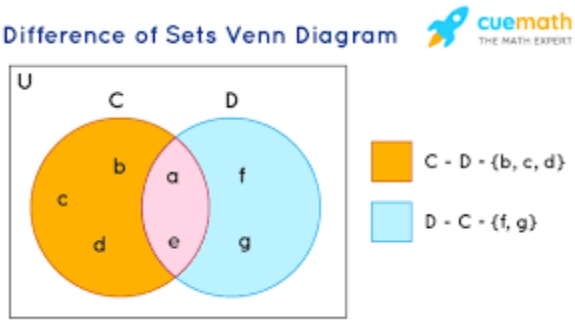
نکته: برای نوشتن پارس و جو یا کوئری سه مرحله باید انجام شود

1- فهم پرس و جو
2- شناسایی جدول یا جدولهای مرتبط با پرس و جو
3- انتخاب دستورالعملهای مناسب برای جستجو

2.1.6 پرس و جو ۵ : تفاضل except

نام افرادی که در کلاس حضور داشته‌اند ولی در لیست دانشجویان نیستند.

راه حل: در اینجا اولین مثالی که باید از دو جدول استفاده کنیم.



اول نام همه افرادی که در کلاسی حاضر بوده‌اند را از جدول att پیدا می‌کنیم.بعد نام همه دانشجویان را از جدول student پیدا می‌کنیم. جواب تفاضل این دو مجموعه است. که با **except** انجام می شود

```
--stage 1
select sname from att
```

	sname
1	Ala
2	Ali
3	Amin
4	Hadi
5	Reza
6	Sara

```
--stage 2
select sname from student
```

	sname
1	Sara
2	Reza
3	Reza
4	Reza
5	Sara
6	Ala
7	Amin

```
--stage 3
(select sname from att)
except
(select sname from student)
```

	sname
1	Ali
2	Hadi

نکته: Except به طور خودکار تمامی سطرهای تکراری را حذف می کند. برای اینکه همه سطرها را داشته باشیم. از **except all** استفاده کنید.

2.1.7 پرس وجو ۶ : تفاضل except

دانشجویانی که درس DB را خریده اند ولی در کلاس آن شرکت نکرده اند.

گام ۱) دانشجویانی که در درس DB را خریده‌اند.
گام ۲) دانشجویانی که در درس DB شرکت داشته‌اند.
گام ۳) این دو را با except از هم کم می کنیم.

```
--stage 1
Select sname from buy
where cname = 'DB'
```

	sname
1	Amin
2	Hadi

```
--stage 2
Select sname from att
where cname = 'DB'
```

	sname
1	Sara
2	Reza
3	Ala
4	Amin

```
--stage 3
(select sname from buy
where cname ='DB')
except
(select sname from att
where cname = 'DB')
```

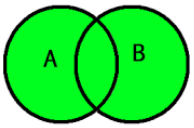
	sname
1	Hadi

2.1.8 پرس وجو ۷: اجتماع union

نام دانشجویانی که به نوعی با درس پایگاه داده علاقمند بوده اند.

یعنی دانشجویانی که این درس را خریده اند یا در کلاسهای آن شرکت کرده اند.

A = {1,2,3}
B = {3,4,5}
AUB = {1,2,3,4,5}



گام ۱) دانشجویانی که درس DB را خریده‌اند.
گام ۲)دانشجویانی که در کلاسهای DB شرکت کرده‌اند.
گام ۳) کافی است با union اجتماع این دو مجموعه را به دست آوریم.

```
--stage 1
Select sname from buy
where cname = 'DB'
```

	sname
1	Amin
2	Hadi

```
--stage 2
Select sname from att
where cname = 'DB'
```

	sname
1	Sara
2	Reza
3	Ala
4	Amin

```
--stage 3
(select sname from buy
where cname ='DB')
union
(select sname from att
where cname = 'DB')
```

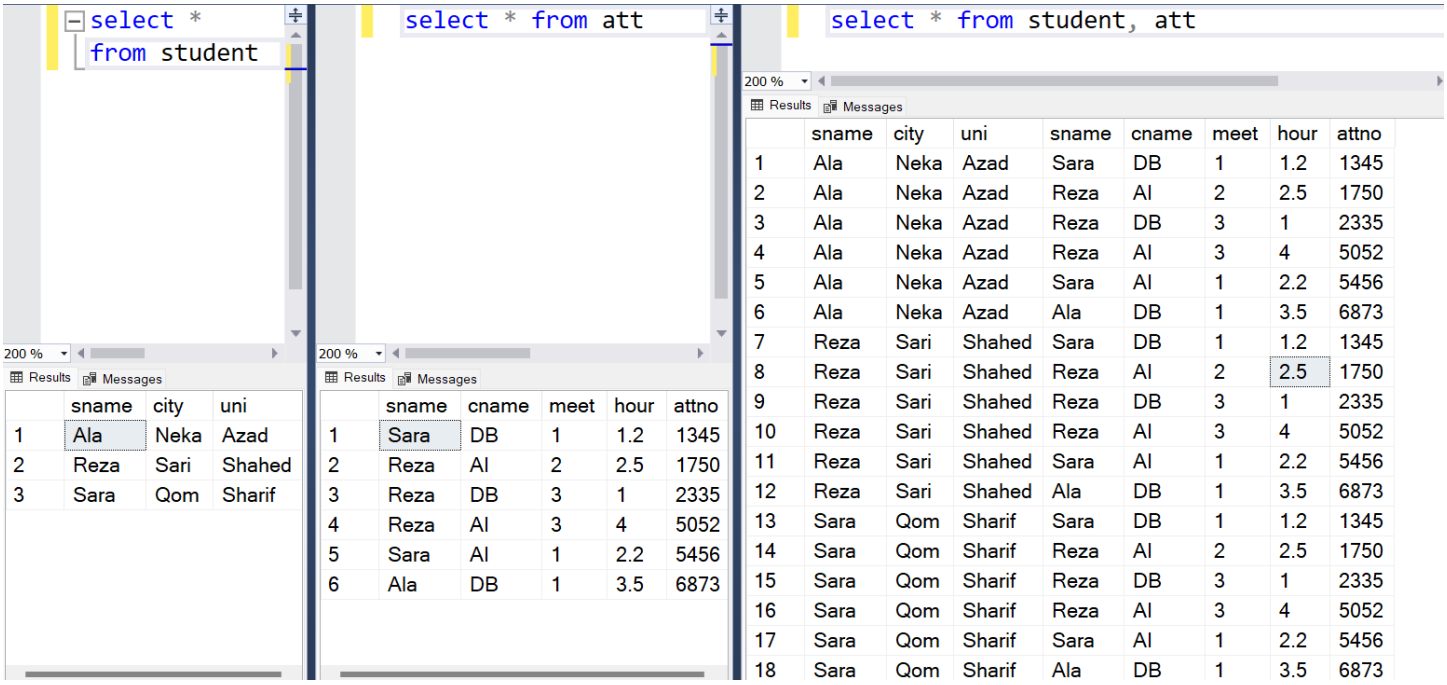
	sname
1	Ala
2	Amin
3	Hadi
4	Reza
5	Sara

نکته در عمل **union** و **except** شرط سازگاری باید رعایت شود. یعنی ستونهایی که با هم union و except می‌شوند باید هم‌نوع باشند.
نکته **union** مثل except سطرهای تکراری را حذف می کند. اگر بخواهیم حذف نشود از union all استفاده کنید.

2.1.9 پرس و جو ۸: ضرب دکارتی (الحاق دو جدول)

نام و نام دانشگاه کلیه دانشجویانی که در درس DB شرکت کرده اند.

نام دانشگاه در جدول student است و اسم دوره در جدول buy این دو باید با هم ترکیب شوند. عمل union و except جوابگو نیست. اینجا یک عمل جدید لازم داریم به نام ضرب دکارتی. اول از همه این ضرب دکارتی را حساب می کنیم. همه سطرهای جدول اول با جدول دوم ترکیب می شود. و در مجموع ۱۸ سطر خواهیم داشت.



```
select *
from student
```

	sname	city	uni
1	Ala	Neka	Azad
2	Reza	Sari	Shahed
3	Sara	Qom	Sharif

```
select * from att
```

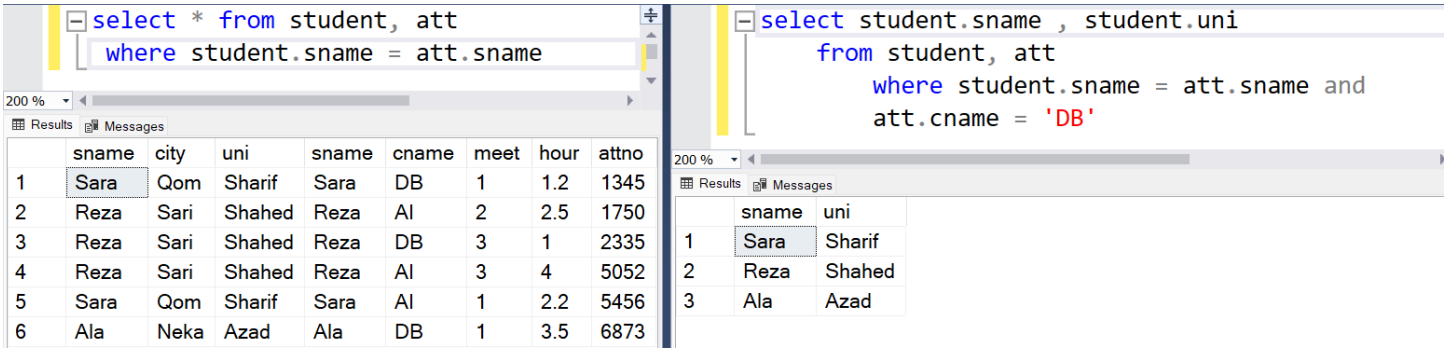
	sname	cname	meet	hour	attno
1	Sara	DB	1	1.2	1345
2	Reza	AI	2	2.5	1750
3	Reza	DB	3	1	2335
4	Reza	AI	3	4	5052
5	Sara	AI	1	2.2	5456
6	Ala	DB	1	3.5	6873

```
select * from student, att
```

	sname	city	uni	sname	cname	meet	hour	attno
1	Ala	Neka	Azad	Sara	DB	1	1.2	1345
2	Ala	Neka	Azad	Reza	AI	2	2.5	1750
3	Ala	Neka	Azad	Reza	DB	3	1	2335
4	Ala	Neka	Azad	Reza	AI	3	4	5052
5	Ala	Neka	Azad	Sara	AI	1	2.2	5456
6	Ala	Neka	Azad	Ala	DB	1	3.5	6873
7	Reza	Sari	Shahed	Sara	DB	1	1.2	1345
8	Reza	Sari	Shahed	Reza	AI	2	2.5	1750
9	Reza	Sari	Shahed	Reza	DB	3	1	2335
10	Reza	Sari	Shahed	Reza	AI	3	4	5052
11	Reza	Sari	Shahed	Sara	AI	1	2.2	5456
12	Reza	Sari	Shahed	Ala	DB	1	3.5	6873
13	Sara	Qom	Sharif	Sara	DB	1	1.2	1345
14	Sara	Qom	Sharif	Reza	AI	2	2.5	1750
15	Sara	Qom	Sharif	Reza	DB	3	1	2335
16	Sara	Qom	Sharif	Reza	AI	3	4	5052
17	Sara	Qom	Sharif	Sara	AI	1	2.2	5456
18	Sara	Qom	Sharif	Ala	DB	1	3.5	6873

سوال) کدام سطر با معنی هستند؟

پاسخ) سطر ۱ بی معنی است. این سطر را با هم می‌خوانیم. علا ساکن شهر نکا و در دانشگاه آزاد درس خوانده است. ضمناً اسمش سارا است! بدیهی است کسی که نامش علا است نمی‌تواند نامش سارا باشد. پس این سطر بی‌معنی یا نامعتبر است.
سطر ۶ معتبر است. این سطر می‌گوید. علا ساکن شهر نکا و دانشجوی دانشگاه آزاد در جلسه ۱ درس پایگاه‌داده به مدت یک ساعت شرکت کرده‌است. پس این سطر با معنی یا معتبر است.
نکته) در ضرب دکارتی تعداد زیادی سطرهای بی معنی یا بی اعتبار ایجاد می شود. تنها سطرهایی که مقدارشان در ستون مشترک یکسان باشد معتبر یا با معنی خواهند بود. ما باید در سطرهای بامعنی یا معتبر که اصطلاحا شرط ضرب را دارند، دنبال اطلاعات باشیم.
تعریف شرط ضرب) سطرهایی که در ستون مشترک مساوی باشند شرط ضرب را دارند.
تعریف ستون مشترک) ستون مشترک در دو جدول باید نوع و معنای یکسانی داشته باشند. یکسان بودن نام دو ستون مشترک الزامی نیست.



```
select * from student, att
  where student.sname = att.sname
```

	sname	city	uni	sname	cname	meet	hour	attno
1	Sara	Qom	Sharif	Sara	DB	1	1.2	1345
2	Reza	Sari	Shahed	Reza	AI	2	2.5	1750
3	Reza	Sari	Shahed	Reza	DB	3	1	2335
4	Reza	Sari	Shahed	Reza	AI	3	4	5052
5	Sara	Qom	Sharif	Sara	AI	1	2.2	5456
6	Ala	Neka	Azad	Ala	DB	1	3.5	6873

```
select student.sname , student.uni
    from student, att
        where student.sname = att.sname and
        att.cname = 'DB'
```

	sname	uni
1	Sara	Sharif
2	Reza	Shahed
3	Ala	Azad

با توجه به تعاریف بالا تنها 6 سطر معتبر (با معنی) داریم. این شش سطر در ستون مشترک مقادیر یکسان داشته و اصطلاحا **شرط ضرب** را دارند. با اضافه کردن شرط att.cname = ‘DB’ فقط سطرهای معتبری درس پایگاه داده را انتخاب می‌کنیم.

فهرست مطالب

Contents

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

1 جبر رابطه‌ای Relational Algebra

	sname	cname	meet	hour	attno
1	Sara	DB	1	1.2	1345
2	Reza	AI	2	2.5	1750
3	Reza	DB	3	1	2335
4	Reza	AI	3	4	5052
5	Sara	AI	1	2.2	5456
6	Ala	DB	1	3.5	6873

برای سازماندهی پایگاه‌داده می‌توان از جدول استفاده کرد. یک سطر جدول شامل مجموعه‌ای از مقادیر values است که با هم رابطه دارند. مثلا معنای سطر ۱ از جدول att که آن را می‌توان به شکل (sara, DB, 1, 1.2, 1345) نمایش داد می‌شود. sara در کلاس DB جلسه 1 به مدت ۱.۲ ساعت با شماره حضور 1345 حاضر بوده است.

نکته: همانگونه که در درس مدار منطقی، برای طراحی و تحلیل مدارهای منطقی از جبر بول Boolean Algebra استفاده می‌کنیم، در درس پایگاه داده هم از جبر رابطه‌ای Relational Algebra استفاده می‌کنیم.در این فصل جبر رابطه‌ای به عنوان یک مبحث نظری ریاضی درس داده می‌شود.

1.1. پرس‌وجو نویسی با جبر رابطه‌ای

یکی از کارهای جالبی که با جبر رابطه‌ای می‌توان انجام داد و در کنکور هم سوال می‌آید نوشتن پرس‌وجو یا کوئری و بازیابی اطلاعات از داخل جدول‌ها است. در این فصل پرس‌وجوهایی که با زبان SQL نوشته بودیم را این‌بار با عملگرهای جبر رابطه‌ای خواهیم نوشت.

1.1.1 پرس و جوی 1 : انتخاب همه ستون‌ها

کلیه مشخصات خریدهای انجام شده دانشجویان

پاسخ) در جبر رابطه ای Buy است.

1.1.2 پرس‌وجوی 2: انتخاب یک ستون با عمگر پرتو

نام کلیه دانشجویانی که دوره ها را خریداری کرده اند. برای این کار باید از عملگر π استفاده کنیم. به این عملگر پرتو یا projection گفته می‌شود.

Select sname **from** Buy

Π_{sname} (buy)

نکته: در جبر رابطه ما عملگر داریم مثل π در SQL ما دستور داریم مثل Select.

1.1.3 پرس‌وجوی 3: انتخاب سطر با عملگر انتخاب

نام کلیه افرادی که در دوره DB شرکت کرده اند. برای این کار باید از عملگر σ استفاده کنیم. به این عملگر انتخاب یا selection گفته می‌شود.

Select sname **from** att **where** cname = 'DB'

Π_{sname} ($\sigma_{cname = 'DB'}$(att))

نکته: عملگر π (تقریباً) معادل دستور select در sql است.
نکته: عملگر Π تاپل‌های تکراری را حذف می‌کند. در حالی که select اینکار را نمی‌کند. برای حذف تاپل‌های تکراری باید از select distinct استفاده کرد.

1.1.4 پرس‌وجو 4: ترکیب شرط‌ها با و منطقی

کلیه دانشجویانی که دوره DB را زیر 600 هزار تومان خریده اند.

select sname **from** buy **where** cname = DB **and** amount < 600

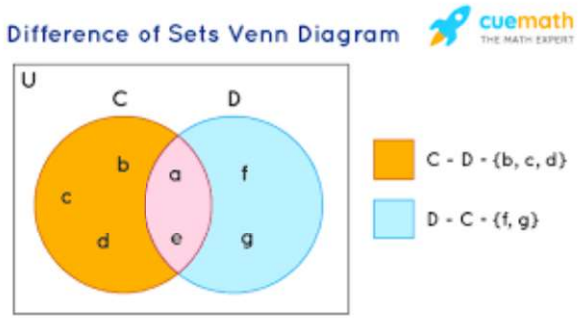
Π_{sname} ($\sigma_{cname = 'DB' \wedge amount<600}$(buy))

1.1.5 پرس‌وجو 5: عمگر تفاضل

نام دانشجویانی که تا کنون در کلاسها شرکت نکرده اند.

(Select sname **from** student**) except (Select** sname **from** att**)**

$\Pi_{sname}(student) - \Pi_{sname}(att)$



نکته: در عملگر تفاضل مانند except باید شرط سازگاری برقرار باشد که در این مثال داریم.
نکته: تفاضل هر دو رابطه دلخواه امکانپذیر نیست. دو رابطه باید شرط سازگاری داشته باشند.
تعریف شرط سازگاری: الف) تعداد مؤلفه‌ها (ستونها) یکسان باشد یا در اصطلاح تاپل‌ها هم درجه باشند. ب) دامنه مولفه‌های (ستونهای)متناظر یکسان باشد.
نکته: تمامی عملگرهای مجموعه‌ای که در دو طرف مجموعه دارند مانند $\cup$ و $\cap$ و – باید شرط سازگاری داشته باشند.

مثال) عبارت زیر غلط است زیرا دو ستون یکی نیست. و شرط سازگاری را ندارد.
$\Pi_{sname}(student) - \Pi_{city}(att)$

1.1.5.1.1.1 تست مهندسی کامپیوتر سراسری سال ۱۴۰۱ ۴

۴۳. کدام مورد، خروجی رابطه روبه‌رو است؟

$(\delta(\mathrm{STUD}))\cap(\delta(\mathrm{CRS})) = ?$

$\mathrm{Avg} > 16 \qquad \mathrm{Unit} = 3$

((معدل) Avg ,(شهر) City , (نام و نام‌خانوادگی) sname , (شماره دانشجویی) S#) (دانشجو) STUD
((مدرک) DEGREE ,(شماره اتاق) Office ,(نام استاد) Pname) (استاد) PROF
((تعداد واحد) Unit ,(نام درس) Cname ,(کد درس) C#) (درس) CRS
((نمره) Score ,Pname ,(کد ترم) Term ,S# ,S# ,C# ,(کد) Sec#) (اخذ درس) SEC

(مهندسی کامپیوتر ۱۴۰۱)

۱) فقط دانشجویان که معدل آنها در دروس ۳ واحدی بالاتر از ۱۶ است را لیست می‌کند.

۲) فقط مشخصات دانشجویانی را که دروس ۳ واحدی اخذ کرده‌اند نمایش می‌دهد.

۳) دانشجویانی که معدل بالاتر از ۱۶ هستند و دروس ۳ واحدی را نیز اخذ کرده‌اند.

۴) این امکان پذیر نیست، زیرا از یک دامنه یکسان گرفته نشده است.

پاسخ گزینه ۴ واضح است که ستونهای $\sigma_{avg>16}(stud)$ با ستونهای $\sigma_{Unit=3}(crs)$ یکسان نیست. پس در عبارت داده شده شرط سازگاری رعایت نشده و این عبارت خطای نحوی یا syntax error دارد. پس امکان پذیری نیست.

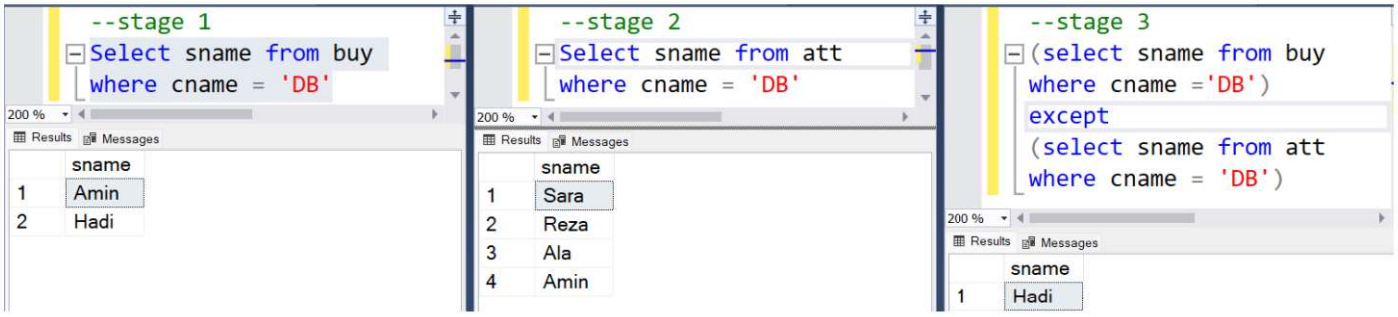
نکته) منظور از خطای نحوی یا syntax error یعنی یک عبارتی که گرامر زبان (یا در اینجا جبر) را رعایت نکرده است. مثلا در فارسی ما نمی توانیم بنویسم. « رفت به مدرسه علی». زیرا در زبان فارسی فعل باید در انتهای جمله باشد. در زبان فارسی می‌گوییم جمله خطای دستوری دارد. در کامپیوتر می‌گوییم خطای نحوی.

1.1.6 پرس‌وجو 6: مثال دیگر از عملگر تفاضل

نام دانشجویانی که درس DB را خریده اند ولی در کلاس آن شرکت نکرده اند.

```
(Select sname from buy where cname = 'DB') except
(Select sname from att where cname = 'DB')
```

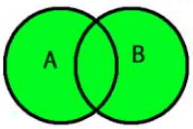
$\Pi_{sname}(\sigma_{cname = 'DB'}(buy)) - \Pi_{sname}(\sigma_{cname = 'DB'}(att))$



1.1.7 پرس‌وجو 7: عملگر اجتماع

نام دانشجویانی که به نوعی با درس پایگاه داده علاقمند بوده‌اند.

A = {1,2,3}
B = {3,4,5}
AUB = {1,2,3,4,5}



(Select sname **from** buy **where** cname = 'DB') **union**
(Select sname **from** att **where** cname = 'DB')

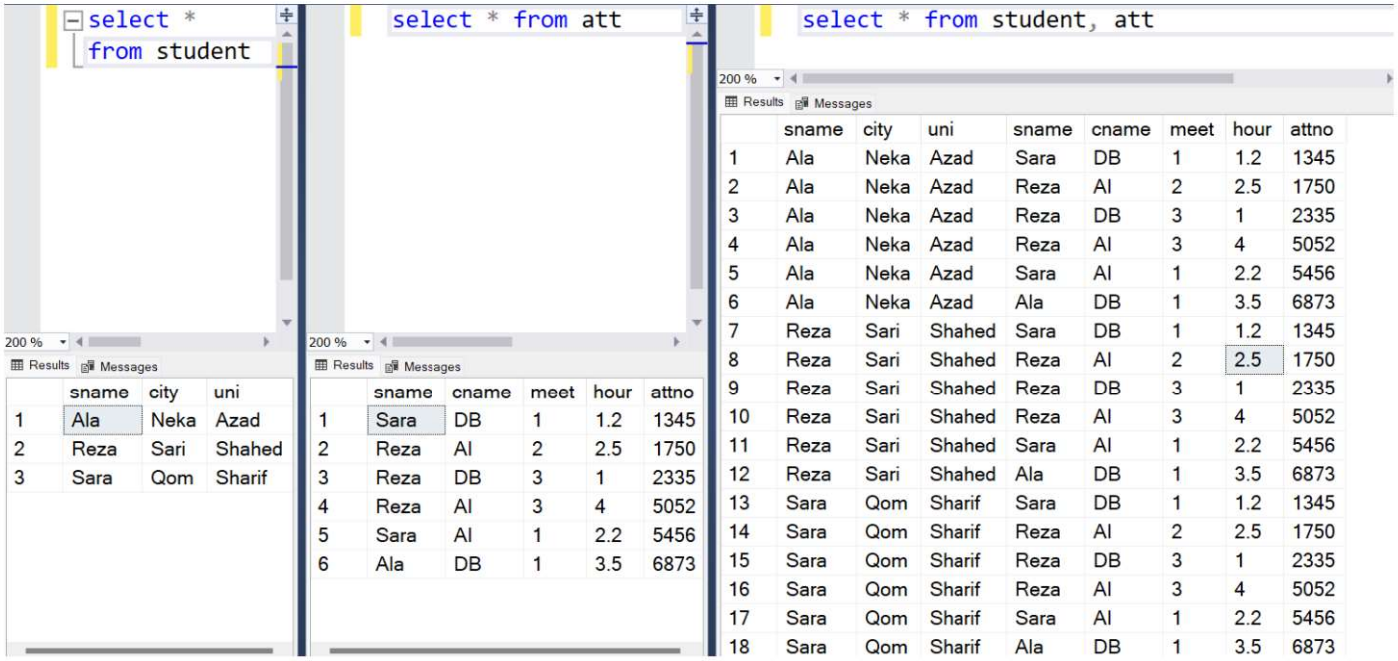
$\Pi_{sname}(\sigma_{cname='DB'}(buy)) \cup \Pi_{sname}(\sigma_{cname='DB'}(att))$

1.1.8 پرس‌وجو8: عملگر ضرب دکارتی

نام و شهر کلیه دانشجویانی که در درس DB حاضر بوده‌اند.

```
select student.sname, student.city
from student, att
where student.sname = att.sname and att.cname = 'DB'
```

$\Pi_{student.sname,\ student.city}(\sigma_{student.sname=att.sname\ \wedge\ att.cname='DB'}(student \times att))$



مشابه پرس‌وجو در اینجا هم باید شرط ضرب دکارتی را داشته باشیم تا سطرهای (تاپل‌های) بی‌معنی حذف شود.

1.2 پرس‌وجوهای معادل

تعریف پرس‌وجوی معادل: دو پرس‌وجو را معادل گویند اگر و فقط اگر همواره روی یک پایگاه داده خروجی یکسان تولید کنند. پرس‌وجوهای معادل با اینکه یک خروجی تولید می‌کنند ولی ممکن است از نظر هزینه اجرایی یا خوانایی، .. با هم متفاوت باشند.

نام دانشجویانی که در کلاس DB شرکت کرده‌اند ولی درس AI را خریده‌اند.

```
Select buy.sname
from buy, Att
WHERE buy.sname = Att.sname and buy.cname = 'AI' and Att.cname = 'DB'
```

$\Pi_{buy.sname}(\sigma_{att.sname=buy.sname \wedge buy.cname='AI' \wedge att.cname='DB'}(buy \times att))$

شکل اول)

$\Pi_{buy.sname}(\sigma_{att.sname=buy.sname \wedge buy.cname='AI' \wedge att.cname='DB'}(buy \times att))$

اول buy در att ضرب شده، بعد به یکباره σ زدیم. در این شکل σ باید روی ۲۴ سطر اعمال (اجرا) شود.

	sname	cname	amount	trno	sname	cname	meet	hour	attno
1	Reza	GS	300	3322	Sara	DB	1	1.2	1345
2	Reza	GS	300	3322	Reza	AI	2	2.5	1750
3	Reza	GS	300	3322	Reza	DB	3	1	2335
4	Reza	GS	300	3322	Reza	AI	3	4	5052
5	Reza	GS	300	3322	Sara	AI	1	2.2	5456
6	Reza	GS	300	3322	Ala	DB	1	3.5	6873
7	Reza	AI	520	3448	Sara	DB	1	1.2	1345
8	Reza	AI	520	3448	Reza	AI	2	2.5	1750
9	Reza	AI	520	3448	Reza	DB	3	1	2335
10	Reza	AI	520	3448	Reza	AI	3	4	5052
11	Reza	AI	520	3448	Sara	AI	1	2.2	5456
12	Reza	AI	520	3448	Ala	DB	1	3.5	6873
13	Ala	AW	320	4423	Sara	DB	1	1.2	1345
14	Ala	AW	320	4423	Reza	AI	2	2.5	1750
15	Ala	AW	320	4423	Reza	DB	3	1	2335
16	Ala	AW	320	4423	Reza	AI	3	4	5052
17	Ala	AW	320	4423	Sara	AI	1	2.2	5456
18	Ala	AW	320	4423	Ala	DB	1	3.5	6873
19	Ala	AI	430	9097	Sara	DB	1	1.2	1345
20	Ala	AI	430	9097	Reza	AI	2	2.5	1750
21	Ala	AI	430	9097	Reza	DB	3	1	2335
22	Ala	AI	430	9097	Reza	AI	3	4	5052
23	Ala	AI	430	9097	Sara	AI	1	2.2	5456
24	Ala	AI	430	9097	Ala	DB	1	3.5	6873

شکل دوم)

$\Pi_{buy.sname}(\sigma_{att.sname=buy.sname \wedge att.cname='DB'}(\sigma_{buy.cname='AI'}(buy) \times att))$

اول روی buy انتخاب زدیم $\sigma_{buy.cname='AI'}(buy)$ بعد ضرب کردیم بعد روی حاصلضرب دوباره انتخاب زدیم. در این حالت حاصلضرب دکارتی فقط ۱۲ سطر دارد.

	sname	cname	amount	trno	sname	cname	meet	hour	attno
1	Reza	AI	520	3448	Sara	DB	1	1.2	1345
2	Reza	AI	520	3448	Reza	AI	2	2.5	1750
3	Reza	AI	520	3448	Reza	DB	3	1	2335
4	Reza	AI	520	3448	Reza	AI	3	4	5052
5	Reza	AI	520	3448	Sara	AI	1	2.2	5456
6	Reza	AI	520	3448	Ala	DB	1	3.5	6873
7	Ala	AI	430	9097	Sara	DB	1	1.2	1345
8	Ala	AI	430	9097	Reza	AI	2	2.5	1750
9	Ala	AI	430	9097	Reza	DB	3	1	2335
10	Ala	AI	430	9097	Reza	AI	3	4	5052
11	Ala	AI	430	9097	Sara	AI	1	2.2	5456
12	Ala	AI	430	9097	Ala	DB	1	3.5	6873

شکل سوم)

$\Pi_{buy.sname}(\sigma_{att.sname=buy.sname \wedge buy.cname='AI'}(buy \times \sigma_{att.cname='DB'}(att)))$

اول روی att انتخاب زدیم، بعد ضرب کردیم. بعد روی حاصلضرب دوباره انتخاب زدیم.

	sname	cname	amount	trno	sname	cname	meet	hour	attno
1	Reza	GS	300	3322	Sara	DB	1	1.2	1345
2	Reza	AI	520	3448	Sara	DB	1	1.2	1345
3	Ala	AW	320	4423	Sara	DB	1	1.2	1345
4	Ala	AI	430	9097	Sara	DB	1	1.2	1345
5	Reza	GS	300	3322	Reza	DB	3	1	2335
6	Reza	AI	520	3448	Reza	DB	3	1	2335
7	Ala	AW	320	4423	Reza	DB	3	1	2335
8	Ala	AI	430	9097	Reza	DB	3	1	2335
9	Reza	GS	300	3322	Ala	DB	1	3.5	6873
10	Reza	AI	520	3448	Ala	DB	1	3.5	6873
11	Ala	AW	320	4423	Ala	DB	1	3.5	6873
12	Ala	AI	430	9097	Ala	DB	1	3.5	6873

شکل چهارم)

$\Pi_{buy.sname}(\sigma_{att.sname=buy.sname}(\sigma_{buy.cname='AI'}(buy) \times \sigma_{att.cname='DB'}(att)))$

اول روی att وbuy هر دو انتخاب زدیم. بعد ضرب کردیم. بعد روی حاصلضرب دوباره انتخاب زدیم. در شکل حاصلضرب دکارتی شش سطر دارد.

	sname	cname	amount	trno	sname	cname	meet	hour	attno
1	Reza	AI	520	3448	Sara	DB	1	1.2	1345
2	Reza	AI	520	3448	Reza	DB	3	1	2335
3	Reza	AI	520	3448	Ala	DB	1	3.5	6873
4	Ala	AI	430	9097	Sara	DB	1	1.2	1345
5	Ala	AI	430	9097	Reza	DB	3	1	2335
6	Ala	AI	430	9097	Ala	DB	1	3.5	6873

نکته: عمل ضرب خیلی پر هزینه است. هر چقدر دیرتر ضرب کنیم و جدول‌هایی که در هم ضرب می‌شوند، کوچکتر باشند بهتر است. در اینجا پرس‌وجوی شکل چهارم از همه سریع تر حساب می‌شود.

نکته: در این مثال تعداد سطرها از ۲۴ به ۶ سطر تغییر کرد. شاید گفته شود که تفاوت زیاد نیست. در عمل جدول‌ها خیلی بزرگتر هستند. مثلا ممکن است ۲۴ هزار سطر به ۶ هزار سطر تغییر کند که کاملا محسوس خواهد بود.

1.3 جبر رابطه‌ای Relational Algebra

از نظر ریاضی هر عبارت جبری از دو عامل تشکیل شده است. الف) مجموعه عملوندها ب)عملگرها . مثلا در درس جبر دبیرستان می‌نوشتیم.

2 + 3 -1 = 4

در جبر دبیرستان عملوندها اعداد بودند و عمگرهای جمع و تفریق و ضرب.. داشتیم. در مهندسی کامپیوتر در درس مدار منطقی هم ما برای تحلیل و طرحی مدارهای منطقی از جبر بول Boolean Algebra استفاده می‌کردیم. مشابه آن در درس پایگاه‌داده برای تحلیل و نوشتن کوئری‌ها ما از جبر رابطه‌ای Relational Algebra استفاده می‌کنیم.

جبر رابطه‌ای	جبر بول	جبر دبیرستان	
رابطه‌ها	عبارت‌های جبر بولی	اعداد	عملوندها
انتخاب select، پرتو project، اجتماع union	AND, OR, NOT, XOR, XNOR,..	+ - × =	عملگرها
درس پایگاه‌داده	درس مدار منطقی	محاسبات	کاربرد

1.4 رابطه Relation

رابطه در واقع عملوند جبر رابطه‌ای است. ما در جبر رابطه‌ای با استفاده از عملگرها، یک سری عملیات روی رابطه‌ها انجام می‌دهیم.
تعریف رابطه: به هر زیر مجموعه دلخواه از حاصلضرب دکارتی دنباله‌ای از دامنه‌ها یک رابطه گفته می‌شود.
مثال) فرض کنید دانشجویان فقط سه نفر باشند. D1={Reza, Ali, Negin}. می‌گوییم دامنه دانشجویان ما سه عضو دارد. فرض کنید فقط دو شهر داشته باشیم D2={Bam, Qom} می‌گوییم دامنه ما شامل این دو شهر است. حاصلضرب دکارتی این دو دامنه همه شش حالت ممکن را شامل می‌شود. D1 × D2={(Reza,Qom), (Reza, Bam), (Ali, Qom), (Ali, Bam), (Negin, Qom), (Negin, Bam)}
هر زیر مجموعه‌ای از این ضرب دکارتی را یک رابطه می‌گویند. مثلاً دو زیرمجموعه S1 و S2 رابطه هستند.
R1 = {(Reza, Bam), (Reza, Qom), (Negin, Bam)}
R2 ={(Reza, Qom), (Ali, Bam), (Negin, Bam)}
نکته: دقت کنید که ما در رابطه ما به معنا کار نداریم مثلا در R1 می‌توانیم تفسیر شود که Rezaهم از شهر Bam هست و هم از شهر Qom . در عمل Reza نمی‌تواند از دو شهر باشد و این رابطه شاید بی‌معنا باشد. ولی ما به بدون توجه به معنای آن با توجه به این که زیرمجموعه‌ای از یک حاصلضرب دکارتی است می‌گوییم که یک رابطه است.

در ریاضیات ما از سمبلها به جای مقادیر واقعی استفاده می‌کنیم. اجازه بدهید، در جدول زیر سمبلها ریاضی را جایگزین مقادیر با معنا کنیم. به عبارت دیگر معنا را حذف کنیم.

دامنه‌های روبرو را خواهیم داشت. D1={a,b,c} D2={1,2} D3={x,y}

Student					
Sname	D1	City	D2	uni	D3
Reza	a	Bam	1	Sharif	X
Ali	b	Qom	2	UoT	Y
Negin	c				

حاصلضرب دکارتی D1 در D2 به شکل زیر خواهد بود.

D1 x D2 = {(a,1), (a,2),(b,1),(b,2),(c,1),(c,2) }

D2 x D3 = {(1,x),(1,y),(2,x),(2,y)}

D1 x D2 x D3 = {(a,1,x), (a,2,x),(b,1,x),(b,2,x),(c,1,x),(c,2,x), (a,1,y), (a,2,y),(b,1,y),(b,2,y),(c,1,y),(c,2,y)}

هر زیر مجموعه دلخوا از ضرب دکارتی یک رابطه است. مثلا

R1= {(a,1), (b,2), (c,1)} R2={} R3= {(a,1,x), (b,1,y),(c,2,y)}

تعریف تاپِل tuple : به هر یک از اعضای رابطه یک تاپِل می گویند.

سوال) آیا موارد زیر یک رابطه محسوب می‌شوند؟

R4={(a,1),(b,1),(2,y)}

R5={(a,2), (b,1,x), (b,2,y)}

R6={((a,2),y), (a,2,x)}

R7= {(1,a), (2,b), (1,c)}

R8= {(1,a), (2,b), (c,1)}

جواب) R4 نیست زیرا دو تاپل اول از D1xD2 است. و تاپل سوم از D2xD3 است.

R5 نیست زیرا تاپل اول دوتایی (درجه2) و دو تاپل آخر سه‌تایی (درجه3) است. پس مطمئناً از یک ضرب دکارتی نیستند.

R6 نیست زیرا تاپل تودرتو نمی‌تواند عضوی از یک ضرب دکارتی باشد.

R7 هست. تمامی تاپلها عضوی از D2xD1 هستند. دقت کنید همان R1 است با این تفاوت که ترتیب ضرب دامنه‌ها و اعضای تاپل عوض شده‌است.

1.5 نگاشت جبر رابطه‌ای و جدول

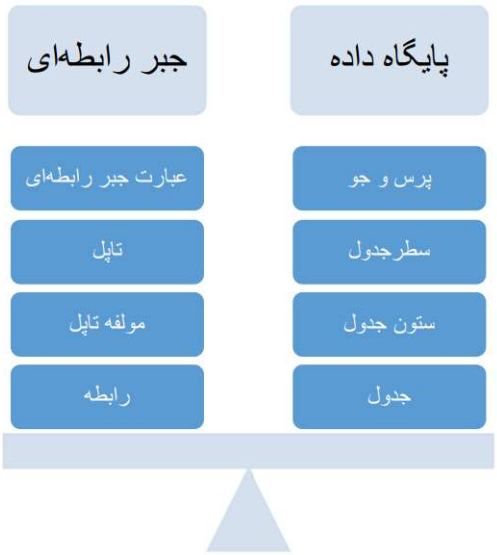
جدول پایگاه‌داده در واقع پیاده‌سازی یک رابطه است. جدول زیر معادل این رابطه است.

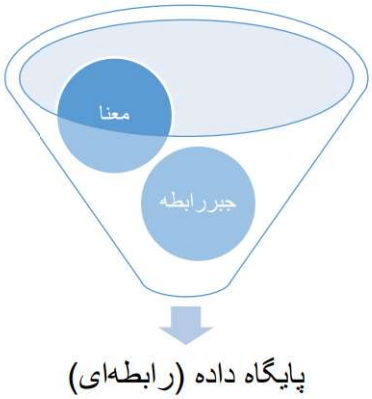
R3= {(a,1,x), (b,1,y),(c,2,y)}

Student					
sname	D1	City	D2	uni	D3
Reza	a	Bam	1	Sharif	X
Ali	b	Bam	1	UoT	Y
Negin	c	Qom	2	UoT	Y

فقط کافی است که به جای سمبلها از اسامی با معنا استفاده کنیم.

Student = {(Reza, Bam, Qom), (Ali, Bam, UoT) , (Negin, Qom , UoT)}





1.6 اسکما schema

تعریف: اسکما رابطه relation schema عبارت است از لیست مولفه‌ها (خصوصیتها) و دامنه‌های متناظر

مثلا student(sname, city, uni) اسکما است و Student = {(Reza, Bam, Sharif), (Ali, Bam, UoT) , (Negin, Qom , UoT)}

یک نمونه از این رابطه یا relation instance است.

مثال) در شکل زیر R1 و R2 هر دو یک اسکما دارند و دو نمونه از یک اسکما هستند. ولی R3 چون دو ستون دارند اصلاً اسکما متفاوتی دارد.

R1		
sname	city	uni
Reza	Bam	Sharif
Ali	Bam	UoT
Negin	Qom	UoT

R2		
sname	city	uni
Reza	Qom	UoT
Negin	Qom	Sharif

R3	
sname	city
Reza	Bam
Ali	Bam
Negin	UoT

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

نکته: در یک رابطه نمی‌توانیم تاپل تکراری داشته باشیم
نکته: ترتیب تاپل‌ها (سطرها) در رابطه (جدول) مهم نیست. مثلا دو جدول زیر با هم برابر هستند.

R1		
sname	city	uni
Reza	Bam	Sharif
Ali	Bam	UoT
Negin	Qom	UoT

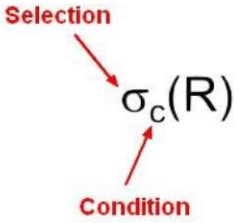
R2		
sname	city	uni
Negin	Qom	UoT
Reza	Bam	Sharif
Ali	Bam	UoT

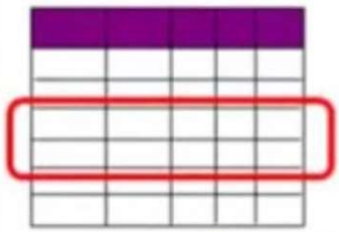
1.7 عملگرهای اصلی رابطه‌ای

شش عملگر اصلی جبر رابطه‌ای عبارت است از: انتخاب، پرتو، ضرب، اجتماع، تفاضل و نام گزاری مجدد. ثابت شده که هر عبارت متصور در جبر رابطه‌ای را می‌توان با این عملگرهای اصلی نوشت.

1.7.1 عملگر انتخاب / select / σ سیگما

از یک رابطه (جدول) تعدادی از تاپلها (سطرها) که یک مؤلفه (ستون) آن شرط خاصی دارد را انتخاب می‌کند.





select

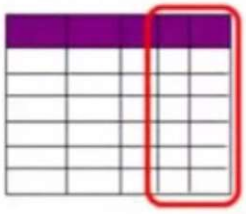
$\sigma_{\text{sname = "ali"}}(\text{student})$

نکته) خروجی عملگر رابطه‌ای یک رابطه است. پس می‌توان دو عملگر انتخاب را به شکل زیر با هم ترکیب کرد. مثلا عبارت زیر سطرهایی که هم سطرهایی که اسم دانشجو ali باشد و دانشجو از شهر bam باشد را در خروجی نمایش می‌دهد.

$\sigma_{\text{city = "bam"}}(\sigma_{\text{sname = "ali"}}(\text{student}))$

1.7.2 عملگر پرتو Project / پای Π

تنها مولفه‌ها (ستونهای) مشخص شده را به خروجی می‌دهد.



project

```
select city, uni from student
Π city, uni (student)
```

همانطور که قبلا گفتیم خروجی یک عملگر رابطه‌ای یک رابطه است. پس می‌توان به شکل زنجیره‌ای عملگرها را استفاده کرد. مثلا عبارت زیر نام دانشجویانی اهل Bam را به خروجی می‌فرستد.

```
select sname from student
    where city = 'Bam'
```


این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

$\Pi_{sname}(\sigma_{city = "Bam"}(student))$

نکته: برای خواندن یک عبارت رابطه‌ای از داخلی‌ترین پرانتز شروع کنید و به سمت بیرون حرکت کنید.

این دو عملگر σ و π خیلی پرکاربرد و مهم هستند. در پرس‌وجو نویسی خیلی استفاده می‌شوند.

قواعد پر کاربرد σ و π	
به طور کلی π خاصیت جابجایی ندارد	$\pi_{L1}(\pi_{L2}(R)) <> \pi_{L1}(\pi_{L2}(R))$
اگر کلیه صفتهای θ در L باشد. σ و π با هم خاصیت جابجایی دارند.	$\sigma_{\theta}(\pi_{L}(R)) = \pi_{L}(\sigma_{\theta}(R))$
σ خاصیت جابجایی دارد.	$\sigma_{\theta 1}(\sigma_{\theta 2}(R)) = \sigma_{\theta 2}(\sigma_{\theta 1}(R)) = \sigma_{\theta 1 \wedge \theta 2}(R) = \sigma_{\theta 1}(R) \cap \sigma_{\theta 1}(R)$
	$\sigma_{\theta 1}(\sigma_{\theta 2}(\sigma_{\theta 3}(R))) = \sigma_{\theta 1 \wedge \theta 2 \wedge \theta 3}(R)$

1.7.2.1.1.1 تست مهندسی کامپیوتر سراسری ۱۳۸۲

۱. عبارت زیر در چه صورت صحیح است؟

$$\sigma_p\left(\Pi_{a_1,a_2,\ldots,a_n}(R)\right) = \Pi_{a_1,a_2,\ldots,a_n}(\sigma_p(R))$$

σ: انتخاب

Π:تصویر

(مهندسی کامپیوتر ۱۳۸۲)

۱) شرط P حداقل برای یک سطر رابطه R برقرار باشد.

۲) شرط P فقط ستون‌های $a_1, \ldots, a_n$ را در برگیرد.

۳) شرط P همه ستون‌های $a_1, \ldots, a_n$ را در برگیرد.

۴) هیچ کدام

پاسخ گزینه ۲ اگر شرط P ستونهایی غیر از a1, ..., an را شامل بشود. آنگاه سمت چپ مساوی خطای نحوی خواهد داشت. پس شرط P باید فقط ستون‌های a1, ..., an را در بگیرد. گزینه ۲ درست است.

1.7.3 ضرب کارتزین (دکارتی)

ضرب کارتزین عمگری است برای ترکیب اطلاعات دو جدول (رابطه). مثلا در شکل زیر student × att محاسبه شده‌است.

```
select *
from student
```

	sname	city	uni
1	Ala	Neka	Azad
2	Reza	Sari	Shahed
3	Sara	Qom	Sharif

```
select * from att
```

	sname	cname	meet	hour	attno
1	Sara	DB	1	1.2	1345
2	Reza	AI	2	2.5	1750
3	Reza	DB	3	1	2335
4	Reza	AI	3	4	5052
5	Sara	AI	1	2.2	5456
6	Ala	DB	1	3.5	6873

```
select * from student, att
```

	sname	city	uni	sname	cname	meet	hour	attno
1	Ala	Neka	Azad	Sara	DB	1	1.2	1345
2	Ala	Neka	Azad	Reza	AI	2	2.5	1750
3	Ala	Neka	Azad	Reza	DB	3	1	2335
4	Ala	Neka	Azad	Reza	AI	3	4	5052
5	Ala	Neka	Azad	Sara	AI	1	2.2	5456
6	Ala	Neka	Azad	Ala	DB	1	3.5	6873
7	Reza	Sari	Shahed	Sara	DB	1	1.2	1345
8	Reza	Sari	Shahed	Reza	AI	2	2.5	1750
9	Reza	Sari	Shahed	Reza	DB	3	1	2335
10	Reza	Sari	Shahed	Reza	AI	3	4	5052
11	Reza	Sari	Shahed	Sara	AI	1	2.2	5456
12	Reza	Sari	Shahed	Ala	DB	1	3.5	6873
13	Sara	Qom	Sharif	Sara	DB	1	1.2	1345
14	Sara	Qom	Sharif	Reza	AI	2	2.5	1750
15	Sara	Qom	Sharif	Reza	DB	3	1	2335
16	Sara	Qom	Sharif	Reza	AI	3	4	5052
17	Sara	Qom	Sharif	Sara	AI	1	2.2	5456
18	Sara	Qom	Sharif	Ala	DB	1	3.5	6873

1.7.4 عملگر اجتماع / U / Union

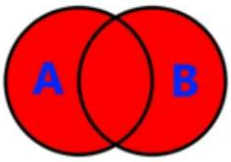
اجتماع دو رابطه شامل تمامی تاپل‌های دو رابطه شده و ضمناً تاپل‌های تکراری حذف می‌شود.

مثال) A = {(a,1,y),(a,2,x),(a,2,y),(b,1,x),(c,1,y),(c,2,x)}

B={(b,2,x),(b,2,y),(c,1,x),(c,1,y),(a,1,x),(a,1,y),(a,2,x)}

A U B = {(a,1,y),(a,2,x),(a,2,y),(b,1,x),(c,1,y),(c,2,x),
(b,2,x),(b,2,y),(c,1,x),~~(c,1,y)~~,(a,1,x),~~(a,1,y)~~,~~(a,2,x)~~}

A U B= {(a,1,y),(a,2,x),(a,2,y),(b,1,x),(c,1,y),(c,2,x),(b,2,x),(b,2,y),(c,1,x), (a,1,x)}



A ∪ B

نکته: اجتماع هر رابطه با خودش برابر با همان رابطه است

نکته: در اجتماع هم باید شرط سازگاری رعایت شده باشد. یعنی اگر دو رابطه با هم اجتماع می‌شوند، اسکما این دو رابطه یکسان باشد. یعنی تعداد ستونها شان یکی باشد و دامنه‌های ستون‌های متناظر یکی باشد.

نکته) $\sigma_{c\vee d}(R) = \sigma_{c}(R) \cup \sigma_{d}(R)$

1.7.4.1.1.1 تست مهندسی کامپیوتر سراسری ۱۴۰۰ ۱

۴۱. هم ارزی‌های جبر رابطه‌ای زیر را در نظر بگیرید. این هم ارزی‌ها ممکن است همواره درست باشند، در بعضی شرایط درست باشند، یا همواره نادرست باشند. در این عبارت‌ها، R یک رابطه (Relation)، ci ها شرط‌هایی بر روی R و ai ها زیر مجموعه‌هایی از صفت‌های R هستند.

کدام هم ارزی همواره درست است؟

(مهندسی کامپیوتر ۱۴۰۰)

۱) $\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$

۲) $\pi_{a1}(\pi_{a2}(R)) \equiv \pi_{a2}(\pi_{a1}(R))$

۳) $\pi_{a1}(\sigma_{c1}(R)) \equiv \sigma_{c1}(\pi_{a1}(R))$

۴) $\pi_{a1}(\pi_{a2}(R)) \equiv \pi_{a1}(R)$

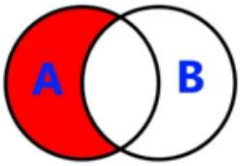
پاسخ گزینه ۱ می‌دانیم که σ خاصیت جابجایی دارد پس گزینه ۱ درست است. جدول قواعد پرکاربرد σ و π در زیر آورده شده است.

قواعد پر کاربرد σ و π	
به طور کلی π خاصیت جابجایی ندارد	$\pi_{L1}(\pi_{L2}(R)) <> \pi_{L1}(\pi_{L2}(R))$
اگر کلیه صفتهای θ در L باشد. σ و π با هم خاصیت جابجایی دارند.	$\sigma_{\theta}(\pi_{L}(R)) = \pi_{L}(\sigma_{\theta}(R))$
σ خاصیت جابجایی دارد	$\sigma_{\theta 1}(\sigma_{\theta 2}(R)) = \sigma_{\theta 2}(\sigma_{\theta 1}(R)) = \sigma_{\theta 1 \wedge \theta 2}(R)$
	$\sigma_{\theta 1}(\sigma_{\theta 2}(\sigma_{\theta 3}(R))) = \sigma_{\theta 1 \wedge \theta 2 \wedge \theta 3}(R)$

بحث) گزینه‌های دیگر را به راحتی با استفاده از مثال نقض می‌توان رد کرد. برای گزینه ۲ سمت راست می‌تواند $\pi_{x,y}(\pi_x(R))$ باشد. این خطای نحوی دارد زیرا $\pi_x(R)$ ستون y ندارد. در حالیکه $\pi_x(\pi_{x,y}(R)) = \{(100), (200)\}$ گزینه ۲ در این مثالی که زدیم درست نیست پس رد می‌شود. برای گزینه ۳ $\sigma_{x=100}(\pi_y(R))$ خطای نحوی دارد زیرا π_y ستون x ندارد. در حالیکه $\pi_y(\sigma_{x=100}(R)) = \{(3)\}$ خواهد شد. پس گزینه ۳ هم با این مثال رد می‌شود. برای گزینه ۴ $\pi_x(R) = \{(100), (200)\}$ و $\pi_x(\pi_y(R))$ خطای نحوی دارد. پس گزینه ۴ هم رد می‌شود. تنها گزینه ۳ باقی ماند که باید به عنوان پاسخ انتخاب شود.

R	
x	y
100	3
200	4

1.7.5 عملگر تفاضل / set difference / -



A - B

A-B تمامی تاپلهایی که در A هستند ولی در B نیستند.

مثال) A = {(a,1,y),(a,2,x),(a,2,y),(b,1,x),(c,1,y),(c,2,x)}

B={(b,2,x),(b,2,y),(c,1,x),(c,1,y),(a,1,x),(a,1,y),(a,2,x)}

A - B = {(a,2,y),(b,1,x),(c,2,x)}

عملوندهای تفاضل هم باید سازگار باشند.

نکته: عملگر انتخاب دارای خاصیت توزیع‌پذیری بر تفاضل است.

$\sigma_T(A-B) = \sigma_T(A) - \sigma_T(B) = \sigma_T(A) - B$

نکته) اگر در صورت پرس‌وجو حرف نفی هیچ بود این می‌تواند نشانه‌ای باشد که در عبارت جبری باید از – استفاده کنیم.

مثال) کتابهای نویسنده‌ای به نام احمدی که توسط هیچ عضو بالای ۱۸ سال در هیچ زمانی به امانت گرفته نشده‌است. (فناوری اطلاعات ۱۴۰۲)

$$\Pi_{\langle ISBN \rangle}\left(\sigma_{(Author='Ahmadi')}(Book)\right) - \Pi_{\langle ISBN \rangle}\left(Borrowed \bowtie \sigma_{(Age>18)}(Member)\right)$$

1.7.6 عملگر نام گزاری مجدد/ rename / ρ رو

یک عبارت جبری را به عنوان ورودی گرفته و روی آن نامگذاری مجدد انجام می‌دهد. در سالهای مختلف کنکور و منابع مختلف از نوتیشن‌های مختلفی استفاده شده که در ادامه با هم مرور می‌کنیم.

مثال) ρ(STUDENT_NAME, $\prod_{NAME}$(STUDENT))

در عبارت بالا از رابطه (جدول) student مولفه (ستون) name انتخاب شده و نام آن را به student_name تغییر می‌دهیم.

مثال) ρ (S(a,b,c1), R)

نام رابطه R را به S تغییر داده و نام ستون‌های آن را به a, b, c1

مثال) ρ (T, S X R)

حاصلضرب دکارتی S X R را T نام‌گذاری می‌کند.

مثال) ρ_T (S X R)

حاصلضرب دکارتی S X R را T نام‌گذاری می‌کند.

نامگذاری مجدد در نگاه اول شاید عملگر مهمی به نظر نرسد ولی این عملگر جزو عملگرهای اصلی است و بعضی از پرس‌وجوها را بدون آن نمی‌توان نوشت. به مثال زیر دقت کنید.

مثال) کوئری بنویسید که دانشجویان هم دانشگاه با علی را در خروجی نمایش دهد.

گام اول) دانشگاه علی را باید پیدا کنیم.

$$\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$$

خروجی عبارت $\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$ یک رابطه است برابر با {(malek)}. زیرا به طور کلی خروجی عملگرهای σ و π یک رابطه است. برای انتخاب سطرهایی که دانشگاه آن دانشگاه malek است. نمی‌توان از عبارت زیر استفاده کرد. زیرا خطای نحوی syntax error دارد.

$$\sigma_{univ = \pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)}$$

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

زیرا univ یک خصوصیت (نام ستون)(attribute) است. آن را باید با یک مقدار مقایسه کرد نه رابطه (جدول)!
$\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$ = {(malek)}پس عبارت از نظر جبری غلط (بی معنی) است.

برای اینکه ستونهای جدول $\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$ را با جدول Student مقایسه کنیم. باید این دو (جدول) رابطه را در هم ضرب کنیم. و از حاصلضرب سطرهایی را انتخاب کنیم که student.univ با ستون univ عبارت $\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$ برابر باشد. به هنگام نوشتن شرط ما نمی‌توانیم ستون univ عبارت $\pi_{univ}\left(\sigma_{sname='Ali'}(studnet)\right)$ را در شرط بنویسیم مگر اینکه آن را با عمگر ρ برای آن نام B را انتخاب کرده باشیم.
نکته: معادل عملگر ρ در SQL دستور AS است.

Student		
sname	city	univ
Ali	Sari	malek
Reza	Qom	Malek
Ahmad	Bam	Payam

?
Univ
malek

$\pi_{univ}(\sigma_{sname='Ali'}(studnet))$

Student			?
sname	city	univ	univ
Ali	Sari	malek	Malek
Reza	Qom	Malek	Malek
Ahmad	Bam	Payam	Malek

$\sigma_{student.univ=?.univ}(student \times \pi_{univ}(\sigma_{sname='Ali'}(studnet)))$

Student X B			
Student			B
sname	city	univ	univ
Ali	Sari	malek	Malek
Reza	Qom	Malek	Malek
Ahmad	Bam	Payam	malek

$student \times \rho_B(\pi_{univ}(\sigma_{sname='Ali'}(studnet)))$

B
Univ
malek

$\rho_B(\pi_{univ}(\sigma_{sname='Ali'}(studnet)))$

Student			B
sname	city	univ	univ
Ali	Sari	malek	Malek
Reza	Qom	Malek	Malek

$\sigma_{student.univ=B.univ}(student \times \rho_B\left(\pi_{univ}(\sigma_{sname='Ali'}(studnet))\right))$

2 کلید key

2.1 ابرکلید super key

تعریف ابر کلید: ترکیبی از یک یا چند attribute یا مشخصه (ستون) که منحصر بفرد باشد.

مثلا فرض کنید جدول زیر تعریف شده است.

ش دانشجویی	نام	نام خانوادگی	آدرس	تلفن	نام پدر	ش ش	محل صدور	ش ملی

سوال کدام از ترکیبهای زیر ابرکلید است؟

نام و نام خانوادگی- نام و نام‌خانوادگی و آدرس- نام، نام‌خانوادگی، آدرس ، شماره تلفن -نام و نام‌خانوادگی، شماره شناسنامه - شماره ملی - شماره ملی و شماره دانشجویی- شماره ملی، شماره دانشجویی، نام و نام‌خانوداگی- ش ش و محل صدور -شماره دانشجویی به تنهایی یا با هر ترکیبی - شماره‌ملی به تنهایی یا با هر ترکیبی-

2.2 کلید کاندید candidate key

تعریف کلید کاندید: ابرکلید کمینه را کلید کاندید می‌گویند. منظور از کلید کمینه کلید با کمترین خصیصه نیست بلکه کلیدی است که خاصیت زائد نداشته باشد.

سوال کدام یک از ترکیبهای زیر می تواند کلید کاندید باشد؟

نام و نام خانوادگی و شماره ملی

شماره ملی

شماره دانشجویی و نام

شماره شناسنامه و محل صدور

در مثلا قبلی ما سه تا کلید کاندید داریم. 1. شماره ملی 2. شماره دانشجویی 3. شماره شناسنامه و محل صدور

سوال) در جدول زیر کدام یک از عبارت صحیح است.

A	B	C
A1	B1	C1
A1	B2	C1
A2	B2	C2

خصیصه A ابر کلید است؟ خصیصه B ابر کلید است؟ خصیصه C ابرکلید است؟

ترکیب A و B ابرکلید است؟ ترکیب A و B کلید کاندید است؟ ترکیب A و C ابر کلید است؟

ترکیب ABC ابر کلید است؟ ترکیب ABC کلید کاندید است؟

2.3 کلید اصلی Primary Key

تعریف کلید اصلی: یکی از کلیدهای کاندید که طراح آن را به عنوان کلید اصلی انتخاب می‌کند.

توصیه: بهتر است کلید کاندید با کمترین تعداد خصیصه‌ها به عنوان کلید اصلی انتخاب شود.

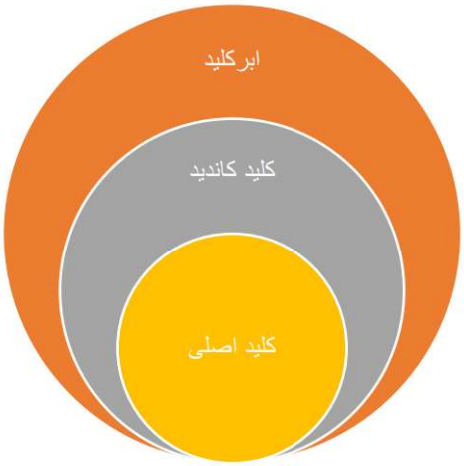
نکته: هر جا گفتند کلید منظور کلید اصلی است.

2.4 کلید ثانویه Secondary Key

تعریف کلید اصلی: یکی از کلیدهای کاندید که طراح می‌تواند آن را به عنوان کلید ثانویه در کنار اصلی انتخاب کند.

انتخاب کلید ثانویه الزامی نیست. در حالیکه بر اساس قانون جامعیت هر جدول باید یک کلید اصلی داشته‌باشد.

2.5 رابطه بین کلیدها



در سوالات کنکور یا در محیط عملی وقتی می گویند کلید منظور کلید اصلی است.

2.6 کلید خارجی Foreign Key

سه نوع کلید اول ماهیتاً برای ایجاد تمایز بین سطرهای جدول (یا تاپل‌ها رابطه) استفاده می‌شود. در حالی که کلید خارجی برای ایجاد ارتباط در بین جدول‌ها (یا رابطه‌ها) استفاده می‌شود.

مثال) فرض کنید دو جدول student (sno, sname, tno) و prof (pno, tname) را به شکل زیر داریم.

خصیصه pno در prof و خصیصه sno در student کلید اصلی است. یعنی یکتاست و قدرت تمایز دارد.

Student

Tno	Sname	sno
P3	Afshin	S1
P2	Sima	S2
P2	Ahmad	S3
P2	Asad	S4

کلید خارجی
ارجاع دهنده
فرزند child

کلید اصلی

Refer

Prof

pname	pno
Sara	P1
Amin	P2
Ali	P3

کلید اصلی
مرجع
والد parent

Student.tno و prof.pno هم معنا و هم نوع هستند. هر دو کد استادی هستند. خصیصه student.pon به خصیصه prof.pno ارجاع refer می‌دهد. به student.tno و student ، ارجاع‌دهنده یا referencing گفته می‌شود. به prof.pno و prof، مرجع یا referenced گفته می‌شود.

تعریف کلید خارجی: اگر خصیصه (ها) ارجاع دهنده به یک **کلید کاندید** ارجاع دهد، کلید خارجی گفته می‌شود.

نکته: به بیان دیگر خصیصه مرجع باید در مرجع قدرت تمایز داشته باشد.

سوال) کاربرد کلید خارجی چیست؟

پاسخ) ارتباط Relationship

سوال) آیا باید اسم کلید خارجی با اسم کلید اصلی مرجع یکسان باشد؟

پاسخ) خیر

سوال) آیا کلید خارجی می‌تواند مقدار تکراری داشته‌باشد؟

پاسخ) بلی. در مثال قبلی استاد Sima، Ahamad و Asad یک نفر است، Amin با کد استادی P2. همانطور که می‌بینید کد استادی امین به عنوان کلید خارجی تکرار شده است.

سوال) آیا کلید خارجی می‌تواند null باشد؟

پاسخ) بلی. کلید خارجی می‌تواند null باشد. مثلا دانشجویی که تازه ثبت‌نام کرده و هنوز استاد راهنما ندارد.

سوال) اگر A=null و B=null آیا A=B ؟

پاسخ) خیر

سوال) آیا مرجع کلید خارجی می‌تواند در خود جدول باشد؟

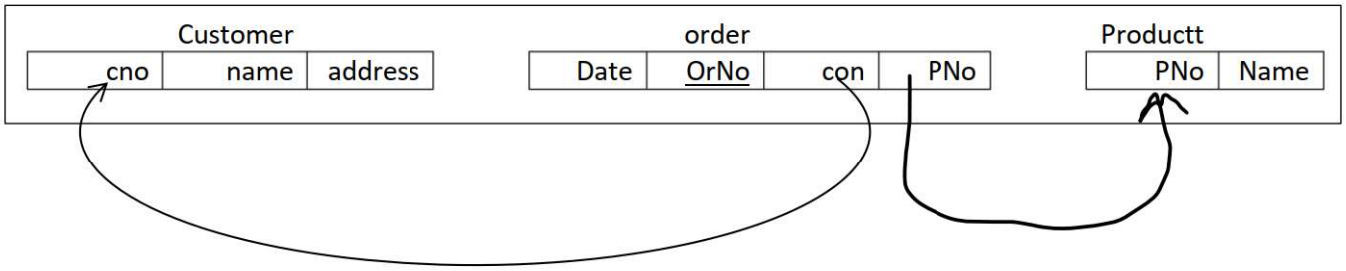
پاسخ)بلی. مثلاً ستون کد ملی پدر به ستون کد ملی ارجاع داده و پدر افراد را مشخص می‌کند.

سوال) آیا می‌تونیم کلید خارجی ترکیبی (بیش از چند ستون) هم داشته باشیم؟

پاسخ) بلی. مثلا قبل از رواج کدملی از ترکیب شماره شناسنامه و محل صدور شناسنامه به عنوان کلید خارجی استفاده می‌کردیم.

سوال) آیا می‌تونیم بیش از یک کلید خارجی داشته باشیم؟

پاسخ) بلی. مثلا زیر جدول order دو تا کلید خارجی دارد که به دو جدول ارجاع می‌دهد.



تست- کنکور IT 1388- کلید خارجی ۸۳

اگر FK کلید خارجی رابطه R1 متناظر با نامزد کلیدی CK از رابطه R2 باشد، کدام عبارت صحیح است؟

۱) FK نمی‌تواند Null باشد. ۲)روابط R1 و R2 متفاوت هستند.
۳) هر مقدار غیر Null برای FK برابر یک مقدار CK است. ۴) هر مقدار غیر NULL برای CK برابر یک مقدار FK است.

پاسخ گزینه ۳ FK می‌تواند null باشد. گزینه ۱ رد می‌شود. FK می‌تواند به CK در خود همان رابطه ارجاع دهد، پس R1 و R2 می‌توانند یکسان باشند پس گزینه ۲ رد می‌شود. FK فقط می‌تواند مقادیر کلید مرجع یعنی CK را بگیرید. گزینه ۳ صحیح است. دقت کنید گزینه ۴ برعکس گزینه ۳ و غلط است. کلید مرجع یا همان CK مستقل از FK هر مقداری را می‌تواند بگیرد.

تست ۳۷- کنکور کامپیوتر ۱۳۸۸-

کدام عبارت، در مورد کلید خارجی (Foreign Key) در مدل رابطه‌ای صحیح است؟

1) کلید خارجی یکی از (Candidate Key) های همان رابطه است.
2) کلید خارجی بایست کلید اصلی رابطه دیگری باشند. (نباید Alternate Key باشد.)
3) کلید خارجی باید خصیصه ساده (Simple Attribute) باشد.م
4) کلید خارجی یک رابطه می‌تواند متناظر با مقادیر Candidate Key همان رابطه باشد.

پاسخ گزینه ۴ کلید خارجی چون وظیفه تمایز ندارد لازم نیست که Candidate Key باشد. گزینه ۱ رد. کلید خارجی باید یک کلید کاندید ارجاع بدهد یا متناظر با آن باشد. لزومی ندارد که این کلید کاندید، کلید اصلی باشد می‌تواند Alternate Key باشد. پس این گزینه ۲ رد می‌شود. کلید خارجی می‌تواند می‌تواند ترکیبی از چند خصیصه باشد. مثال شماره شناسنامه و محل صدور را به یاد آورید. پس گزینه ۳ هم رد می‌شود. دیدیم که کلید خارجی می‌تواند به کلید کاندید همان رابطه (یا جدول) ارجاع دهد (یا متناظر با آن باشد) پس گزینه ۴ صحیح است. پایان.

3 قوانین جامعیت

جامعیت (یکپارچگی) یا integrity یعنی مقادیر پایگاه‌داده صحیح باشد. برای حفظ جامعیت (صحیح بودن)، رابطه‌ها باید از یک سری قواعد پیروی کنند. مثلا

1) در بانکهای ایران شما نمی‌توانید بیشتر از 10 میلیون در روز کارت به کارت کنید.
2) دانشجویی که در مقطع ارشد ثبت‌نام میشه باید مدرک کارشناسی داشته باشد.
3) هر ایرانی باید کدملی داشته باشد.

بعضی از قوانین یا محدودیت‌ها اختصاصی است و از یک پایگاه‌داده به پایگاه‌داده دیگر عوض می‌شود. مثلا سقف کارت به کارت ممکن است تغییر کند. ولی ما در پایگاه‌داده رابطه‌ای قوانین و محدودیت‌های داریم که همیشه در هر پایگاه باید رعایت شود. به این قواعد، قوانین سه‌گانه جامعیت گفته می‌شود.

جمع بندی قوانین جامعیت پایگاه داده				
	نام انگلیسی	نام فارسی	درباره	
هر رابطه‌ای باید به تنهایی درست باشد	Intra-Relation	درون رابطه‌ای	رابطه	قانون اول
کلید اصلی نباید نال باشد	Entity	موجودیت	کلید اصلی	قانون دوم
ارجاع ساختاری: مرجع کلید خارجی باید کلید کاندید باشد. ارجاع محتوایی: تغییر کلید خارجی و کلید کاندید باید هماهنگ باشد	Referential	ارجاع	کلید خارجی	قانون سوم
کلید اصلی نباید نال باشد. مؤلفه تاپل باید عضوی از دامنه باشد	Domain	دامنه	کلید اصلی+دامنه	قانون دوم (توسعه)

3.1 قانون اول

نباید رابطه (جدول) به گونه‌ای تعریف شود که جبر رابطه‌ای نقض نشود. در نتیجه:

- تاپل تکراری ممنوع است.
- تاپلها با درجه‌های متفاوت نمی‌تونیم داشته باشیم.
- تاپل‌های تودرتو غیر مجاز است.
- یک رابطه حتما باید کلید اصلی داشته باشد.

3.2 قانون دوم

توسعه یافته قانون دوم در برخی از کتابها مطرح شده است. در اکثر کتابها "مؤلفه تاپل باید عضوی از دامنه باشد" در ذیل قانون اول آمده است.

3.3 قانون سوم

این قانون دو بخش دارد، ساختاری و محتوایی.

- ساختاری: اگر خصیصه‌ای به عنوان کلید خارجی به خصیصه‌ی دیگری ارجاع می‌دهد. مرجع خود باید کلید کاندید باشد و قدرت تمایز داشته باشد.
- محتوایی:
  - کلید خارجی اگر عوض شد باید هماهنگ با کلید مرجع باشد.
  - اگر کلید مرجع عوض شد باید هماهنگ با کلید ارجاع دهنده باشد.

مثال و بحث) customer (cname, city, street) deposit(cname, bname, ano, balance)

نکته: در کتابها و کنکور اگر زیر یک خصیصه underline گذاشته شد یعنی کلید اصلی است.

سوال) نام یک مشتری در customer تغییر کرده ولی در deposit تغییر نکرده است. آیا قانونی نقض شده است؟

پاسخ) بلی. قانون سوم. referential

سوال) آیا می‌توان در جدول deposit برای مشتری حساب باز کرد که در customer نیست؟

پاسخ) خیر. قانون سوم بخش محتوایی نقض می‌شود. دقت کنید قانون سوم درباره کلید خارجی است.

سوال) آیا می توان از جدول customer مشتری را حذف کرد که هنوز در deposit حساب دارد؟

پاسخ) خیر. قانون سوم نقض بخش محتوایی نقض می‌شود.

سوال) آیا می توان مشتری بی نام ثبت نام کند؟

پاسخ) خیر. قانون دوم نقض می‌شود.

4 عملگرهای فرعی

نکته: هر عبارت جبر رابطه‌ای باید را می‌توان با عملگرهای اصلی نوشت.

سوال) پس کاربردهای عملگرهای فرعی چیست؟

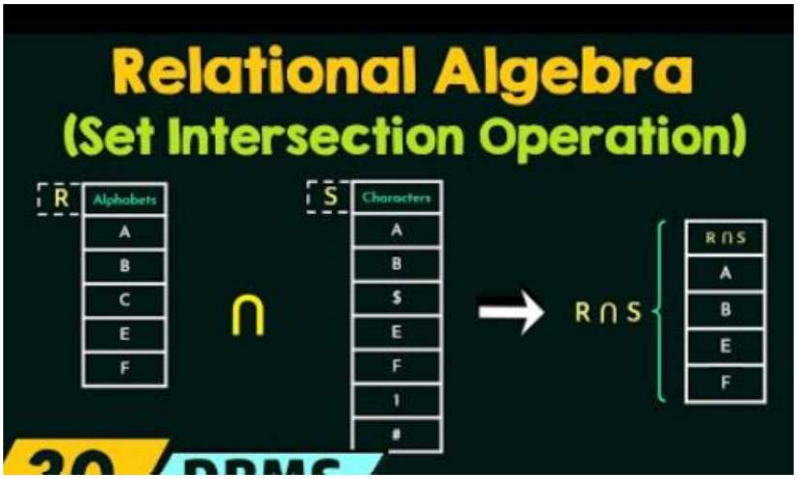
نکته: یکی از نکاتی که در کنکور سوال می آید. نوشتن عملگرهای اصلی با استفاده از عملگرهای فرعی است.

4.1 عملگر اشتراک A∩B intersection

تعریف اشتراک: اگر A و B دو رابطه باشند، اشتراک آن دو عبارت است از تمامی تاپلهایی که هم در A هست و هم در B.

نکته: برای عملگر اشتراک باید شرط سازگاری داشته باشیم. یعنی تاپلها هم درجه و هم دامنه. درباره‌ی تمام عملگرهای مجموعه‌ای یعنی اجتماع، اشتراک و تفاضل باید شرط سازگاری داشته باشیم.

تعریف دقیق و ریاضی شرط سازگاری: A(P1, P2, ..., Pn) و B(Q1, Q2, ..., Qn) سازگار هستند اگر و تنها اگر `domain(Pi) = domain(Qi) for 1<= i <= k.`



سوال) عملگر اشتراک را با عملگرهای اصلی بنویسید.

A-(A-B) = B – (B-A)=A ∩ B = ((A ∪ B) - (A - B)) - (B - A)

نکته) $\sigma_{c \wedge d}(R) = \sigma_c(\sigma_d(R)) = \sigma_c(R) \cap \sigma_d(R)$

4.2 عملگر انتساب assignment

خیلی شبیه انتساب در زبانهای برنامه‌نویسی است. وقتی می‌خواهیم مقدار یک متغیر را عوض کنیم.

Count = count + 1 یا count ← count + 1

برای تغییر یک رابطه کاربرد دارد با استفاده از rename استفاده می‌شود.

مثال) می خواهیم به رابطه Student(sname, city, uni) یک دانشجوی جدید اضافه کنیم؟

Student ← student U {'hadi', 'Tehran', 'Tehran'}

مثال) می خواهیم دانشجوهایی دانشگاه sharif را از رابطه (جدول) دانشجویان حذف کنیم؟

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

شکل اول) (student) Student ← student – σ uni = 'sharif'

شکل دوم) (student) Student ← student – σ uni = 'sharif'

سوال) این دو شکل با هم چه فرق دارند؟

4.3 عملگر الحاق طبیعی ⋈ natural join

ضرب کارتزین X ستونهای دو جدول را به هم ملحق می کرد. الحاق طبیعی هم با یک تفاوت کوچک همین کار را می کند. به همین خاطر سمبل آن خیلی شبیه سمبل ضرب است. هر جا بگویند عملگر الحاق منظور عملگر الحاق طبیعی است.

برای محاسبه عمل الحاق طبیعی کارهای زیر را انجام دهید.

روش اول)

1. دو رابطه را در هم ضرب کنید
2. از ستونهای مشترک فقط یکی حفظ شود.
3. سطرهایی که در ستون مشترک هم مقدار نیستند را حذف کنید.

در روش اول باید عملیات پرهزینه ضرب دکارتی را انجام داده و سپس بعضی از سطرها و ستونهای را حذف کنیم. که وقت‌گیر و پرهزینه است. به جای آن از روش دوم استفاده می‌کنیم.

تعریف سطرهای الحاق‌پذیر: سطرهایی که در ستونهای مشترک هم مقدار هستند.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Mary	1257	Human Resources

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

روش دوم)

1. ستون‌های مشترک را یکبار بنویسید.
2. سطرهای الحاق پذیر را با هم الحاق کرده و بنویسید.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Mary	1257	Human Resources

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee ⋈ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

مثال) عملگر الحاق طبیعی را برای employee و dept با عملگرهای اصلی بنویسید.

4.4 عملگر شبه الحاق semi join ⋉

از خروجی الحاق طبیعی ستونهای رابطه سمت راست را حذف کنید.

مثال)

customer

cname	City
Ali	Tehran
Reza	Tabriz
hamid	Zanjan
Ahmad	Mashad

Borrow

cname	Branch
ali	Bahar
reza	Banafshe
hamid	Bahar
ali	karimkhani

Customer⋈ borrow

Cname	City	Branch
Ali	tehran	bahar
Ali	tehran	Kari.
Reza	Tabriz	banafseh
Hamid	zanjan	bahar

Customer ⋉ borrow

Cname	City
Ali	Tehran
Reza	Tabriz
Hamid	Zanjan

نکته: بعد از حذف ستون branch دو سطر تکراری خواهیم داشت که یکی حذف می‌شود.

4.5 عملگر الحاق خارجی راست ⟖

به خروجی الحاق سطرهای الحاق ناپذیر سمت راست را اضافه کرده و برای ستون‌های سمت چپ مقدار null بذارید.

مثال)

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

Dept

DeptName	Manager
Sales	Harriet
Production	Charles

Employee ⟖ Dept

Name	EmpId	DeptName	Manager
Sally	2241	Sales	Harriet
Harriet	2202	Sales	Harriet
ω	ω	Production	Charles

4.6 علمگر الحاق خارجی چپ ⟕

به خروجی الحاق، سطرهای الحاق ناپذیر سمت چپ را اضافه کرده و برای ستون‌های سمت راست مقدار null بذارید.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

Dept

DeptName	Manager
Sales	Harriet
Production	Charles

Employee ⟕ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	ω
Sally	2241	Sales	Harriet
George	3401	Finance	ω
Harriet	2202	Sales	Harriet
Tim	1123	Executive	ω

4.7 عملگر الحاق خارجی کامل ⟗

به خروجی الحاق،1 سطرهای الحاق ناپذیر را اضافه کرده و برای ستون‌های بی مقدار null بگذارید.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

Dept

DeptName	Manager
Sales	Harriet
Production	Charles

Employee ⟗ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	ω
Sally	2241	Sales	Harriet
George	3401	Finance	ω
Harriet	2202	Sales	Harriet
Tim	1123	Executive	ω
ω	ω	Production	Charles

جمع‌بندی علمگرهای الحاق				
	جابجایی	پیش شرط	سمبل	عملگر
الحاق تمام سطرها، تمامی ستونها	دارد	ندارد	A X B	ضرب دکارتی
الحاق سطرهای الحاق پذیر، حذف ستونهای تکراری	دارد	ندارد	$A⋈B$	الحاق طبیعی
Select زدن روی خروجی ضرب	دارد	ندارد	$A⋈_θB$	تتا جوین
حذف ستونهای B از خروجی الحاق	ندارد	ندارد	$A⋉B$	شبه الحاق
اضافه کردن سطرهای الحاق‌ناپذیر B به خروجی الحاق	ندارد	ندارد	$A⟕B$	الحاق خارجی راست
اضافه کردن سطرهای الحاق‌ناپذیر A به خروجی الحاق	ندارد	ندارد	$A⟖B$	الحاق خارجی چپ
اضافه کردن سطرهای الحاق‌ناپذیر A و B به خروجی الحاق	دارد	ندارد	$A⟗B$	الحاق خارجی کامل

نکته: مقایسه تعداد سطرهای عملگرهای الحاق $A⟗B \le A⟕B , A⟖B \le A⟗B \le A⋉B \le A⋈B \le A \text{ X } B$

نکته: اگر ستونهای A و B یکسان باشد. $A⋈B = A⋉B = B⋉A = A \cap B$

نکته: برای به خاطر سپردن تفاوت انواع الحاق خارجی می‌توان گفت که **علامت گوش**] به هر طرف که باشد، الحاق‌ناپذیرهای آن طرف به نتیجه الحاق طبیعی اضافه می‌شود.

الحاق ناپذیر راست	عملگر	الحاق ناپذیر چپ
	$A⋈B$	
	$A⟕B$	
	$A⟖B$	
	$A⟗B$	

4.8 عملگرها و کلیدها

سوال) دو رابطه A با کلید کاندید Ak و B با کلید کاندید Bk داریم. وقتی عملیاتی روی A و B انجام می‌دهیم درباره کلید کاندید نتیجه عملیات چه می‌توان گفت؟

سوال) درباره کلید A U B چه می توان گفت؟

الف) Ak یا Bk به تنهایی الزاماً کلید نخواهند بود. اثبات با مثال نقض می‌بینید نه Ak یعنی ستون Q و نه Bk یعنی ستونS به تنهایی برای می‌توانید A ∪ B کلید نیستند.

A ∪ B

S	Q
s1	q1
s1	q2
~~s1~~	~~q1~~
s2	q1

B

S	Q
s1	q1
s2	q1

کلید(تمایز)

A

S	Q
s1	q1
s1	q2

کلید(تمایز)

ب) ترکیب Ak و Bk کلید کاندید است. برای اثبات نادرستی این جمله به مثال زیر توجه کنید. در این مثال ترکیب کلیدها، کلید کاندید نخواهد بود زیرا کلید کاندید نمی‌تواند خصوصیت زیادی داشته باشد. در A ∪ B هر کدام از ستونهای Q و S تمایز ایجاد کرده و کلید کاندید هستند.

A ∪ B

S	Q
s1	q2

کلید(تمایز) کلید(تمایز)

B

S	Q
s1	q2

کلید(تمایز) کلید(تمایز)

A

S	Q
s1	q2

کلید(تمایز) کلید(تمایز)

د) تمام خصیصه‌های A و B با هم ابر کلید A ∪ B هستند. این گزاره بدیهی است و واقعیت جدیدی به ما نمی‌گوید.

سوال) درباره کلید اشتراک A و B چه می‌توان گفت؟

Ak و به Bk به تنهایی کلید A ∩ B هستند. اثبات) از برهان خلف استفاده می‌کنیم. می‌گوییم فرض کنید که Ak کلید A ∩ B نباشد. یعنی دو سطر در A ∩ B هست که قدرت تمایز آن دو را ندارد. چون A ∩ B ⊂ A پس این دو سطر در A هم هست. پس Ak قدرت تمایز دو سطر در A را هم نداشته و نمی‌تواند کلید A باشد! پس Ak باید کلید A ∩ B باشد.

سوال) درباره کلید کاندید تفاضل A-B چه می‌توان گفت؟

A-B حالت خاصی از اشتراک است. می‌دانیم A-B=A ∩ B′ پس می‌توان نتیجه گرفت که Ak کلید A-B است.

سوال) درباره کلید کاندید ضرب دکارتی AxB چه می‌توان گفت؟

ترکیب Ak و Bk کلید است. در مثال زیر ترکیب کلید Ak یعنی ستون Q و کلید Bk یعنی ستون T قدرت تمایز داشته و کلید است.

A x B

T	S	R	Q
t1	s1	r1	q1
t2	s1	r1	q1
t1	s1	r2	q2
t2	s1	r2	q2

B

T	S
t1	s1
t2	s1

کلید(تمایز)

A

R	Q
r1	q1
r1	q2

کلید(تمایز)

سوال) درباره کلید کاندید الحاق طبیعی A⋈ B چه می‌توان گفت؟

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

اگر دو جدول ستون مشترک نداشته باشند A⋈B=AxB و از همان نتیجه‌گیری ضرب دکارتی می‌توان استفاده کرد. وقتی ستون مشترک داشته باشیم اگر Ak در ستون مشترک نباشد کلید کاندید الحاق طبیعی خواهد بود. مثلا در مثال زیر ستون R در A کلید است و قدرت تمایز دارد ولی در الحاق طبیعی A و B این قدرت را نداشته و الحاق پذیر نیست.

A

Q	R
q1	r1
q2	r2

الحاق(پیوند) پذیر

B

S	T
r1	s1
r1	s2

A ⋈ B

Q	R	S
q1	r1	s1
q1	r1	s2

کلید در نتیجه	عملیات
Ak یا Bk به تنهایی کلید کاندید است	اشتراک
Ak کلید کاندید است.	A-B
Ak یا Bk الزاما کلید اجتماع نخواهند بود	اجتماع
ترکیب Ak و Bk ابر کلید است	ضرب دکارتی
	الحاق طبیعی بدون ستون مشترک
Ak کلید کاندید است، به شرط نبودن در ستون مشترک Bk کلید کاندید است، به شرط نبودن در ستون مشترک	الحاق طبیعی

تست مهندسی کامپیوتر سراسری ۱۳۸۲

فرض کنید R1 وR2 دو رابطه باشند و R3= R1 UNION R2 کدامیک از گزاره‌های زیر صحیح است؟

۱) کلید اصلی R3 اجتماع کلیدهای اصلی R1 و R2 است. ۲) کلید اصلی R3 اجتماع تمام خصیصه‌های R1 و R2 است.
۳) کلید اصلی R3 کلید اصلی R1 یا کلید اصلی R2 است. ۴) کلید اصلی R3 تقاطع تمام خصیصه‌های(ستون‌های) R1 و R2 است.

پاسخ گزینه ۲و ۴ بر اساس آنچه درس داده شد گزینه ۱ و ۳ رد می‌شود. گزینه ۲ و ۴ معادل هم هستند. زیرا با توجه به شرط سازگاری خصیصه‌های R1 و R2 یکی باید باشند. پش اجتماع این خصیصه‌ها با تقاطع (اشتراک) آنها یکسان است.

اجتماع خصیصه‌های R1 و R2 = اشتراک خصیصه‌های R1و R2 = خصیصه‌های R1 = خصیصه‌های R2 = خصیصه‌های R3
با توجه به این که ما در رابطه نمی‌توانیم تاپل تکراری داشته باشیم. تمام خصیصه‌های R3 ابرکلید بدیهی R3 است. ولی دقت کنید ابرکلید ممکن است خصوصیت اضافه داشته باشد و در نتیجه شاید کلید اصلی و کلید کاندید نباشند. پس اگر خیلی دقیق باشیم گزینه ۲ و ۴ هم رد می‌شوند. هر چند در کنکور بهتر است یکی از این دو گزینه را انتخاب می‌کردید.

4.9 عملگر تقسیم ÷

عملگر تقسیم جزو عملگرهای فرعی است.

4.9.1 کاربرد تقسیم

سوال) کجا از تقسیم استفاده می کنیم؟ در چه پرس و جوهایی از تقسیم استفاده می‌کنیم؟

پاسخ)
دانشجویانی که در همه دوره‌ها ثبت‌نام کرده‌اند.
مشتریانی که در هر شعبه‌ای حساب دارند
مشترکینی که تمام قسمتهای سریال زخم کاری را دیده‌اند.

در پرس‌وجوهایی که از واژه همه (یا هر) (یا تمام) استفاده شده از عملگر تقسیم استفاده می‌شود. معمولاً این همه در وسط جمله است و غیرقابل حذف.

نکته واژه همه (یا هر) نباید قابل حذف باشد. در مثالهای زیر واژه همه به راحتی قابل حذف است.

همه دانشجویانی که در کلاس پایگاه‌داده ثبت‌نام کرده‌اند == دانشجویانی که در کلاس پایگاداده ثبت‌نام کرده‌اند.

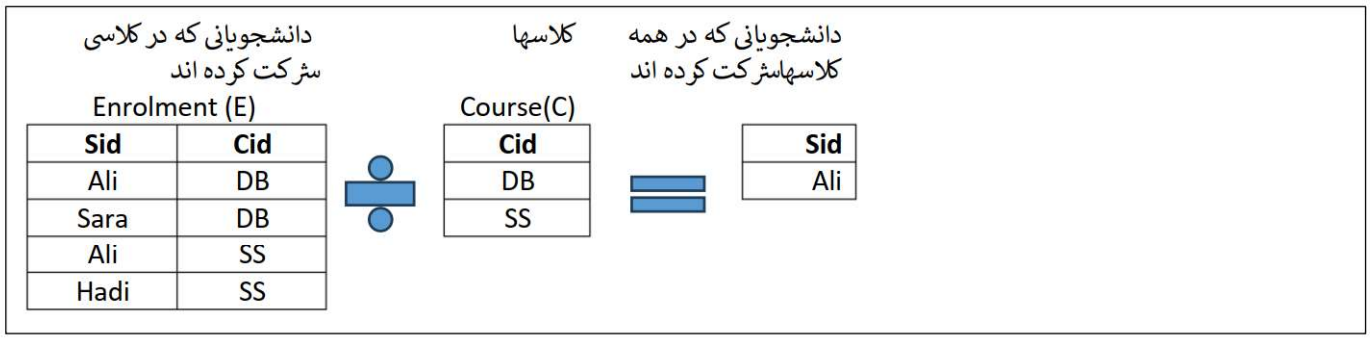
همه اعضایی که از کتابخانه کتاب به امانت گرفته‌اند. == اعضایی که از کتابخانه کتاب به امانت گرفته‌اند.

سوال) آیا در کوئری «دانشجویانی که هیچ درسی نباشد که نگرفته باشند » باید از تقسیم استفاده کنیم؟

پاسخ) بلی. «دانشجویانی که هیچ درسی نباشد که نگرفته باشند» معادل «دانشجویانی که همه درسها را گرفته‌اند». این نکته در فناوری اطلاعات سراسری ۱۳۹۳ آمده بود.

سوال) خروجی عملگر تقسیم چیست؟ E ÷ C

پاسخ) سطرهایی (تاپلهایی)از جدول(رابطه) E که با همه سطرهای(تاپلهای) جدول (رابطه) C متناظر باشد در خروجی ظاهر می‌شود. به مثال زیر دقت کنید. جدول E دو ستون Sid و ستون Cid دارد که به ترتیب نام دانشجو و درسی که در آن شرکت کرده را نشان می‌دهد. جدول C یک ستون Cid داشته که نام کورسها را نمایش می‌دهد.



سوال) آیا می‌توان هر رابط (جدول) R1 را بر هر رابطه‌ی R2 تقسیم کرد؟

پاسخ) خیر. عمل تقسیم وقتی امکان پذیر است که همه خصوصیت‌ها (ستونهای) R2 در R1 باشد. یعنی R2 ⊆ R1 . در مثال بالا E بر C تقسیم می‌شود زیرا خصوصیاتی C زیرمجموعه خصوصیت‌های E هستند. {Sid, Cid} ⊆ {Cid} .

نکته) تقسیم را می‌توان به شکل روبرو هم نوشت $E \div C = \frac{E}{C}$

سوال) آیا می‌توان C را بر E تقسیم کرد؟

پاسخ) خیر. در E خصوصیت sid را داریم که در C نیست پس C زیر مجموعه E نیست و تقسیم تعریف نشده است.

تست مهندسی کامپیوتر ۱۳۸۹- رابطه R(a, b, c) و دو عبارت جبری زیر را درنظر بگیرید:

$$Q_1: \Pi_{b,c}\left(\sigma_{b=c}(R)\right)$$

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

$$Q_2: \Pi_{a,b}(\sigma_{a=b}(R))$$

کدام عبارت همواره تعریف‌شده است؟
(مهندسی کامپیوتر 1389)

1) $Q_1 \div Q_2$ 2) $Q_1 \bowtie Q_2$

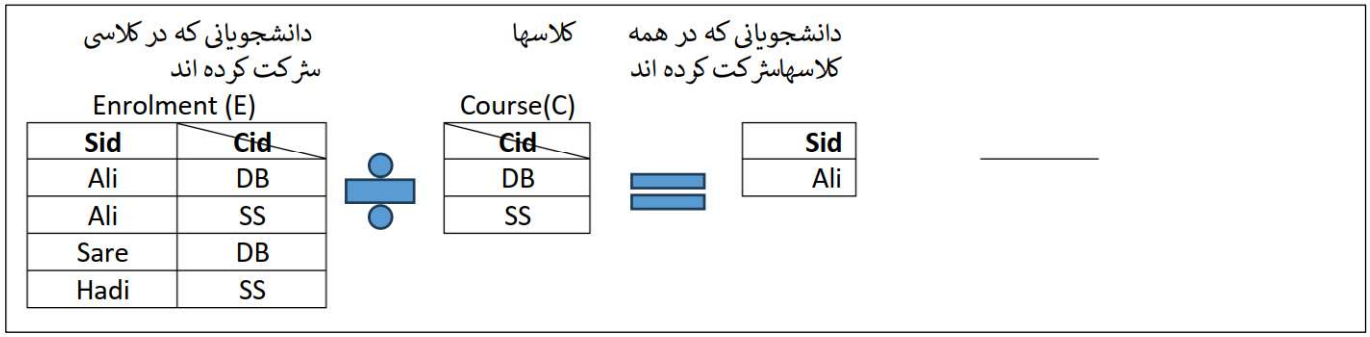
3) $Q_1 \cap Q_2$ 4) موارد 1 و 2

پاسخ گزینه ۲ خصوصیات Q1 یعنی {b,c} زیرمجموعه خصوصیاتQ2 {a,b} نبوده و Q1÷ Q2 تعریف نشده است گزینه ۱ رد می‌شود. گزینه ۴ رد می‌شود. چون شامل گزینه ۱ می‌شود. دو گزینه ۲ و ۳ باقی می‌ماند. با فرض اینکه سواد کافی بری انتخاب بین گزینه ۲ و ۳ را نداریم. باید بین این دو گزینه یکی را انتخاب کنید. این تست را به طور کامل بعداً حل میکنم. در اینجا فقط می‌خواهیم ببنیم به چه راحتی دو گزینه را می‌توان رد کرد.

4.9.2 وجه تسمیه

سوال) چرا به این عملیات تقسیم گفته می‌شود؟

پاسخ) به دلیل شباهتهایی که به تقسیم اعداد صحیح داریم.



فرض کنید تاپل‌ها (سطرها) را بتوان به شکل ضرب بنویسیم. مثلاً Ali × DB ≡ (Ali, DB) یعنی Ali درس DB را شرکت کرده‌است. تاپلهای جدول E را روی کسر با هم جمع کنید و تاپلهای جدول C را در مخرج بنویسید. با فاکتور گیری، ضریب Ali را می‌توان با مخرج ساده کرد ولی دو کسر دیگر این ساده سازی امکان‌پذیر نیست. در نتیجه می‌توان تصور کرد که نتیجه تقسیم Ali می‌شود.

$$\frac{E}{C} = \frac{(Ali \times DB + Ali \times SS) + Sara \times DB + Hadi \times SS}{(DB + SS)} = \frac{(Ali \times (DB + SS))}{(DB + SS)} + \frac{Sara \times DB}{(DB + SS)} + \frac{Hadi \times SS}{(DB + SS)}$$
$$= Ali + \emptyset + \emptyset = Ali$$

شباهت ۱) در اعداد صحیح همه تقسیم‌ها تعریف نشده است. مثلا تقسیم های زیر تعریف نشده است. $\frac{4}{0}$
شباهت ۲) در تقسیم اعداد صحیح عامل مشترک در نتیجه حذف می‌شود، در تقسیم جبر رابطه‌ای ستونهای مشترک.
شباهت ۳) هر چه جدول E بزرگتر بود یعنی تعداد شرکت‌کنندگان در کلاسها بیشتر بود، تعداد افرادی که در همه کلاسها شرکت کرده بودند بیشتر می‌شد. این شبیه تقسیم است که هر چه صورت کسر بزرگتر باشد نتیجه کسر بزرگتر می‌شود.
شباهت ۴) هر چه جدول C کوچکتر، تعداد کلاسهای کمتر و تعداد افرادی که در همه کلاسها شرکت کرده بودند بیشتر می‌شد. این مثل این است که بگوییم در کسر هر چه مخرج کوچکتر باشد نتیجه کسر بزرگتر می‌شود.

مثال) در این مثال می‌بینید که هر چه مخرج کسر بزرگتر باشد نتیجه تقسیم بزرگتر می‌شود.
وقتی مخرج کسر یک جدول تک سطری است نتیجه تقسیم ۴ سطر دارد.
وقتی مخرج کسر یک جدول دو سطری است نتیجه تقسیم ۳ سطر دارد. سه منطقه Manhattan، Brooklyn و Queens دو گونه درخت honey locust و American linden را دارد و در خروجی دومین تقسیم ظاهر می‌شود.
وقتی مخرج کسر یک جدول سه سطری است نتیجه تقسیم ۱ سطر دارد. فقط منطقه Manhattan سه گونه درخت honey locust، American linden و pin oak را دارد و در خروجی سومین تقسیم ظاهر می‌شود.

Division (/)

BORO	SPC_COMMON
Brooklyn	honeylocust
Brooklyn	American linden
Brooklyn	London planetree
Manhattan	honeylocust
Manhattan	American linden
Manhattan	pin oak
Queens	honeylocust
Queens	American linden
Bronx	honeylocust

/ | SPC_COMMON: honeylocust | = | BORO: Brooklyn, Manhattan, Queens, Bronx

/ | SPC_COMMON: honeylocust, American linden | = | BORO: Brooklyn, Manhattan, Queens

/ | SPC_COMMON: honeylocust, American linden, pin oak | = | BORO: Manhattan

4.9.3 پیاده‌سازی تقسیم با عملگرهای اصلی

سوال) عملگر تقسیم را با عملگرهای اصلی چگونه پیاده‌سازی می‌کنیم؟

پاسخ) به سختی!! 🙁 🙁

با یک مثال توضیح می‌دهم. می‌خواهیم کوئری بنویسیم که دانشجویانی که در همه کلاسهای شرکت کرده‌اند را بدهد. به ترتیب زیر عمل می‌کنیم. دو جدول Y و Z را در نظر بگیرید. از تفاضل این دو مجموعه می‌توان به جدول X رسید که شامل دانشجویانی است که در همه کلاس‌ها شرکت کرده‌اند.

همه دانشجویان
Z
$\Pi_{sid}(E)$

Sid
Ali
Sara
Hadi

تفاضل

دانشجویانی که در بعضی از کلاسهاشرکت نکرده‌اند
Y
?

Sid
Sara
Hadi

=

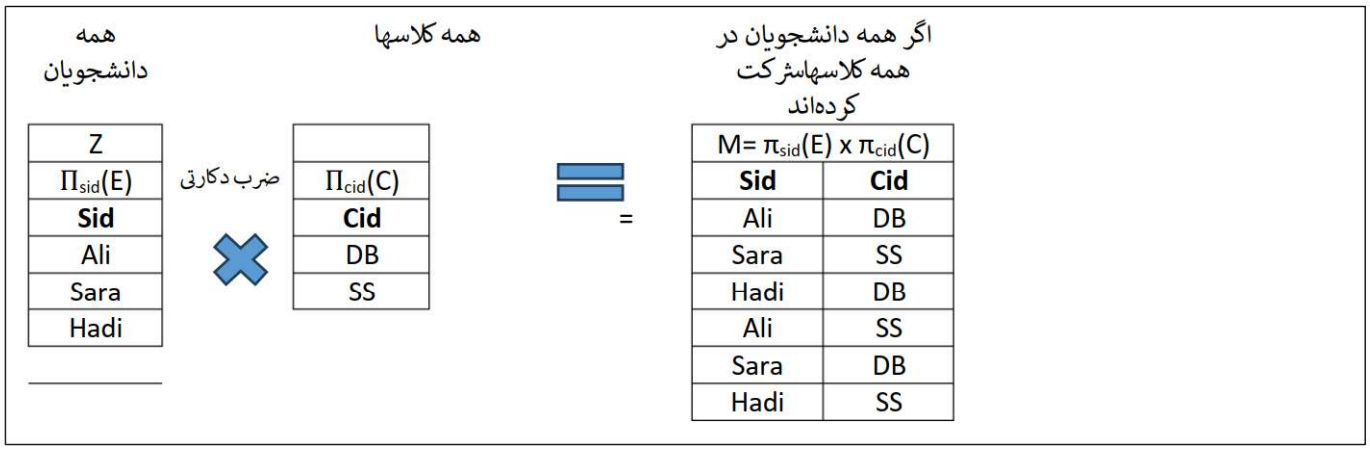
دانشجویانی که در همه کلاس‌هاشرکت کرده‌اند
X= Z-Y

Sid
Ali

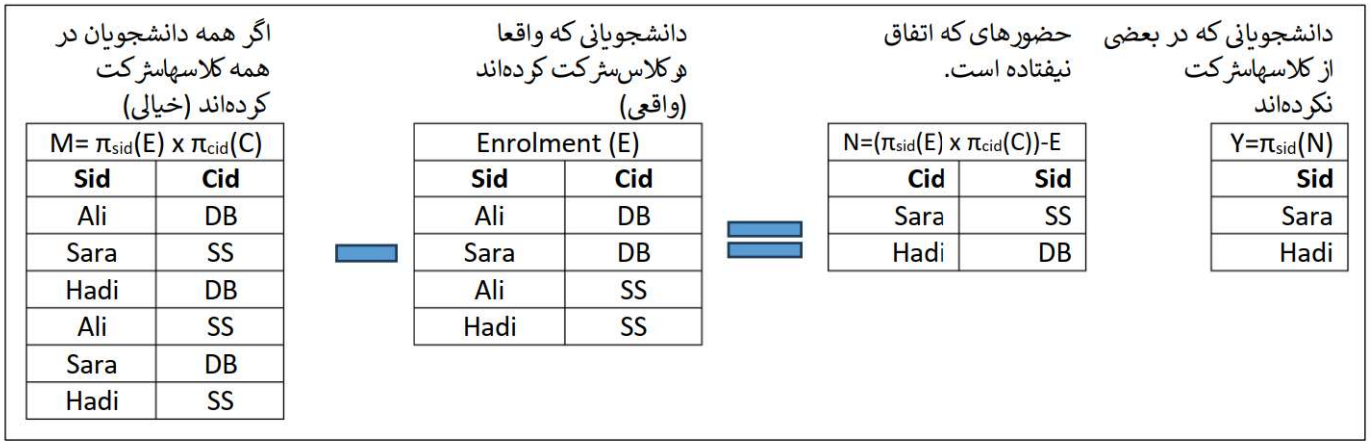
Sid	Cid
Ali	DB
Sara	DB
Ali	SS
Hadi	SS

جدول Z به راحتی با یک عملیات پرتو روی جدول E به دست می‌آید. ولی جدول Z به این راحتی نیست. و بقیه راه حل به این جدول می‌پردازد.

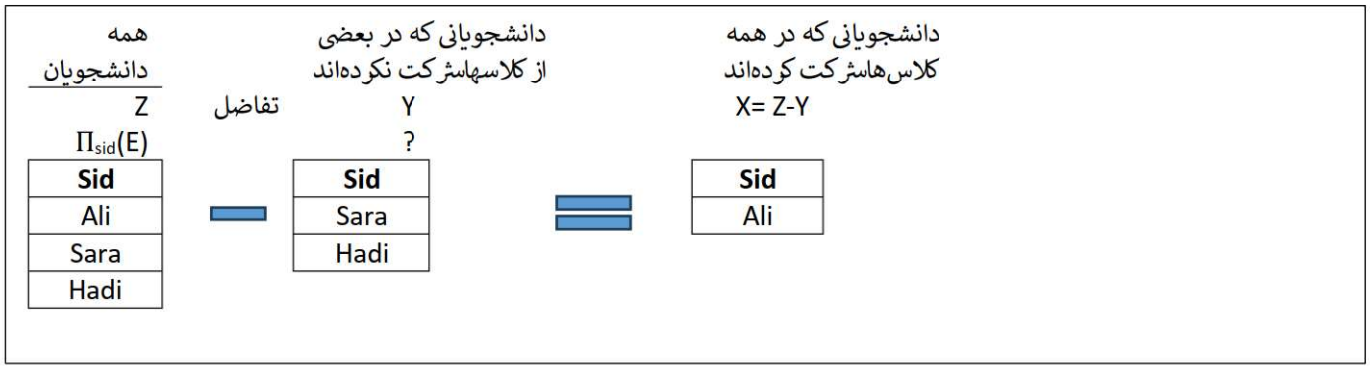
جدول M حالتی را نشان می‌دهد که همه دانشجویان در همه کلاسها شرکت کرده‌اند و با ضرب دکارتی همه دانشجویان در همه کلاسها به دست می‌آید. دقت کنید محتوی جدول M *خیالی* است. یعنی ما***تصور*** می‌کنیم که همه دانشجویان در همه کلاسهای شرکت کرده‌اند.



اگر جدول E را از جدول Mکم کنیم حضورهای که می‌توانست باشند ولی اتفاق نیفتاده است به دست می‌آید. مثلا Sara می‌توانست در SS شرکت کند که نکرده است. یا Hadi می‌توانست در DB شرکت کند که نکرده است. در آخرین گام برای به دست آوردن نام دانشجویانی که در بعضی از کلاسها شرکت نکرده‌اند کافی است از عملگر پرتو π روی این جدول استفاده کنیم و به جدول Y برسیم!



در پایان کافی است از Z همه دانشجویان، Y دانشجویانی که در بعضی از کلاسها شرکت نکرده‌اند را کم کنیم و به جواب نهایی برسیم. Ali تنها دانشجویی است که در همه کلاسها شرکت کرده است.



مراحل راه حل در فرمول زیر نمایش داده شده است. همانطور که می‌بینید کوئری تنها شامل عملگرهای پرتو π و تفاضل – و ضرب دکارتی × که همه جزو عملگرهای اصلی جبر رابطه‌ای هستند. دقت کنید مثل همیشه برای تحلیل (درک) کوئری از داخلی ترین پرانتز شروع کرده به سمت بیرون حرکت می‌کنیم.

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

$$\overbrace{\overbrace{\Pi_{sid}(E)}^{\text{همه دانشجویان}} - \Pi_{sid}\Big(\underbrace{\Pi_{sid}(E) \times \Pi_{cid}(C)}_{\text{همه حالتهای ممکن}} - \underbrace{E}_{\text{شرکتهای واقعی}}\Big)}^{\text{دانشجویانی که در همه کلاسها شرکت کرده اند}}$$

4.9.4 تعریف عملیات تقسیم با عملگرهای اصلی

با الهام از مثالی که حل کردیم. فرمول تقسیم را در حالت کلی به شکل زیر خواهیم داشت.

$r \div s = \pi_{R-S}(r) - \pi_{R-S}((\pi_{R-S}(r) \times s) - \pi_{R-S,S}(r))$

تقسیم خصیصه‌های R از رابطه r بر خصیصه‌های S از رابطه s. دقت کنید که عملیات تقسیم فقط وقتی S زیر مجموعه R است تعریف شده.

نکته: عملگرهای اصلی مورد نیاز پرتو، ضرب و منها

نکته: به ترتیب منها، ضرب، منها توجه کنید.

تست فناوری اطلاعات ۱۳۸۷ ۴ تقسیم- اگر r, s دو رابطه باشند و R, S به ترتیب Heading این دو رابطه باشند و با فرض آنکه $S \subseteq R$ باشد، کدام‌یک از عبارت‌های زیر معادل با $r \div s$ است؟ (فناوری اطلاعات 1387)

1) $\Pi_{R-S}(r) - \Pi_{R-S}\big((\Pi_{R-S}(r) \times s)\big)$

2) $\Pi_{R-S}(r) \cap \Pi_{R-S}\big((\Pi_{R-S}(r) \times s)\big)$

3) $\Pi_{R-S}\big((\Pi_{R-S}(r) \times s) - \Pi_{R-S\cdot S}(r)\big)$

4) $\Pi_{R-S}(r) - \Pi_{R-S}\big((\Pi_{R-S}(r) \times s) - \Pi_{R-S\cdot S}(r)\big)$

پاسخ گزینه ۴ طبق تعریف تقسیم گزینه 4 صحیح است. به خاطر بیاورید که که در پیاده‌سازی تقسیم با عملگرهای اصلی به ترتیب از سمت چپ تفاضل، ضرب دکارتی، تفاضل داشتیم. تنها در گزینه ۴ این الگو دیده می‌شود.

سوال) باتوجه به جداول زیر مطلوب است دستور جبر رابطه‌ای برای تعیین شماره داوطلبانی S# که در همه‌ی آزمون‌ها شرکت کرده‌اند؟

داوطلب(S#, Sname, address)S
آزمون(T#, Tname, no-of-Ques)T
شرکت در آزمون(S#, T#, date, time, Code)ST

راه‌حل) اطلاعات شرکت در آزمون در جدول (رابطه) ST است. اطلاعات همه آزمون‌ها در جدول T است. کافی است که دو ستون S#و T# از جدول ST را بر ستون T# از جدول T تقسیم کنیم. ستون مشترک یعنی T# حذف می‌شود و در ستون باقیمانده -یعنی S#- شماره داوطلبانی که در همه آزمونها شرکت کرده‌اند (یا با تمام سطرهای جدول T رابطه داشته‌اند) را خواهیم داشت. $\pi_{S\#,T\#}(ST) \div \pi_{T\#}(T)$

توضیح بیشتر) به زبان ساده کوئری می‌گوید. S# هایی را بده که در همه آزمونها شرکت کرده اند. $\pi_{S\#,T\#}(ST) \div \pi_{T\#}(T)$

طرز نوشتن:

1. سمت راست همه را بنویسید را اول بنویسید.
2. تقسیم را بنویسید
3. سمت چپ همه را بنویسید.
4. خصیصه سمت چپ را با کاما به پرتو اول اضافه کنید. ***اخطار***: در برگه‌های امتحانی خیلی از دانشجویان این قسمت را فراموش می‌کنند.

تست مهندسی کامپیوتر ۱۳۸۸ ۱ - باتوجه به جداول زیر مطلوب است دستور جبر رابطه‌ای برای تعیین مشخصات کامل داوطلبانی که در همه‌ی آزمون‌ها شرکت کرده‌اند؟

داوطلب(S#, Sname, address)S
آزمون(T#, Tname, no-of-Ques)T
شرکت در آزمون(S#, T#, date, time, Code)ST

(مهندسی کامپیوتر 1388)

1) $\left(\pi_{S\#,T\#}(ST) \div \pi_{T\#}(T)\right) \bowtie S$

2) $\left(\pi_{S\#,T\#}(ST) \div \pi_{T\#}(T)\right) \propto S$

3) $S \bowtie \left(\pi_{S\#,T\#}(ST) \div \pi_{T\#,Tname}(T)\right)$

4) $S\left(\pi_{S\#,T\#}(ST) \div \pi_{T\#,Tname}(T)\right)$

پاسخ گزینه ۱ راه حل تستی: در تعریف تقسیم $s \div r$ همه مؤلفه‌های سمت راست s باید در سمت چپ r باشد. در گزینه 3 و 4 Tname در عملوند راست وجود دارد ولی در مؤلفه‌های عملوند چپ نیست. پس گزینه 3 و 4 قابل محاسبه نیست. گزینه ۳ و ۴ رد می‌شود.

گزینه ۱ و ۲ مثل هم هستند تنها در عملگر الحاق طبیعی $\bowtie$ و شبه الحاق $\propto$ با هم تفاوت دارند. در شبه‌الحاق ستون‌های سمت راست در خروجی ظاهر نمی‌شود. پس گزینه 2 غلط است زیرا ما مشخصات کامل دانشجویان را می‌خواهیم. یعنی تمام ستون‌های s باید در خروجی باشد. گزینه ۲ رد می‌شود. پاسخ گزینه 1 است.

نکته: اگر تعداد سطرهای R1 عدد m و سطرهای R2 عدد n باشد. در آنصورت مینیم و ماکزیمم سطرهای نتیجه برای عملگرهای مختلف در جدول زیر نمایش داده شده است.

	ماکزیمم سطرها	مینیمم سطرها	عملگر
	m× n	m× n	R1× R2
	m+n	max(m,n)	R1 ∪ R2
	min(m,n)	0	R1 ∩ R2
	m	0	R1-R2
	m×n	0	R1⋈ R2
	$\left\lfloor \frac{m}{n} \right\rfloor$	0	R1÷R2
	m	0	σ_p(R1)
	m	1	$\pi_{a1,a2,..}$(R1)

5 Min یا Max

در جبر رابطه‌ای عملگر Min یا Max نداریم پیدا کردن Min یا Max با مجموعه‌ای از عملیات‌های مقایسه تفاضل نامگذاری مجدد و ضرب انجام می‌شود.

سوال) نمرات دانشجویان در جدول T(name, GPA) ذخیره شده است می‌خواهیم نام دانشجو یا دانشجویانی که بیشترین معدل را دارند پیدا کنیم؟

پاسخ) در گام اول باید معدل دانشجوها در جدول T را با معدل دانشجویان در همین جدول T مقایسه کنیم. چون هر دو جدول هم نام هستند یکی از جدول‌ها را با نام NewT نامگذاری می کنیم. این دو جدول را در هم ضرب می‌کنیم تا بتوانیم کار مقایسه را انجام دهیم.
در گام دوم تاپل‌ها یا دانشجویانی انتخاب می‌شوند که معدل آنها از حداقل یکی از دانشجویان دیگر کمتر باشد. تنها معدلی که از هیچ کدام از معدل‌ها کمتر نیست معدل Max است. به عبارت دیگر خروجی این مقایسه، همه معدل‌ها را می‌دهد به غیر از معدل Max.
در پایان برای به دست آوردن معدل Max کافیست. از مجموعه همه معدل‌ها مجموعه مرحله قبل را کم کنیم.

T		$T \times \rho_{NewT}(T)$				$A=\sigma_{T.GPA<NewT.GPA}(T \times \rho_{NewT}(T))$				$T-\pi_{T.Name,T.GPA}(A)$	
name	GPA	name	GPA	name	GPA	Name	GPA	name	GPA	name	GPA
Reza	17	Reza	17	Reza	17	Sara	14	Reza	17	Ali	19
Ali	19	Ali	19	Reza	17	Reza	17	Ali	19		
Sara	14	Sara	14	Reza	17	Sara	14	Ali	19		
		Reza	17	Ali	19						
		Ali	19	Ali	19						
		Sara	14	Ali	19						
		Reza	17	Sara	14						
		Ali	19	Sara	14						
		Sara	14	Sara	14						

مراحل بالا را می‌توان به شکل زیر هم نمایش داد.

(همه معدلهای به غیر از بیشترین معدل)-همه معدل‌ها = بیشترین معدل

سوال) مطلوبست نام دانشجویی که کمترین معدل min را دارد؟

پاسخ)$T-\pi_{T.Name,T.GPA}(\sigma_{T.GPA>NewT.GPA}(T \times \rho_{NewT}(T)))$

5.1.1.1.1.1 تست سراسری فناوری اطلاعات ۱۳۸۴

6) رابطهٔ حرارت (کد ناحیه، نام، تاریخ، حرارت بالا، حرارت پایین) برای ثبت درجه حرارت‌های بیشینه و کمینه در نواحی مختلف در زمان‌های متفاوت به‌کار می‌رود.
کلید این رابطه < کد ناحیه، تاریخ > می‌باشد. اگر روابط جبر رابطه‌ای زیر را درنظر بگیرید:

$R1\ (\text{کد، تاریخ، H}) = \Pi_{\text{کد ناحیه، تاریخ، حرارت بالا}}(\text{حرارت})$

$R2\ (\text{کد، تاریخ، L}) = \Pi_{\text{کد ناحیه، تاریخ، حرارت پایین}}(\text{حرارت})$

$R3\ (\text{کد ناحیه}) = \Pi_{\text{کد}}(R1 \overset{\bowtie}{}_{H < \text{حرارت بالا}} \text{حرارت})$

$R4\ (\text{کد ناحیه}) = \Pi_{\text{کد}}(R2 \overset{\bowtie}{}_{L > \text{حرارت بالا}} \text{حرارت})$

آنگاه کدام گزینه نام نواحی با درجه حرارت بیشینه و کمینه را به ما می‌دهد؟

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

1) $\Pi_{\text{نام}}(\text{حرارت} \bowtie (R3 \bowtie R4))$

2) $\Pi_{\text{نام}}(\text{حرارت} \bowtie (R3 \cup R4))$

3) $\Pi_{\text{نام}}(\text{حرارت} \bowtie ((\Pi_{\text{کد ناحیه}}(\text{حرارت}) - R3) \cup (\Pi_{\text{کد ناحیه}}(\text{حرارت}) - R4))$

4) $\Pi_{\text{نام}}(\text{حرارت} \bowtie ((\Pi_{\text{کد ناحیه}}(\text{حرارت}) \cap R3) - (\Pi_{\text{کد ناحیه}}(\text{حرارت}) \cap R4))$

پاسخ گزینه ۳ راه حل تستی) در این تست برای نامگذاری مجدد از یک نوتیشن خاص استفاده کرده است. مثلا نامگذاری R2 و R1 را می‌توان به شکل زیر نوشت.

$\rho_{R1(H,\text{تاریخ},\text{کد})}(\pi_{\text{کدناحیه، تاریخ، حرارت بالا}}(\text{حرارت}))$

$\rho_{R1(L,\text{تاریخ},\text{کد})}(\pi_{\text{کدناحیه،تاریخ، حرارت پایین}}(\text{حرارت}))$

علاوه بر این به جای ضرب دکارتی و سیگما از تتا جوین استفاده کرده است. می‌توان R3 و R4 را به شکل زیر بازنویسی کرد.

R3(کد ناحیه) = $\pi_{\text{کد}}(\sigma_{H<\text{حرارت بالا}}(\text{حرارت} \times R1))$ همه کد ناحیه‌ها به غیر از ناحیه با دمای بیشینه=

عبارت بالا را می‌توانید این گونه تفسیر کنید. تنها ناحیه‌ای که دمایش H از حرارت بالا هیچ ناحیه‌ای کمتر نیست. ناحیه با دمای بیشینه است.

R4(کد ناحیه) = $\pi_{\text{کد}}(\sigma_{L>\text{حرارت پایین}}(\text{حرارت} \times R۲))$ همه کد ناحیه‌ها به غیر از ناحیه با دمای کمینه=

عبارت بالا را می‌توانید این گونه تفسیر کنید. تنها ناحیه‌ای که دمایش L از حرارت پایین هیچ ناحیه‌ای بیشتر نیست. ناحیه با دمای کمینه است.

مطابق عملیات بیشینه کمینه که در جزوه گفته شد. R3 و R4 همه کد ناحیه‌ها را می‌دهند به جز ناحیه با دمای بیشینه و کمینه. برای به دست آوردن بیشینه و کمینه علاوه بر ضرب، تغییرنام، پرتو π و انتخاب σ ما به تفاضل هم نیاز داریم. R3 و R4 باید بعد از تفاضل قرار بگیرید. تنها گزینه‌ای که این مورد را رعایت کرده است. گزینه ۳ است. پایان راه حل تستی.

راه حل تشریحی) گزینه۳ را به شکل زیر تفسیر می‌کنیم. مثل همیشه از پرانتزهای داخلی باید به سمت بیرون حرکت کنیم.

$$\left(\pi_{\text{نام}}\left(\overbrace{\text{حرارت} \bowtie \left(\underbrace{\left(\pi_{\text{کدناحیه}}(\text{حرارت}) - R3\right)}_{\text{کد ناحیه با دمای بیشینه}} \cup \underbrace{\left(\pi_{\text{کدناحیه}}(\text{حرارت}) - R4\right)}_{\text{کد ناحیه با دمای کمینه}}\right)}^{\text{الف}}\right)\right)$$

قسمت الف با الحاق طبیعی جدول حرارت با دو کدناحیه اطلاعات کامل این دو کد ناحیه را به ما می دهد. یعنی اطلاعات کامل (شامل تمام ستونها) ناحیه با دمای کمینه و بیشینه.
در پایان هم عملگر پرتو π تنها نام این دو ناحیه را در خروجی نمایش می‌دهد.

SQL

SQL = Structured Query Language یک زبان برای نوشتن پرس‌وجو در پایگاه‌داده است.

اکثر زبانهایی برنامه‌نویسی مثل پایتون، سی، جاوا، .. رویه‌ای هستند، در حالیکه SQL زبان توصیفی است؟

سوال) تفاوت زبان توصیفی و رویه‌ای در چیست؟

زبان SQL بر اساس جبر رابطه‌ای طراحی شده‌است، توصیف (اجرا) از داخلی‌ترین پرانتز شروع می‌شود به سمت بیرونی‌ترین پرانتز

$$\Pi_{sid}(E) - \Pi_{sid}(\Pi_{sid}(E) \times \Pi_{cid}(C) - E)$$

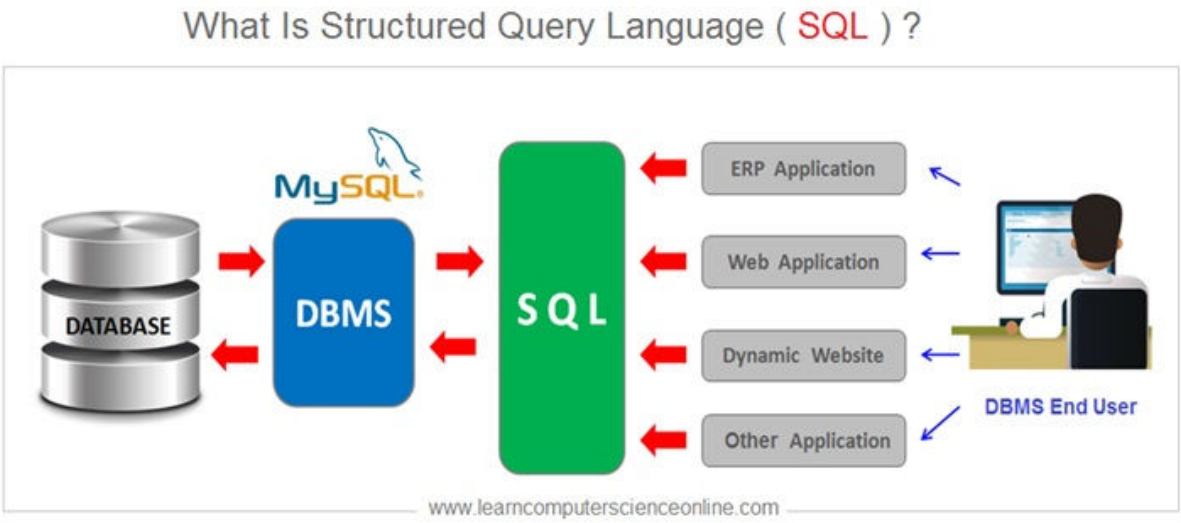
زبان رویه‌ای (مثلا پایتون) بر اساس الگوریتم طراحی شده‌است. اجرا از خط اول شروع می‌شود به سمت پایین.

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

sum = num1 + num2
print("Sum:", sum)
```

نکته: زبان SQL بر اساس جبر رابطه‌ای پیاده‌سازی‌شده ولی بعضی از امکانات آن را ندارد و برخی از قواعد آن را نقض کرده است.

سوال) آیا زبان SQL یک DBMS است؟



برخی از DBMS هایی که از SQL پشتیبانی می‌کنند؟

دسته‌بندی دستورات SQL

این دسته‌بندی در منابع مختلف متفاوت است.

DDL

Data Definition Language

این دستورات در ساختار پایگاه‌داده اثر گذراند و محتوی پایگاه‌داده را تغییر نمی‌دهند.

مثلا Create Table برای ایجاد جدول، Drop Table برای حذف جدول، دستور Primary key برای تعیین کلید اصلی جدول..

DML

Data Manipulation Language

این دستورات برای دستکاری و تغییر محتوی پایگاه‌داده کاربرد دارند.

مثلا) insert into برای اضافه‌کردن سطر به جدول، delete from برای حذف سطر از جدول،

DCL

Data Control Language

این دستورات برای تعریف محدود

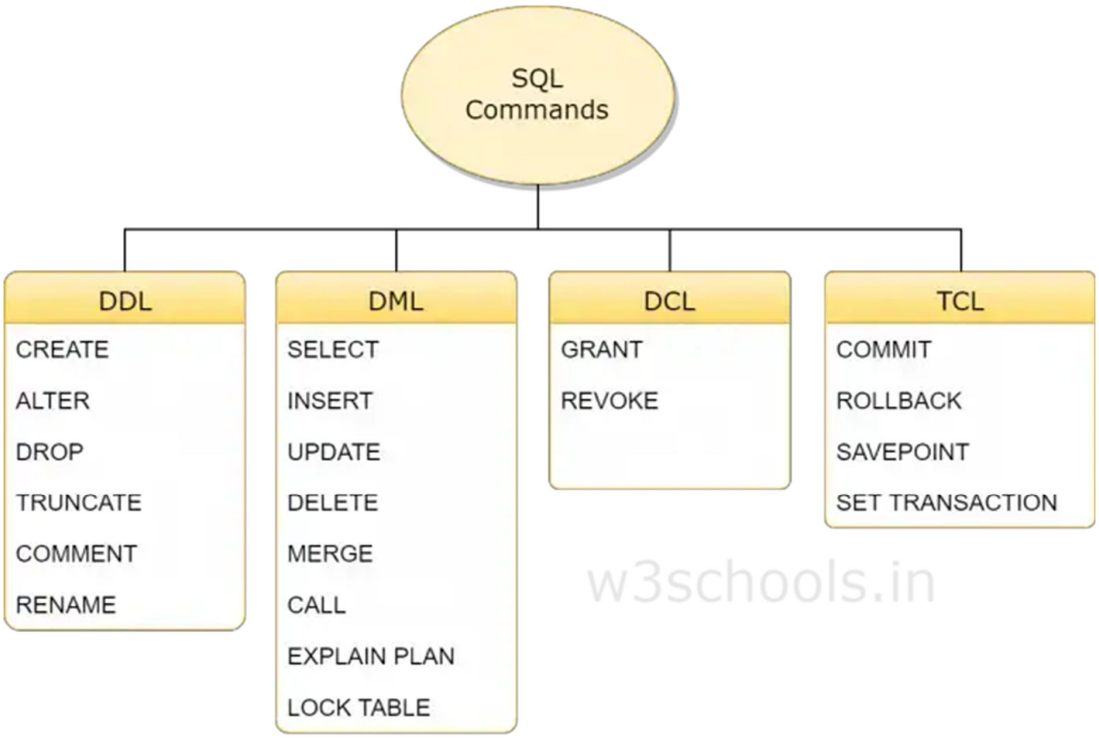
مثلا) grant برای اعطای مجوز دست

TCL

Transaction Control Language

این دستورات برای کنترل اجرای تراکنش‌ها استفاده می‌شود.

مثلا) rollback برای عقب‌گرد از یک تراکنش به هنگام بروز خطا



انواع داده‌ها در زبان SQL

وقتی یک خصیصه (ستون) جدول تعریف می‌شود باید نوع آن هم مشخص شود. بر اساس نیاز تحلیلگر خصیصه (داده) می‌تواند انواع مختلفی داشته باشد.

بعضی از DBMS ها ممکن است برخی از انواع داده را پیاده‌سازی نکرده باشند. یا پیاده‌سازی آنها با هم متفاوت باشند. در اینجا آنچه عمومیت دارد، گفته شده است. توصیه می‌شود درباره هر DBMS به مستندات مربوطه مراجعه کنید.

INTEGER

مقدار عدد صحیح معمولا از **-2,147,483,648** تا **2,147,483,647**

SMALLINT

مقدار عدد صحیح معمولا از **-32,768** تا **32,767**

NUMERIC(p,s)

مجموعه اعداد حقیقی که تعداد رقم‌های آن توسط کاربر تعریف می‌شود.

The " `balance` " column can safely store the number 173226.62.

DATE

Microsoft SQL Server: Can only store date from **January 1, 0001** to **December 31, 9999**

MySQL : can store date in format of **YYYY-MM-DD** ranging from **1000-01-01** to **9999-12-31**

سوال) چرا باید از DATE استفاده کنیم؟

نکته) با استفاهد از DATE همیشه داده معتبر ذخیره می‌شود. مثلا اگر ماه را INTEGER ذخیره کنیم. باید حواسمان باشد که کسی عدد خارج از بازه 1 تا 12 وارد نکند. علاوه بر این توابعی مثل DATEPART()، DATEADD()، DATEDIFF برای این نوع تعریف شده که کار پرس‌وجو نوشتن را بسیار ساده می‌کند.

DATETIME

can store date and time in format of **YYYY-MM-DD hh:mm:ss**

TIME

فقط زمان را نگه می‌دارد. **hh:mm:ss**

Char(n)

رشته کاراکتری با طول ثابت و مشخص

Varchar(n)

رشته کاراکتری با طول متغیر و حداکثر n کاراکتر

Boolean

نوع بولین می‌تواند یک از دو مقدار درست یا غلط را قبول کند.

نحوه نمایش درست: TRUE, YES, ON, 1

نحوه نمایش درست: FALSE, NO, OFF, 0

Blob

نوع داده blob برای ذخیره‌سازی داده‌های حجیم ناشناخته مثل تصویر، ویدیو، فایل.. استفاده می‌شود. مثلا اگر قرار باشد تصویر کارت ملی دانشجوها را به جدول اضافه کنیم باید یک ستون از این نوع Blob به جدول اضافه کنیم.

سوال) اگر نوع داده‌ای نیاز داشته‌باشیم که در DBMS تعریف نشده باشد، چه کار باید کرد؟

نکته) در SQL امکان تعریف نوع داده (type) جدید وجود دارد. برای این کار می‌توان از دو دستور Create type و create domain استفاده کرد.

مثال)

```
create domain NationonNumber as numeric(10,0)
```

عددی که شامل ده رقم است و اعشار هم ندارد

نکته) Domain قوی‌تر است. در زمان تعریف می‌توان محدودیتهایی مثل not null یا چک را به آن اضافه کرد.

```
CREATE DOMAIN contact_name AS
   VARCHAR NOT NULL CHECK (value !~ '\s');
```

رشته کاراکتری که نباید null باشد یا شامل کاراکتر blank

یادآوری پایگاه داده course

در جلسات گذشته از یک پایگاه‌داده به عنوان مثال استفاده کردیم، به نام course . این پایگاه داده چهار جدول دارد که عبارت است از.

Student (sname, city, univ)

Course (cname, tname)

Buy (sname, cname, tr_no, amount)

Att (Sname, cname, meeting, hour, att_no)

دستورات DDL

Create Database

```
create database  courses
```

دستور Create Table

با اجرای این دستور جدولهای پایگاه‌داده ایجاد می‌شود. مثلا برای ایجاد جدول student می‌نویسیم.

```
CREATE TABLE student(
sname char(20),
cname char(20),
uni char(20)
);
```

سوال) دستور زیر با دستور بالا چه تفاوتی دارد؟

```
CREATE TABLE student(sname char(20), cname char(20), uni char(20) );
```

نکته) نوشتن دستورات SQL در یک خط یا چند خط تفاوتی در اجرای آنها ندارد.

سوال) چگونه کلید اصلی جدول یعنی student را مشخص کنیم.

```
CREATE TABLE student(
sname char(20),
cname char(20),
uni char(20)
PRIMARY KEY (sname)
);
```

تعیین کلید اصلی یک سری محدودیت جامعیتی برای جدول تعریف می‌کند. مثلا در جدول بالا sname نمی‌تواند مقدار تکراری بگیرد و نمی‌تواند null باشد.

نکته) به هنگام ایجاد جدول باید تمامی محدودیتهای جامعیتی آن را نیز مشخص کرد. بهترین مثال تعریف کلیدهای اصلی و خارجی یک جدول به هنگام ایجاد است.

در ادامه جدول حضور یا att را ایجاد می‌کنیم.

```
CREATE TABLE att (
sname char(30),
cname char(20),
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no));
```

دقت کنید که در اینجا ستون att_no را به عنوان کلید اصلی تعیین کردیم.

در گام بعدی کلید خارجی ها را تعریف می کنیم. Sname این جدول را به student و cname این جدول را به course ارتباط می‌دهد.

```
CREATE TABLE att (
sname char(30),
cname char(20),
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student)
```

در کد بالا خطاهای زیر وجود دارد.

1. در کد بالا نوع sname در att با نوع sname در جدول student یکسان نیست. این باعث خطای referential error خواهد شد. یعنی وقتی این دستور را اجرا می کنید خوشبختانه DMBMS به شما خطا می‌دهد.
2. جدول مرجع cname یعنی course ایجاد نشده است.

```
CREATE TABLE course(
cname char(20),
tname char(20))

CREATE TABLE att (
sname char(20),
cname char(20),
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student))
```

کد بالا همچنان مشکل دارد. کلید مرجع حتما باید کلید اصلی باشد. یعنی باید ایجاد course را به شکل زیر اصلاح کنیم.

```
CREATE TABLE course(
cname char(20),
tname char(20),
PRIMARY KEY (cname))
```

کلید خارجی در sql می‌تواند null باشد و قوانین سه گانه جامعیت را نقض نمی‌کند. مثلا کد زیر صحیح است. ولی در کابرد ما بی معناست. زیرا (null, DB, 10, 2, 5544) به عنوان سطر جدول att یعنی دانشجویی که معلوم نیست در جلسه 10 پایگاه داده به مدت 2 ساعت با کد حضور 5544 حضور داشته است. برای جلوگیری از این حالتها باید صراحتا not null بودن را به هنگام ایجاد قید کنیم.

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student))
```

به نظر شما کد زیر صحیح است؟

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES student,
FOREIGN KEY (sname) REFERENCES) course))
```

این کد یک خطای معنایی دارد که متاسفانه DBMS متوجه آن نشده و در حین کار دردسرهای بزرگی برای استفاده کنندگان پایگاه‌داده ایجاد می‌کند.

نکات مهم به هنگام تعریف کلید خارجی:

1. کلید خارجی باید با کلید اصلی مرجع هم نوع باشد
2. کلید خارجی بهتر است با کلید اصلی مرجع هم نام باشد.
3. کلید خارجی باید با کلید اصلی مرجع هم معنی باشد.
4. کلید خارجی باید به کلید کاندید (در عمل کلید اصلی) ارجاع دهد.
5. اگر کلید خارجی not null باشد باید صراحتاً ذکر شود.

مثال مهر (بچسب) زمانی timestamp

یک جدول دو ستونه ایجاد می‌کنیم.

```
CREATE TABLE TestTable2 (ID INT,
MyTime TIME DEFAULT CURRENT_TIMESTAMP,
MyDate DATE DEFAULT CURRENT_TIMESTAMP)
```

حالا یک سطر به این جدول اضافه می‌کنیم.

```
INSERT INTO TestTable2 (ID)   VALUES (4)
```

و بعد از یک دقیقه یک سطر دیگر

```
INSERT INTO TestTable2 (ID)   VALUES (7)
```

پس از اضافه کردن رکورد مقدار آن را نمایش می‌دهیم.

```
SELECT * FROM TestTable2
```

	id integer	mytime time without time zone	mydate date
1	4	11:46:51.297225	2023-09-20
2	7	11:47:04.325151	2023-09-20

درلحظه ثبت رکورد 1 با مقدار ID=4 زمان ثبت به شکل برچسب زمانی ذخیره شده است.

سوال) کاربرد برچسب زمانی چیست؟

نکته) برچسب زمانی اجازه ردیابی تغییرات و دسترسی‌ها را به مدیر پایگاه داده و برنامه نویسان می‌دهد.
مثلا سوالات زیر به راحتی پاسخ داده می شود.

- رکوردهایی که در ده روز گذشته به پایگاه داده اضافه شده چه بوده
- چه کسی آخرین بار به اطلاعات پایگاه داده دسترسی داشته است.

دستور drop table

از این دستور برای حذف یک جدول از پایگاه‌داده استفاده می‌شود.

نکاتی که باید در دستور drop رعایت شود.

1. جدول باید خالی باشد.
2. جدول مورد ارجاع نباشد. یعنی کلید اصلی جدول در جدول دیگر کلید خارجی نباشد.

مثلا

```
CREATE TABLE course(
cname char(20),
tname char(20),
PRIMARY KEY (cname))

CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student))
```

فرض کنید جدولهای زیر را ایجاد کرده‌ایم و هنوز خالی هستند.

سوال) آیا دستور زیر صحیح است؟

```
DROP TABLE course
```

پاسخ) خیر کلید اصلی این جدول یعنی cname در جدول att به عنوان کلید خارجی تعریف شده است. اجرا این دستور referential integrity error می‌دهد.

سوال) در صورت حذف course کدام قانون جامعیت نقض می‌شود؟

پاسخ) قانون سوم از منظر ساختاری

سوال) آیا دستور زیر صحیح است؟

```
DROP TABLE att
```

پاسخ) بلی. البته به شرط آنکه جدول att خالی باشد، هیچ سطری نداشته باشد.

دستور Alter table

از این دستور برای تغییر در جدول ایجاد شده می‌توان استفاده کرد. 1. اضافه کردن ستون 2. تغییر نوع ستون 3. حذف ستون

سوال) این دستور چه کار می‌کند؟

```
ALTER TABLE att ADD attdate DATETIME
```

پاسخ) به جدول att یک ستون جدید به نام attdate از نوع DATETIME اضافه می‌کند.

Sname	cname	Meeting	hour	attno	Attdate

سوال) اگر جدول att چند سطر داشته باشد و ستون جدید اضافه کنیم، مقدار این ستون در سطرهای قبلی چه می‌شود؟

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760

پاسخ)

این ستون در سطرهای قبلی با مقدار null مقدار دهی می‌شوند.

Sname	cname	meeting	hour	attno	Attdate
Ali	DB	2	2.5	1576	Null
Reza	AI	1	1.5	1760	Null

سوال) بعد از مدتی متوجه می‌شویم که 20 کاراکتر برای sname کافی نیست. چگونه جدول را تغییر بدهیم.

پاسخ)
```
ALTER TABLE att MODIFY sname CHAR(30)
```

سواال) بعد از این که چند سطر وارد کردیم. به این نتیجه می‌رسیم که تنها 10 کاراکتر برای ستون cname کافی است، چه باید کرد؟

پاسخ) هیچ کار نمی‌توان کرد! در دستور ALTER نوع قبلی باید زیر مجموعه نوع جدید باشد. مگر اینکه این ستون جدول خالی باشد و هیچ مقداری نداشته باشد.

```
ALTER TABLE att MODIFY cname CHAR(10)
```

سوال) درباره دستورهای زیر چه می‌توان گفت.

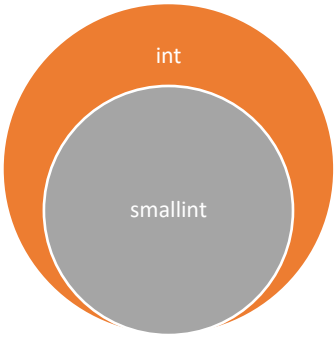
```
ALTER TABLE att MODIFY cname SMALLINT
ALTER TABLE att MODIFY cname CHAR(40)
ALTER TABLE att MODIFY attno SMALLINT
```

پاسخ)

دستور اول در صورتی درست است که ستون cname خالی باشد.

دستور دوم همیشه درست است زیر CHAR(20) زیر مجموعه CHAR(40) است.

دستور سوم در صورتی درست است که ستون attno خالی باشد.



نکته در زبان pgSQL، تغییر نوع ستون به شکل زیر نوشته می‌شود.

```
ALTER TABLE att ALTER COLUMN cname TYPE CHAR(40)
```

سوال) آیا می‌توان یک ستون از جدول را حذف کرد؟

پاسخ) در برخی از پیاده‌سازی های این امکان وجود دارد. مثلا دستور زیر meeting و مقادیر آن را از جدول att حذف می‌کند.

```
ALTER TABLE att DROP meeting
```

نکته این حذف در صورتی امکان‌پذیر است که ستون مورد نظر کلید اصلی نباشد.

دستورات DML

دستور Insert Into

برای اضافه کردن سطر به جدول از این دستور استفاده می‌کنیم.

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760

سوال) به جدول att یک سطر جدید اضافه کنید.

```
INSERT INTO att VALUES ('amin', 'AI', 3, 3.5, 1790)
```

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790

سوال) دستور زیر چه کار می‌کنند؟

```
INSERT INTO att VALUES ('sara', 'AI', , , 2040)
```

پاسخ)

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790
sara	AI	null	null	2040

نکته) در برخی از پیاده‌سازی‌ها باید به این شکل نوشت.

```
INSERT INTO att VALUES ('sara', 'AI', null, null, 2040)
```

سوال) دستور زیر چه کار می‌کند؟

```
INSERT INTO att VALUES ('sara', 'AI', null, null, null)
```

پاسخ) این دستور به خطا مواجه می‌شود، زیرا attno ستون اصلی است و نباید null باشد.

سوال) دستور زیر چه کار می‌کند؟

```
INSERT INTO att VALUES (, 'DB', 5, 2, 1890)
```

پاسخ) این دستور خطا دارد. زیرا ستونsname طبق تعریف نباید null باشد.

```
CREATE TABLE att (
sname char(20) NOT NULL, cname char(20) NOT NULL, meeting int,  hour float,
att_no int, PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,FOREIGN KEY (sname) REFERENCES student))
```

نکته) در دستور insert into ستون کلید اصلی و ستونهای not null نمی‌توانند null بگیرند.

سوال) معادل دستور insert into در جبر رابطه‌ای چیست؟

پاسخ)`Att ← Att U ('amin', 'AI', 3, 3.5, 1790)`

Insert into با اجتماع و انتساب پیاده‌سازی می‌شود.

سوال) خروجی این دستور چیست؟ `INSERT INTO att VALUES ('amin', 'AI', 2005)`

پاسخ) خطا. نمی‌توان در یک جدول با 5 ستون، سطر 3 ستونی وارد کرد! شرط نخست سازگاری عمل اجتماع را نقض کرده‌است.

سوال) خروجی این دستور چیست؟

```
INSERT INTO att VALUES ('amin', 'AI',2.5, 3.5, 2005)
```

پاسخ) خطا. در ستون meeting فقط باید مقادیر int وارد کرد. شرط دوم سازگاری را نقض کرده است.

نکته) دستور insert into را می‌توان به جای

```
INSERT INTO att VALUES ('amin', 'AI', 3, 3.5, 1760)
```

به شکل زیر هم نوشت.

```
INSERT INTO att (sname, cname, meeting, hour, attno) VALUES ('amin', 'AI', 3, 3.5,
1760)
```

در این شکل ستونهایی که باید مقدار بگیرند، صراحتاً مشخص شده و بقیه ستونها null می‌شوند.

سوال) خروجی این دستور چیست؟

```
INSERT INTO att (sname, cname, attno) VALUES ('amin', 'AI', 2005)
```

پاسخ) سطری با مقدار amin, ai, null, null, 2005 وارد می‌شود.

نکته) در دستور insert into کلید خارجی باید در جدول مرجع وجود داشته باشد.

به عنوان نمونه اگر در جدول course درس AI و در جدول student، sara نداشته باشیم دستور زیر خطا تولید می‌کند.

```
INSERT INTO att VALUES ('sara', 'AI', null, null, 2040)
```

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student))
```

سوال) نتیجه اجرای دستور زیر در جدول چه خواهد بود؟

```
INSERT INTO att VALUES ('sara', 'AI', 3, 2.5, 1790)
```

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790

پاسخ) این دستور با خطا روبرو می‌شود چون attno کلید اصلی است و کلید اصلی نمی‌تواند تکراری باشد.

دستور Delete

با این دستور می‌توان یک سطر یا چند سطر از پایگاه داده حذف کرد.

مثال) حضور امین را می‌خواهیم از جدول حذف کنیم.

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790
sara	AI	null	null	2040

```
DELETE from att WHERE sname = 'amin'
```

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
~~Amin~~	~~AI~~	~~3~~	~~3.5~~	~~1790~~
Sara	AI	null	null	2040

سوال) با جبر رابطه‌ای امین را از جدول حذف کنید.

پاسخ) $Att \leftarrow att - \sigma_{sname = 'amin'}(att)$

مثال) می خواهیم همه سطرهای جدول را حذف کنیم.

```
DELETE from att
```

نکته) با این دستور می‌توان تمام سطرهای یک جدول را حذف کرد ولی خود جدول را نمی‌تواند از ساختار پایگاه داده حذف کند!

نکته) در عمل برای جلوگیری از حذف سهوی داده شرط حذف بر اساس کلید اصلی نوشته می‌شود.

سوال) از جدول student علی را حذف کنید.

sname	city	Univ
ali	qom	Sharif
reza	sari	Kerman
sara	bam	Tabriz

پاسخ)
```
DELETE from student WHERE sname = 'ali'
```

Sname	cname	meeting	Hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760

sara	AI	null	Null	2040

اگر با دستور بالا ali را حذف کنیم. ارجاع ali در جدول att ، قانون سوم جامعیت را نقض می‌کند. DBMS خوشبختانه مانع نقض قانون جامعیت شده و این دستور را اجرا نمی‌کند. خطا میدهد. برای حل این مشکل دو راه‌حل داریم.

1. قبل از حذف ali از جدول student تمامی ارجاعات را حذف کنیم.

```
DELETE from att WHERE sname = 'ali'
```

البته ممکن است جدول buy هم ارجاع به ali ارجاع داده باشد.

2. در تعریف کلید خارجی از on delete cascade استفاده کنیم.

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course ,
FOREIGN KEY (sname) REFERENCES student))
```

با اضافه کردن on delete cascade به تعریف کلید خارجی با حذف ali از جدول student هیچ مشکلی پیش نمی‌آید. علاوه بر آن حضورهای علی در جدول att هم به شکل خودکار از این جدول حذف می شود.

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course ON DELETE CASCADE ,
FOREIGN KEY (sname) REFERENCES student ON DELETE CASCADE))
```

اینبار با اجرای `DELETE from student WHERE sname = 'ali'` جدول ها به شکل زیر در می آیند.

sname	city	Univ
~~Ali~~	~~qom~~	~~Sharif~~
reza	sari	Kerman
sara	bam	Tabriz

Sname	cname	meeting	Hour	Attno
~~Ali~~	~~DB~~	~~2~~	~~2.5~~	~~1576~~
Reza	AI	1	1.5	1760
sara	AI	Null	Null	2040

نکته) برای حذف یک سطر باید کلید اصلی آن مرجع هیچ کلید خارجی نباشد.

نکته) از on delete cascade به شکل زنجیره‌ای هم می‌توان استفاده کرد.

دستور update

با استفاده از این دستور می‌توان تغییرات دلخواه را روی محتوی جدول انجام داد. مثلا فرض کنید. شهر محل سکونت یک دانشجو عوض شود.

مثال) شهر محل سکونت ali به تهران عوض شده‌است.

```
UPDATE student SET city = 'tehran' WHERE sname = 'ali'
```

sname	city	Univ
Ali	~~Qom~~ Tehran	Sharif
reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	kish	Babol

مثال) حضورهای att بیشتر از 2 ساعت را 50 درصد زیاد کنید.

```
UPDATE att SET hour = hour * 1.5 WHERE hour > 2
```

Sname	cname	meeting	hour	Attno
Ali	DB	2	~~2.5~~3.75	1576
Reza	AI	1	1.5	1760
Amin	AI	3	~~3.5~~5.25	1790
Sara	AI	null	null	2040

مثال) حضورهای att بیشتر از 2 ساعت را 50 درصد زیاد کنید. و کمتر از 2 ساعت را 20 درصد

```
UPDATE att SET hour = CASE
                        WHEN hour > 2 THEN hour * 1.5
                        ELSE hour * 1.2
                                END
```

مثال) دستوری بنویسید که اسم amin را به ala تغییر دهد.

sname	city	Univ
Ali	Tehran	Sharif
reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	Kish	Babol

```
UPDATE student SET sname = 'ala' WHERE sname = 'amin'
```

این دستور تولید خطا می‌کند. چرا؟

پاسخ) amin در جدول att هست و به amin در جدول student ارجاع می‌دهد. تغییر نام باعث نقض قانون سوم جامعیت از نظر محتوایی می شود!

Sname	cname	meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790
Sara	AI	null	null	2040

راه حل استفاده on update cascade به هنگام ایجاد جدول و تعریف کلید خارجی می‌باشد.

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meeting int,
hour float,
att_no int,
PRIMARY KEY (att_no),
FOREIGN KEY (cname) REFERENCES course ON UPDATE CASCADE ,
FOREIGN KEY (sname) REFERENCES student ON UPDATE CASCADE))
```

اگه به هنگام تعریف کلید خارجی تعریف مانند بالا باشد، خطا نخواهیم داشت.

```
UPDATE student SET sname = 'ala' WHERE sname = 'amin'
```

بعد از اجرای دستور بالا هم در student و هم در att، امین به علا تغییر می‌کند.

Student		
sname	city	Univ
Ali	Tehran	Sharif
Reza	Sari	Kerman

Sara	Bam	Tabriz
~~Amin~~ ala	Kish	Babol

Att				
Sname	***cname***	meeting	Hour	**Attno**
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
~~Amin~~ ala	AI	3	3.5	1790
Sara	AI	null	null	2040

سوال) خروجی پرس‌وجوی زیر چیست؟

```
UPDATE student SET sname = 'reza' WHERE sname = 'amin'
```

Student		
sname	city	Univ
Ali	Tehran	Sharif
Reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	Kish	Babol

پاسخ) این دستور خطای duplicate می‌دهد زیرا sname کلید اصلی است و نمی‌تواند تکراری باشد.

سوال) معادل جبر رابطه‌ای update چیه؟

پاسخ) در update ما یک تعداد سطر را حذف delete کرده و دوباره با مقادیر بروزرسانی وارد insert کرده‌ایم. در جبر رابطه‌ای معادل update را نداریم.

دستور SELECT

دستور select مهمترین و پرکاربردترین دستور SQL برای گزارش‌گیری از جدول‌های پایگاه داده کاربرد دارد. امکانات دستور select را با مثال و پرس‌وجوهای متنوعی در هفته اول درس توضیح دادیم. علاوه بر آن بعداً به هنگام پیاده‌سازی آن پرس‌وجوها با عملگرهای جبر رابطه‌ای دوباره آنها را یادآوری کردیم.

ساختار کلی دستور Select

ستون‌ها Select

جدول‌ها Form

شرط‌ها Where

دستور نامگذاری مجدد AS

در SQL با دستور AS می‌توان عملگر rename ρ جبر رابطه‌ای را شبیه‌سازی کنیم.

مثال) نام و شهر دانشجویانی که در کلاس DB حضور داشته‌اند.

student		
sname	City	Univ
Ali	Tehran	Sharif
Reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	Kish	Babol

Att				
Sname	*cname*	meeting	Hour	**Attno**
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
~~Amin~~ ala	AI	3	3.5	1790
Sara	AI	null	null	2040

اطلاعات شهر دانشجویان در جدول student و اطلاعات حضور در جدول att است. پس ما باید این دو جدول را در پرس‌وجو با هم ترکیب(الحاق)(ضرب) کنیم.

```
SELECT student.sname, student.city
FROM student, att
WHERE student.sname = att.sname
```

دقت کنید که شرط student.sname = att.sname فقط سطرهای با معنی (معتبر) را در نگه می‌دارد. حالا یک شرط جدید اضافه می‌کنیم که حضور در کلاس DB باقی بماند.

```
SELECT student.sname, student.city
FROM student, att
WHERE student.sname = att.sname AND att.cname = 'DB'
```

حالا با استفاده از دستور AS می‌خواهیم پرس‌وجو را کمی ساده‌تر کنیم.

```
SELECT S.sname, S.city
FROM student AS S, att AS A
WHERE S.sname = A.sname AND A.cname = 'DB'
```

در این مثال AS کار پرس‌وجو نویسی را ساده‌تر کرد. ولی بعضی مواقع بدون AS نمی‌توان پرس‌وجو را نوشت.

مثال) نام دانشجویانی که با reza در یک شهر زندگی می‌کنند و با رضا هم دانشگاهی نیز هستند.

Student		
sname	city	Univ

Ali	Tehran	Sharif
Reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	Sari	Kerman

در گام اول باید شهر و دانشگاه رضا را پیدا کنیم.

```
SELECT city, uni
FROM student
WHERE sname = 'Reza'
```

پرس‌وجوی بالا جدول زیر را تولید می‌کند. به این جدول اصطلاحاً جدول میانی، مشتق (intermediate) گفته‌می‌شود. اسم مستعار B را به آن می‌دهیم. این جدول میانی سطری (بخشی) از جدول student است.

B	
city	Univ
Sari	Kerman

این جدول را با جدول student الحاق (ضرب) می‌کنیم.

B x Student				
city	Univ	sname	city	Univ
sari	Kerman	Ali	Tehran	Sharif
Sari	Kerman	Reza	Sari	Kerman
Sari	Kerman	Sara	Bam	Tabriz
Sari	Kerman	Amin	Sari	Kerman

در جدول حاصلضرب بالا سطرهایی که شرط B.city = student.city AND B.uni = student.uni دارند جواب هستند.

```
SELECT B.city
from (
      SELECT city, uni
      FROM student
      WHERE sname = 'Reza') AS B, student
WHERE B.city = student.city AND B.uni = student.uni
```

نوشتن پرس‌وجو اینجا به پایان می‌رسد. ولی می‌توان با یک نامگذاری دیگر آن را ساده‌تر نوشت.

```
SELECT B.city
from (
      SELECT city, uni
      FROM student
      WHERE sname = 'Reza') AS B, student  AS S
WHERE B.city = S.city AND B.uni = S.uni
```

نکته در برخی از پیاده‌سازی (و بعضی از سوالات کنکور) از AS استفاده نشده و به شکل زیر نوشته‌می‌شود.

```
SELECT B.city
from (
      SELECT city, uni
```

```
      FROM student
      WHERE sname = 'Reza') B, student  S
WHERE B.city = S.city AND B.uni = S.uni
```

تعریف جدول پایه base) جدولی که توسط طراح پایگاه‌داده با استفاده از create table به طور دایمی ایجادمی‌شود.

مثال) جدول student، course ، att

تعریف جدول میانی (مشتق) intermediate) DBMS نتیجه‌ی هر پرس‌وجو SQL به طور موقت در یک جدول میانی ذخیره می‌کند.

مثال) جدول B

مقایسه جدول پایه و میانی		
	جدول پایه	جدول میانی
چه کسی ایجاد می کند؟	طراح پایگاه‌داده	DBMS
چه مدت می‌ماند؟	دایمی	موقت
با چه دستوری ایجاد می‌شود؟	Create table	خروجی هر پرس‌وجو
کاربران چه کارهایی می‌توانند انجام دهند؟	خواندن، درج، حذف، بروزرسانی	خواندن

مثال) نام دانشجویانی که در کلاس AI شرکت کرده‌اند ولی درس پایگاه داده را خریده‌اند.

شکل اول)

```
SELECT B.sname
FROM buy  AS B, Att AS A
WHERE B.sname = A.sname and B.cname = 'DB' and A.cname = 'AI'
```

شکل دوم)

```
SELECT B.sname
FROM ( SELECT *
            FROM buy
                  WHERE  cname = 'DB') AS B, Att AS A
WHERE B.sname = A.sname and A.cname = 'AI'
```

شکل سوم)

```
SELECT B.sname
FROM buy AS B,
      (  SELECT *
      FROM Att
      WHERE cname = 'AI' ) AS A
WHERE B.sname = A.sname and B.cname = 'DB'
```

شکل چهارم)

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

```
SELECT B.sname
FROM ( SELECT *
            FROM buy
                  WHERE  cname = 'DB') AS B,
      (  SELECT *
      FROM Att
      WHERE cname = 'AI' ) AS A
WHERE B.sname = A.sname
```

نکته) do select as early as possible

Do product as late as possible

عملگرهای ویژه

این عملگرها کار نوشتن پرس‌وجوها را راحت‌تر می‌کند. مزایای استفاده از این عملگرها به شرح زیر است.

1. ساده سازی 2. خلاصه‌سازی 3. کمک به درک سریعتر

در بین این عملگرها exists، not exists، in و not in مهمتر هستند.

یادآوری) ساختار کلی دستور Select

ستون‌ها Select

جدول‌ها Form

شرط‌ها Where

Between

Att				
Sname	*cname*	Meeting	hour	Attno
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Amin	AI	3	3.5	1790
sara	AI	2	4	2040

مثال) مطلوب است دانشجویانی که بیش از 2 و کمتر از 4 ساعت در کلاسها حضور داشته‌اند.

```
SELECT sname
FROM att
WHERE hour > 2 AND HOUR < 4
```

این پرس‌وجو را با between می‌توان بازنویسی کرد.

```
SELECT sname
FROM att
WHERE hour BETWEEN 2 AND 4
```

Is null و is not null

مثال) مطلوب است دانشجویانی که نام دانشگاه برای آنها ثبت‌شده‌است.

Student		
sname	city	Univ
Ali	Tehran	Sharif
Reza	Sari	Kerman
Sara	Bam	Null
Amin	Sari	Kerman

```
SELECT sname
FROM student
WHERE univ IS NOT NULL
```

با توجه به این‌که null با null مساوی نیست. این پرس‌وجو را به شکل زیر هم می‌توان نوشت.

```
SELECT sname
FROM student
WHERE univ=univ
```

مثال) مطلوب است دانشجویانی که نام دانشگاه آنها معلوم نیست.

```
SELECT sname
FROM student
WHERE univ IS NULL
```

Like و not like

مثال) مطلوب است، نام دانشجویانی که نام آنها به حرف A ختم می‌شود.

```
SELECT sname
FROM student
WHERE sname IS LIKE '%A'
```

نکته: در برخی از پیاده‌سازی‌ها ممکن است IS وجود نداشته‌باشد. و به شکل زیر نوشته شود.

```
SELECT sname
FROM student
WHERE sname LIKE '%A'
```

مثال) مطلوبست دانشجویانی که نام آنها با حرف A شروع نمی‌شود.

```
SELECT sname
FROM student
WHERE snmae IS NOT LIKE 'A%'
```

مثال) مطلوبست دانشجویانی که نام شهر آنها شامل حرف A می‌شود.

```
SELECT sname
FROM student
WHERE city IS LIKE '%A%'
```

مثال) مطلوبست دانشجویانی که نام آنها چهار کاراکتری است.

```
SELECT sname
FROM student
WHERE city IS LIKE '----'
```

نکته: امکانات نوشتن رشته‌های کاراکتری در پیاده‌سازی‌های مختلف خیلی با هم متفاوت است.
توصیه‌می‌شود به مستندات پیاده‌سازی‌ها مراجعه‌کنید.

in و Not in

مثال) خروجی query زیر چیست؟

```
SELECT city
FROM student
WHERE sname IN ('Ali', 'Reza', 'Sara')
```

پاسخ) شهر ali و sara ، reza از جدول student نمایش می‌دهد.

Student		
sname	City	Univ
Ali	Tehran	Sharif
Sara	Bam	Tabriz

مثال)نام و شهر دانشجویانی که در کلاس AI حضور داشته‌اند.

student		
sname	City	Univ
Ali	Tehran	Sharif
Reza	Sari	Kerman
Sara	Bam	Tabriz
Amin	Kish	Babol

Att				
Sname	*Cname*	meeting	Hour	**Attno**
Ali	DB	2	2.5	1576
Reza	AI	1	1.5	1760
Ala	AI	3	3.5	1790
Sara	AI	Null	null	2040

اطلاعات شهر دانشجویان در جدول student و اطلاعات حضور در جدول att است. در گذشته این دو جدول را با ضرب دکارتی ترکیب‌کرده و جواب را به دست آوردیم. حالا می‌خواهیم از in استفاده کنیم.

گام اول) نام دانشجویانی که در کلاس AI شرکت کرده‌اند را از att بدست می‌آوریم.

```
SELECT sname
FROM att
WHERE cname = 'AI'
```

Sname
Reza
Ala
Sara

گام دوم) حالا با IN از مجموعه بدست آمده با query بالا در نوشتن query نهایی استفاده می‌کنیم.

```
SELECT city
FROM student
WHERE sname IN (SELECT sname
                FROM att
                WHERE cname =
'AI'
                )
```

تعریف زیرپرس‌وجو) اگر یک جفت select ..from داخل بدنه یک select...from دیگر باشد، به جفت select..from داخلی، زیرپرس‌جو subquery گفته می‌شود. به پرس‌وجوی اصلی پرس‌وجوی والد (پدر یا مادر) گفته می‌شود.

سوال) در مثال‌های قبلی کجا از زیرپرس‌وجو استفاده کرده بودیم؟

پاسخ) مثلا در اینجا

```
SELECT B.sname
FROM buy AS B,
      (  SELECT *
       FROM Att
       WHERE cname = 'AI' ) AS A
WHERE B.sname = A.sname and B.cname = 'DB'
```

نکته) نوع و معنای مجموعه‌ای بعد از IN باید هم‌نوع و هم‌معنا با خصیصه‌ی قبل از IN باشد. مثلا sname نام دانشجو است. و زیرپرس‌وهم مجموعه از نام دانشجویان است. پس درست است.

سوال) نام دانشجویانی اهل Bam که در کلاس DB حضور داشته‌اند ولی هیچ کدام از درس‌های دکتر خانی را نخریده‌اند.

student			course		buy				Att				
sname	city	univ	cname	tname	sname	cname	amount	trno	Sname	cname	meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

پاسخ)

```
SELECT sname
FROM att
WHERE cname = 'DB'
```

دانشجویانی که در درس DB حضور داشته‌اند

```
AND sname IN (SELECT sname
                    FROM student
                    WHERE city = 'Bam'
```

اهل بم هستند

```
                    AND sname NOT IN (
                           SELECT sname
                           FROM buy
                           WHERE cname IN (
                                  SELECT cname
                                  FROM course
                                  WHERE tname = 'Khani')
                                  )
```

و در بین دانشجویانی که درسهای دکتر خانی را خریده‌اند نیستند.

سوال) پرس و جوی زیر چه کار می‌کند؟

```
SELECT sname
FROM att
WHERE cname IN ( SELECT CITY
                           FROM student)
```

پاسخ) این پرس‌وجو بی‌معنی و غلط است. زیرا cname اسم درس نمی‌تواند عضوی از مجموعه city شهرها باشد. توجه کنید با اینکه پرس‌وجو بی‌معنی است، ولی به خاطر اینکه cname و city هر دو رشته کاراکتری و هم‌نوع هستند، خطای نحوی نخواهیم داشت.

سوال) پر و جوی زیر چه کار می‌کند؟

```
SELECT sname
FROM att
WHERE cname IN ( SELECT attno
                           FROM att)
```

پاسخ) این پرس‌وجو خطای نحوی تولید می‌کند. زیر cname از نوع رشته کاراکتری است ولی attno از نوع int است.

نکته) نوع و معنای مجموعه‌ای بعد از IN باید هم‌نوع و هم‌معنا با خصیصه‌ی قبل از IN باشد. مثلا sname نام دانشجو است. و زیرپرس‌وهم مجموعه از نام دانشجویان است. پس درست است.

در ویدیو مثالهای بیشتری حل شده است. عملگر all هم گفته شده است

Some

مثل IN در بخش شرط نوشته می‌شود.

مثل IN باید دوطرف این عملگر هم‌نوع باشند.

مثال) نام و شهر دانشجویانی که در کلاس AI حضور داشته‌اند.

پیاده سازی some	پیاده سازی in

```sql
SELECT city
FROM student
WHERE sname IN (SELECT sname
                FROM att
                WHERE cname = 'AI'
                )
```

```sql
SELECT city
FROM student
WHERE sname = SOME (SELECT sname
                    FROM att
                    WHERE cname = 'AI'
                    )
```

کاربرد اصلی some در مقایسه‌های عددی دیده می‌شود.

Exists و not Exists

مثال) مطلوبست نام دانشجویانی که در کلاسها را خریده‌اند و هم در کلاسهای حاضر بوده‌اند.

پاسخ) نام دانشجویانی که کلاسها را خریده‌اند در جدول att است. `SELECT sname FROM att`

نام دانشجویانی که کلاسها را خریده‌اند در جدول buy است. `SELECT sname FROM buy`

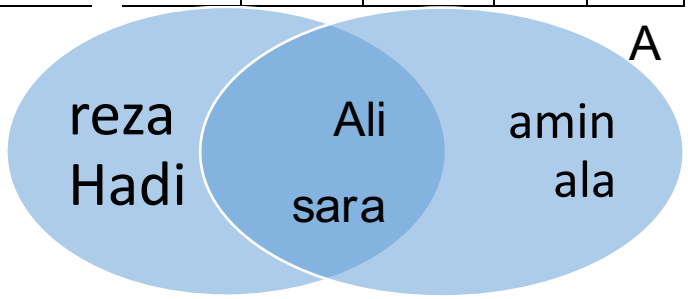
حالا باید با intersect اشتراک این دو مجموعه را حساب کنیم.

```sql
(SELECT sname FROM att) INTERSECT (SELECT sname FROM buy)
```

نام دانشجویانی که در کلاسها حاضر بوده‌اند در جدول att است.

Buy			
Sname	*Cname*	Tr_no	Amount
Ali	AI	1720	300
Reza	DB	1500	550
Sara	AI	1755	430
Hadi	DB	3466	550

Att				
Sname	*Cname*	meeting	Hour	Attno
Ali	DB	2	2.5	1576
Amin	AI	1	1.5	1760
Ala	AI	3	3.5	1790
Sara	AI	Null	null	2040



برای اینکه با exists بهتر آشنا شویم. این پرس‌وجو را این‌بار بدون استفاده از intersect می‌نویسیم.

فرم 1) هر عضوی از A که در B هم وجود داشته‌باشد

1. آیا ali در B هست؟ بلی
2. آیا amin در B هست؟ خیر
3. ایا sara در B هست؟ بلی
4. آیا ala در B هست؟ خیر

فرم 2) هر عضو A که وجود دارد عضوی از B که همنام باشد.

فرم 3) هر sname سطری (عضو) att که وجود داشته باشد سطری (عضوی) ازbuy که داشته باشیم
buy.sname = att.sname

فرم 4) نوشتن با استفاده از exists

```
SELECT sname
FROM att
WHERE EXISTS (SELECT *
              FROM buy
              WHERE buy.sname = att.sname)
```

اینجا هم زیر پرس‌وجو داریم. سوال این زیر پرس‌وجو با زیرپرس‌وجوهای قبلی چه تفاوتی دارد.

```
SELECT sname
FROM att
WHERE EXISTS (SELECT *
              FROM buy
              WHERE buy.sname = att.sname)
```

```
SELECT city
FROM student
WHERE sname IN (SELECT sname
                FROM att
                WHERE cname = 'AI'
                )
```

پاسخ) در قسمت where زیرپرس‌وجو سمت چپ، قسمتی از FROM پرس‌وجوی مادر آمده است. اصطلاحا گفته می‌شود زیرپرس‌وجو به پرس‌وجوی مادر متصل شده‌است. و به att می‌گوییم نقطه اتصال.

تعریف پرس‌وجوی متصل) اگر زیر پرس‌وجو در بخش شرط به پرس‌وجوی مادر مرتبط باشد. پرس‌وجوی متصل (Correlated query) گفته می‌شود.

تعریف زیرپرس‌وجوی مستقل) زیر پرس‌وجوی بدون نقطه اتصال، پرس‌وجوی مستقل نامیده می‌شود.

نکته) برای عملگرهای exists و not exists وجود نقطه اتصال الزامی است.

به فرم زیر دوباره نگاه کنید.

فرم 1) هر عضوی از A که در B هم وجود داشته‌باشد

1. آیا ali در B هست؟ بلی
2. آیا amin در B هست؟ خیر
3. ایا sara در B هست؟ بلی
4. آیا ala در B هست؟ خیر

سوال) اگر att به جای 4 سطر 4000 سطر داشت، به چند سوال باید جواب می‌دادید.

نکته) استفاده از پرس‌وجوهای متصل هزینه اجرا را بالا می برد.

```
SELECT sname
FROM att
WHERE EXISTS (SELECT *
              FROM buy
              WHERE buy.sname = att.sname)
```

سوال) به نظر شما این دو پرس‌جو با پرس‌وجوی بالا چه تفاوتی دارد؟

```
SELECT sname FROM att
WHERE EXISTS ( SELECT trno
       FROM buy
       WHERE buy.sname = att.sname )
```

```
SELECT sname FROM att
WHERE EXISTS ( SELECT sname
       FROM buy
       WHERE buy.sname = att.sname )
```

پاسخ) هیچ تفاوتی ندارد. هر سه نام دانشجویانی که هم دوره‌ها را خریده‌اند و هم در دوره‌ها شرکت کرده‌اند را نمایش می‌دهد.

نکته) پرس‌وجوی با exists را به گونه‌ی دیگر هم می‌توان تفسیر کرد. سطرهایی از att که به ازای آن عبارت (مجموعه) بعد از exists تهی نباشد.

نکته) اگر به جای exists از not exists استفاده کنیم می شود. سطرهایی از att که به ازای آن عبارت (مجموعه) بعد از exists تهی باشد.

مثالی که تا اینجا گفتیم فقط برای آشنایی با exists بود. هنر اصلی exists در شبیه‌سازی تقسیم دیده می‌شود. در sql عملگر تقسیم جبر رابطه‌ای پیاده‌سازی نشده و باید آن را شبیه‌سازی کنیم.

پیاده سازی اول تقسیم EXCEPT ... NOT EXISTS

مثال) مطلوبست دانشجویانی که همه کلاسها را خریده‌اند.

Buy				Cname		student		
sname	***Cname***	amount	**Trno**	**cname**	tname	**sname**	city	uni
Reza	AI	520	3448	AI	Razavi	Ala	Neka	Azad
Hadi	AI	550	5654	DB	Khani	Ali	Bam	UoT
Amin	DB	600	7074			Amin	Qom	Azad
Hadi	DB	500	8997			Hadi	Bam	Sharif
Ala	AI	430	9097			Reza	Sari	Shahed
						Sara	Qom	Sharif

پاسخ) Hadi است. ولی ببینیم چه جوری به آن می‌توانیم برسیم.

اول) لیست دانشجویان {Ala, Ali, Amin, Hadi, Reza, Sara}

دوم) لیست دوره‌ها را از course به دست می‌آوریم.

سوم) لیست خرید هر دانشجو را از buy به‌دست می‌آوریم.

Ala = {AI}, Ali ={}, Amin={DB}, Hadi={AI,DB}, Reza={AI}, Sara={}

چهارم) از لیست دوره‌ها، لیست خرید هر دانشجو را کم کنید.

Ala = {AI,DB}-{AI} = {DB}

Ali = {AI,DB}-{} = {AI,DB}

Amin= {AI,DB}-{DB} = {AI}

Hadi={AI,DB}-{AI,DB} = {}

Reza = {AI,DB}-{AI} = {DB}

Sara = {AI,DB}-{} = {AI,DB}

هر دانشجویی که نتیجه تفاضل لیست خریدش از لیست دوره‌ها تهی باشد، همه دوره‌ها را خریده‌است.

حالا همین جمله را به زبان SQL می‌نویسیم.

```
SELECT student.sname
FROM student
WHERE NOT EXISTS ( SELECT cname FROM course
                        EXCEPT
                   SELECT buy.cname FROM buy
                   WHERE student.sname = buy.sname)
```

نکته) برای ساده‌سازی از Rename استفاده می‌کنیم.

```
SELECT S.sname
FROM student S
WHERE NOT EXISTS ( SELECT cname FROM course
                        EXCEPT
                   SELECT B.cname FROM buy B
                   WHERE S.sname = B.sname)
```

یادآوری) دقت کنید که همچنان نقطه اتصال داریم.

نکته) این اولین روش پیاده‌سازی (شبیه‌سازی) تقسیم با دستورات SQL است. این روش از not exists و except استفاده می‌کند.

سوال) دانشجویانی که همه کلاسهای دکتر خانی را خریده‌اند؟

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

پاسخ) این پرس‌وجو شبیه پرس‌وجوی قبل است. فقط به جای همه کلاسها گفته همه کلاسهای دکتر خانی.

پس هر دانشجویی که نتیجه تفاضل لیست خریدش از لیست دوره‌های دکتر خانی تهی باشد، همه دوره‌ها را خریده‌است.

```
SELECT S.sname
FROM student S
WHERE NOT EXISTS ( SELECT cname FROM course  WHERE tname = 'Khani'
                        EXCEPT
                   SELECT B.cname FROM buy B
                   WHERE S.sname = B.sname)
```

پیاده سازی دوم تقسیم NOT EXISTS ... NOT EXISTS

عملگر تقسیم را به شکل دومی هم می‌توان پیاده‌سازی کرد.

سوال) دانشجویانی که همه کلاسهای دکتر خانی را خریده‌اند؟

گام اول) شرح پرس‌وجو را به شکل زیر بازنویسی می‌کنیم.

دانشجویانی

که وجود ندارد درسی از دکتر خانی

که آن دانشجویان آن درس را نخریده باشند.

گام دوم) بازنویسی دیگر

دانشجویانی

که وجود ندارد درسی از دکتر خانی

که آن دانشجویان آن درس در لیست خریدشان وجود نداشته باشد.

گام سوم) باز نویسی با زبان SQL

```
SELECT S.sname
FROM student S
WHERE NOT EXISTS (SELECT C.cname
                  FROM course C
                  WHERE tname = 'Khani'
                  AND NOT EXISTS ( SELECT B.cname
                                   FROM buy B
                                   WHERE B.cname = C.cname
                                   AND S.sname = B.sname)
```

نکته) در این پیاده سازی چون دو تا NOT EXISTS داریم، دو نقطه اتصال لازم داریم.

توابع عددی در SQL

تابع Min

مثال) کمترین مبلغ پرداخت شده برای خرید کلاسها را مشخص کنید.

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

```
SELECT MIN (amount)
FROM buy
WHERE cname = 'AI'
```

نکته) این تابع یک ستون را به عنوان ورودی گرفته و مقدار کمینه را به خروجی می‌دهد.

نکته) از نظر نحوی syntax توابع عددی بعد از SELECT ظاهر می‌شوند.

تابع MAX

مثال) طولانی ترین حضور در کلاس DB دانشجویانی که اهل Bam هستند.

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

```
SELECT MAX (hour)
FROM Att
WHERE cname = 'DB'
AND snmae IN ( SELECT sname
                 FROM student
                   WHERE city = 'Bam')
```

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

نکته) تابع عددی روی تمامی سطرها اعمال نمی‌شود. با شرط‌های نوشته شده، تعدادی از سطرها انتخاب شده و تابع فقط روی آن سطرها اعمال می‌شود.

تابع AVG

مثال) میانگین رقم پرداخت شده برای خرید چند بوده‌است؟

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

```
SELECT AVG (amount)
FROM Buy
```

مثال) دانشجویانی که درس AI را خریده‌اند و از میانگین رقم خرید همه دانشجویان بیشتر است.

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	1	3.5	6873
									Amin	DB	2	1	7365

```
SELECT sname
FROM Buy
WHERE amount > AVG (amoutn)
and cname = 'AI'
```

نکته) این عبارت SQL خطای نحوی دارد. AVG مانند بقیه توابع عددی باید بعد از SECLECT بیاد.

پاسخ صحیح)

```
SELECT sname
FROM Buy
WHERE amount >  (SELECT AVG (amount)
                 FROM Buy)
and cname = 'AI'
```

سوال) در عبارت بالا پرس‌وجوی مادر و فرزند را مشخص کنید؟

نکته) خروجی هر SELECT یک جدول است به غیر از 1) توابع عددی 2) exists, not exists

تابع Count

مثال) تعداد حضورهای درس DB را مشخص کنید.

student			course		Buy				Att				
sname	city	univ	cname	tname	**Sname**	**cname**	amount	Trno	**Sname**	**cname**	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	Null	Null	6873
									Amin	DB	2	1	7365

پرس‌وجوی اول)

```
SELECT COUNT (*)
FROM Att
WHERE cname = 'DB'
```

پرس‌وجوی دوم)

```
SELECT COUNT (Meet)
FROM Att
WHERE cname = 'DB'
```

در پرس‌وجوی اول سطر Ala هم شمرده می شود. و خروجی 6 خواهد بود.

در پرس‌وجوی دوم سطر Ala شمرده نشده و خروجی 5 خواهد بود.

نکته) در اجرای توابع عددی روی ستونی از جدول، عملیات مربوطه بدون در نظر گرفتن مقادیر null در آن ستون انجام می‌گیرد.

مثال) خروجی پرس‌وجوی زیر چیست؟

```
SELECT AVG (hour)
FROM Att
```

student			course		Buy				Att				
sname	city	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	Null	Null	6873
									Amin	DB	2	1	7365

الحاق Join در SQL

یادآوری) در جبر رابطه‌ای دیدیم که جدولها را می‌توان به روش‌های زیر با هم الحاق join کرد. و جدول‌های جدیدی به دست آورد. در این فصل می‌خواهیم ببینیم این الحاق‌ها در SQL چگونه نوشته می‌شود.

جمع‌بندی علمگرهای الحاق				
	جابجایی	پیش شرط	سمبل	عملگر
الحاق تمام سطرها، تمامی ستونها	دارد	ندارد	A X B	ضرب دکارتی
الحاق سطرهای الحاق پذیر، حذف ستونهای تکراری	دارد	ندارد	$A \bowtie B$	الحاق طبیعی
Select زدن روی خروجی الحاق	دارد	ندارد	$A \bowtie_\theta B$	تتا جوین
حذف ستونهای B از خروجی الحاق	ندارد	ندارد	$A \ltimes B$	شبه الحاق
اضافه کردن سطرهای الحاق‌ناپذیر B به خروجی الحاق	ندارد	ندارد	$A \bowtie B$	الحاق خارجی راست
اضافه کردن سطرهای الحاق‌ناپذیر A به خروجی الحاق	ندارد	ندارد	$A \rtimes B$	الحاق خارجی چپ
اضافه کردن سطرهای الحاق‌ناپذیر A و B به خروجی الحاق	دارد	ندارد	$A \rtimes B$	الحاق خارجی کامل

ضرب دکارتی A X B

```
SELECT * FROM Employee, Dept
```

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Finance	George
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Finance	George
5	Mary	1257	HR	Finance	George
6	Hary	3415	Finance	Sales	Harriet
7	Sally	2241	Sales	Sales	Harriet
8	George	3401	Finance	Sales	Harriet
9	Harriet	3415	Sales	Sales	Harriet
10	Mary	1257	HR	Sales	Harriet
11	Hary	3415	Finance	Production	Charles
12	Sally	2241	Sales	Production	Charles
13	George	3401	Finance	Production	Charles
14	Harriet	3415	Sales	Production	Charles
15	Mary	1257	HR	Production	Charles

یاداوری شرط ضرب) سطرهایی که در ستونهای مشترک، مقادیر یکسان دارند. این سطرهای با معنی هستند. وقتی شرط ضرب را روی ضرب دکارتی اعمال می‌کنیم سطرهای بی معنی حذف می‌شوند.

یادآوری سطرهای الحاق پذیر) سطرهایی که در ستونهای مشترک، مقادیر یکسان دارند.

```
SELECT *
FROM Employee, Dept
WHERE Employee.DeptName = Dept.DeptName
```

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Sales	Harriet
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Sales	Harriet

الحاق طبیعی $A \bowtie B$

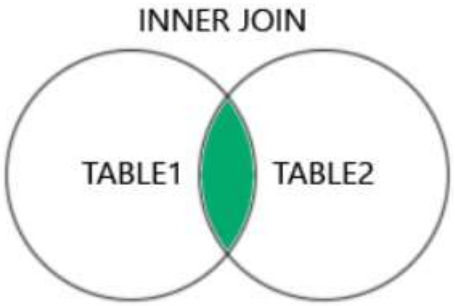
الحاق طبیعی همان ضرب دکارتی است که شرط ضرب روی آن اعمال شده‌است. آن را به شکل زیر می‌نویسند.

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Sales	Harriet
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Sales	Harriet

```
SELECT *
FROM Employee
INNER JOIN Dept ON Employee.DeptName = Dept.DeptName;
```



نکته) در نوشتن شرط الحاق با ON لازم نیست ستون مشترک هم‌نام باشد.

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	Department	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	Department	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Sales	Harriet
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Sales	Harriet

```
SELECT *
FROM Employee INNER JOIN Dept
ON Employee.DeptName = Dept.Department;
```

نکته) در بعضی از پیاده‌سازی الحاق طبیعی به شکل زیر نوشته می‌شود. natural شرط الحاق است. در شکل زیر ستون مشترک بر اساس رابطه‌ای که کلید خارجی ایجاد می‌کند، تشخیص داده می‌شود.

```
Employee NATURAL INNER JOIN Dept
Employee NATURAL JOIN Dept
```

نکته) در بعضی از پیاده‌سازی شرط الحاق به جای ON با USING نوشته می‌شود. به هنگام استفاده از USING حتمال باید ستون مشترک هم‌نام باشند.

```
SELECT *
FROM Employee INNER JOIN Dept
USING (DeptName)
```

نکته)برای نوشتن شرط الحاق از بین ON ، USING و NATURAL بهتر است از ON استفاده کنید.

نکته) در بعضی از پیاده‌سازی‌ها شرط در inner join اجباری نیست. اگر در inner join هیچ شرطی نداشته‌باشیم. Inner join همان ضرب دکارتی خواهد بود.

تتاجوین A⋈θB

```
SELECT Employee.Name, Employee.DeptName, Dept.Manager
FROM Employee INNER JOIN Dept
ON Employee.DeptName = Dept.DeptName
WHERE Employee.DeptName = 'Sales'
```

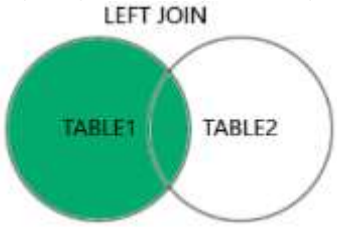
Name	DeptName	Manager
Sally	Sales	Harriet
Harriet	Sales	Harriet

الحاق خارجی سمت چپ

```
SELECT *
FROM Employee LEFT OUTER JOIN Dept
ON Employee.DeptName = Dept.DeptName;
```

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles



	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Sales	Harriet
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Sales	Harriet
5	Mary	1257	HR	NULL	NULL

الحاق خارجی سمت راست

```
SELECT *
FROM Employee RIGHT OUTER JOIN Dept
ON Employee.DeptName = Dept.DeptName;
```

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	George	3401	Finance	Finance	George
3	Sally	2241	Sales	Sales	Harriet
4	Harriet	3415	Sales	Sales	Harriet
5	NULL	NULL	NULL	Production	Charles

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

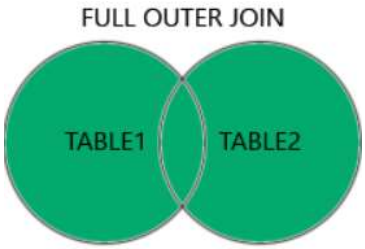
الحاق خارجی کامل

```
SELECT *
FROM Employee FULL OUTER JOIN Dept
ON Employee.DeptName = Dept.DeptName;
```

	Name	EmpId	DeptName
1	Hary	3415	Finance
2	Sally	2241	Sales
3	George	3401	Finance
4	Harriet	3415	Sales
5	Mary	1257	HR

	DeptName	Manager
1	Finance	George
2	Sales	Harriet
3	Production	Charles

	Name	EmpId	DeptName	DeptName	Manager
1	Hary	3415	Finance	Finance	George
2	Sally	2241	Sales	Sales	Harriet
3	George	3401	Finance	Finance	George
4	Harriet	3415	Sales	Sales	Harriet
5	Mary	1257	HR	NULL	NULL
6	NULL	NULL	NULL	Production	Charles



گروهبندی Group By

مثال) تعداد خریدهای هر درس را مشخص کنید.

Student			course		Buy				Att				
sname	City	univ	cname	tname	Sname	cname	amount	Trno	Sname	cname	Meet	Hour	Attno
Ala	Neka	Azad	AI	Razavi	Sara	AW	300	2044	Ali	DB	5	1.5	1325
Ali	Bam	UoT	DB	Khani	Reza	GS	400	2345	Sara	DB	1	1.2	1345
Amin	Qom	Azad	AW	Khani	Reza	AI	520	3448	Hadi	DB	4	2.3	1444
Hadi	Bam	Sharif	GS	Khani	Hadi	AI	550	5654	Reza	AI	2	2.5	1750
Reza	Sari	Shahed			Amin	DB	600	7074	Reza	DB	3	1	2335
Sara	Qom	Sharif			Hadi	DB	500	8997	Reza	AI	3	4	5052
					Ala	AI	430	9097	Sara	AI	1	2.2	5456
									Ala	DB	Null	Null	6873
									Amin	DB	2	1	7365

	cname	(No column name)
1	AI	3
2	AW	1
3	DB	2
4	GS	1

پاسخ)

```
SELECT cname, COUNT(*)
FROM buy
GROUP BY cname
```

نکته) cname بین تابع عددی COUNT و SELECT فاصله انداخته است. چون از GROUP BY استفاده‌شده این خطای نحوی نمی‌دهد.

نکته) سطرهای بر اساس مقدار سطر cname گروهبندی شده و سپس تابع عددی اعمال می‌شود.

مثال) مجموع ساعت‌های حضور هر دانشجو را حساب کنید.

```
SELECT sname, SUM(hour)
       FROM att
       GROUP BY sname
```

	sname	(No column name)
1	Ala	NULL
2	Ali	1.5
3	Amin	1
4	Hadi	2.3
5	Reza	7.5
6	Sara	3.4

سوال) خروجی پرس‌وجوی زیر چیست؟

```
SELECT  SUM(hour)
       FROM att
       GROUP BY sname
```

پاسخ)

	(No column name)
1	NULL
2	1.5
3	1
4	2.3
5	7.5
6	3.4

سوال) خروجی پرس‌وجوی زیر چیست؟

```
SELECT  hour
       FROM att
       GROUP BY sname
```

پاسخ) خطای نحوی تولید می‌کند.

مثال) مطلوبست جمع خرید(فروش) ها به تفکیک هر استاد

پاسخ)

در ابتدا سعی می کنیم منظور پرس‌وجو را درک کنیم.

	tname	(No column name)
1	Khani	1720
2	Razavi	1500

پاسخ)

گام اول نام استادها در course و رقم فروش در buy پس باید این دو جدول با هم ضرب دکارتی بشوند

```
SELECT *
FROM course  C, Buy  B
```

	cname	tname
1	Al	Razavi
2	AW	Khani
3	DB	Khani
4	GS	Khani

	sname	cname	amount	trno
1	Sara	AW	300	2044
2	Reza	GS	400	2345
3	Reza	Al	520	3448
4	Hadi	Al	550	5654
5	Amin	DB	600	7074
6	Hadi	DB	500	8997
7	Ala	Al	430	9097

	cname	tname	sname	cname	amount	trno
1	Al	Razavi	Sara	AW	300	2044
2	Al	Razavi	Reza	GS	400	2345
3	Al	Razavi	Reza	Al	520	3448
4	Al	Razavi	Hadi	Al	550	5654
5	Al	Razavi	Amin	DB	600	7074
6	Al	Razavi	Hadi	DB	500	8997
7	Al	Razavi	Ala	Al	430	9097
8	AW	Khani	Sara	AW	300	2044
9	AW	Khani	Reza	GS	400	2345
10	AW	Khani	Reza	Al	520	3448
11	AW	Khani	Hadi	Al	550	5654
12	AW	Khani	Amin	DB	600	7074
13	AW	Khani	Hadi	DB	500	8997
14	AW	Khani	Ala	Al	430	9097
15	DB	Khani	Sara	AW	300	2044
16	DB	Khani	Reza	GS	400	2345
17	DB	Khani	Reza	Al	520	3448
18	DB	Khani	Hadi	Al	550	5654
19	DB	Khani	Amin	DB	600	7074
20	DB	Khani	Hadi	DB	500	8997
21	DB	Khani	Ala	Al	430	9097
22	GS	Khani	Sara	AW	300	2044
23	GS	Khani	Reza	GS	400	2345
24	GS	Khani	Reza	Al	520	3448
25	GS	Khani	Hadi	Al	550	5654
26	GS	Khani	Amin	DB	600	7074
27	GS	Khani	Hadi	DB	500	8997
28	GS	Khani	Ala	Al	430	9097

گام دوم) شرط ضرب دکارتی را اعمال کرده تا سطرهای بی معنی حذف شود.

```
SELECT C.tname,C.cname, B.cname,  B.amount
FROM course  C, Buy  B
WHERE C.cname = B.cname
```

	tname	cname	cname	amount
1	Khani	AW	AW	300
2	Khani	GS	GS	400
3	Razavi	Al	Al	520
4	Razavi	Al	Al	550
5	Khani	DB	DB	600
6	Khani	DB	DB	500
7	Razavi	Al	Al	430

گام سوم) حالا GROUP BY برای جمع کردن جمع فروش هر گروه اضافه می‌کنیم.

```
SELECT C.tname, SUM (B.amount)
FROM course  C, Buy  B
WHERE C.cname = B.cname
GROUP BY C.tname
```

	tname	(No column name)
1	Khani	1800
2	Razavi	1500

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.

	tname	(No column name)
1	Khani	1720
2	Razavi	1500

مثال) تعداد ساعتهای حضور هر دانشجو برای درس DB را حساب کنید.

پاسخ)

	sname	(No column name)
1	Ala	3.5
2	Amin	1
3	Reza	2.5
4	Sara	2.7

گام اول) سطرهای درس DB را از جدول att استخراج می‌کنیم.

```
SELECT sname, hour
FROM att
WHERE cname = 'DB'
```

	sname	hour
1	Sara	1.2
2	Sara	1.5
3	Reza	1
4	Reza	1.5
5	Ala	3.5
6	Amin	1

گام دوم) با GROUP BY، بر اساس sname گروه‌بندی کرده و جمع ساعتهای هر گروه را حساب می‌کنیم.

	sname	(No column name)
1	Ala	3.5
2	Amin	1
3	Reza	2.5
4	Sara	2.7

مثال) مطلوبست دانشجویانی که حضور آنها بیشتر از 3 ساعت است.

در اینجا شرط باید از گروهبندی و محاسبه ساعت حضور اعمال شود پس WHERE کاربرد ندارد.

گام اول) ساعت حضور هر دانشجو را حساب می‌کنیم

```
SELECT sname, SUM (hour)
FROM att
GROUP BY sname
```

	sname	(No column name)
1	Ala	3.5
2	Amin	1
3	Reza	9
4	Sara	4.9

گام دوم) شرط را روی محاسبه‌ها اعمال می‌کنیم.

```
SELECT sname, SUM (hour)
FROM att
GROUP BY sname
HAVING SUM (hour) > 3
```

	sname	(No column name)
1	Ala	3.5
2	Reza	9
3	Sara	4.9

پیاده‌سازی تقسیم با GROUP BY

همانطور که پیش از این گفتیم، عملگر تقسیم در SQL وجود ندارد. آن را باید شبیه‌سازی کرد. دو تا شبیه‌سازی (پیاده‌سازی) با not exists گفتیم. اینجا آخرین پیاده‌سازی با GROUP BY گفته می‌شود.

مثال) نام دنشجویانی که در تمامی درس‌ها حضور داشته‌اند.

درک پرس‌وجو) منظور دانشجویانی است که حداقل در یک جلسه یک درس حضور داشته باشند.

لیست درسها عبارت است از

	sname	cname	meet	hour	attno
1	Sara	DB	1	1.2	1345
2	Reza	AI	2	2.5	1750
3	Sara	DB	2	1.5	1788
4	Reza	AI	1	1.7	1844
5	Sara	AW	3	1	2001
6	Sara	GS	2	1.5	2002
7	Ali	GS	1	2.5	2005
8	Ali	AI	1	2.5	2023
9	Ali	AW	1	1.1	2044
10	Ali	DB	2	0.5	2323
11	Reza	DB	3	1	2335
12	Reza	DB	2	1.5	2400
13	Reza	AI	3	4	5052
14	Ala	DB	1	3.5	6873
15	Amin	DB	2	1	7365

	cname	tname
1	AI	Razavi
2	AW	Khani
3	DB	Khani
4	GS	Khani

Reza در 5 جلسه حضور داشته ولی این 5 جلسه برای 2 درس AI و DBاست.

Ali 4 جلسه حضور داشته و این جلسه برای کلاسهای متمایز بوده و در همه کلاسها (AI, AW, DB, GS) حضور داشته است.

گام اول) پیدا کردن تعداد کلاسهای متمایز برای هر دانشجو

```
SELECT  sname , COUNT (DISTINCT cname)
FROM att
GROUP BY sname
```

	sname	(No column name)
1	Ala	1
2	Ali	4
3	Amin	1
4	Reza	2
5	Sara	3

دقت کنید اگر DISTINCT حذف شود. کلاسهای تکراری هم شمرده می‌شود.

```
SELECT  sname , COUNT ( cname)
FROM att
```

```
GROUP BY sname
```

	sname	(No column name)
1	Ala	1
2	Ali	4
3	Amin	1
4	Reza	5
5	Sara	4

گام دوم) بدست آوردن تعداد کل کلاسها

```
SELECT COUNT (cname) FROM course
```

گام سوم) دانشجویانی که تعداد کلاسهای متمایز آنها با کل کلاسها برابر نیست را حذف می‌کنیم.

```
SELECT  sname, COUNT (DISTINCT cname)
FROM att
GROUP BY sname
HAVING COUNT (DISTINCT cname) = (SELECT COUNT (cname) FROM course)
```

	sname	(No column name)
1	Ala	1
2	Ali	4
3	Amin	1
4	Reza	2
5	Sara	3

دقت کنید چون روی نتیجه حسابی هر گروه باید شرط بگذاریم باید از Having استفاده کنیم.

به عنوان گام آخر می‌توان پرس‌وجو را به شکل زیر هم نوشت.

```
SELECT  sname
FROM att
GROUP BY sname
HAVING COUNT (DISTINCT cname) = (SELECT COUNT (cname) FROM course)
```

مثال) دانشجویانی که در تمامی درسهای دکتر خانی حضور داشته‌اند.

درس‌های دکتر خانی در course مشخص شده و حضور در att پس باید course و att در هم ضرب دکارتی شوند. Att ، 15 سطر و course 4 سطر دارد، پس ضرب دکارتی 60 سطر دارد. با اعمال شرط ضرب دکارتی سطرهای نتیجه 15 می شود.

```
SELECT *
FROM att A, course C
WHERE A.cname = C.cname
```

	sname	cname	meet	hour	attno	cname	tname
1	Sara	DB	1	1.2	1345	DB	Khani
2	Reza	AI	2	2.5	1750	AI	Razavi
3	Sara	DB	2	1.5	1788	DB	Khani
4	Reza	AI	1	1.7	1844	AI	Razavi
5	Sara	AW	3	1	2001	AW	Khani
6	Sara	GS	2	1.5	2002	GS	Khani
7	Ali	GS	1	2.5	2005	GS	Khani
8	Ali	AI	1	2.5	2023	AI	Razavi
9	Ali	AW	1	1.1	2044	AW	Khani
10	Ali	DB	2	0.5	2323	DB	Khani
11	Reza	DB	3	1	2335	DB	Khani
12	Reza	DB	2	1.5	2400	DB	Khani
13	Reza	AI	3	4	5052	AI	Razavi
14	Ala	DB	1	3.5	6873	DB	Khani
15	Amin	DB	2	1	7365	DB	Khani

گام دوم) ما به سطرهای علاقه داریم که tname آنها دکتر خانی باشد

```
SELECT A.sname, A.cname
FROM att A, course C
WHERE A.cname = C.cname AND C.tname = 'Khani'
```

	sname	cname
1	Sara	DB
2	Sara	DB
3	Sara	AW
4	Sara	GS
5	Ali	GS
6	Ali	AW
7	Ali	DB
8	Reza	DB
9	Reza	DB
10	Ala	DB
11	Amin	DB

گام سوم) جدول بالا را بر اساس sname گروه بندی کرده و تعداد درسهای متمایز هر دانشجو را می شماریم

```
SELECT A.sname, COUNT(DISTINCT A.cname)
FROM att A, course C
WHERE A.cname = C.cname AND C.tname = 'Khani'
GROUP BY sname
```

	sname	(No column name)
1	Ala	1
2	Ali	3
3	Amin	1
4	Reza	1
5	Sara	3

گام چهارم) تعداد درسهای دکتر خانی را می شماریم.

```
SELECT COUNT (cname) FROM course WHERE tname = 'Khani'
```

	(No column name)
1	3

گام پنجم) سطرهای را انتخاب می‌کنیم که تعداد سطرهای متمایز دکتر خانی برای آنها 3 باشد.

```
SELECT A.sname, COUNT(DISTINCT A.cname)
FROM att A, course C
WHERE A.cname = C.cname AND C.tname = 'Khani'
GROUP BY sname
HAVING COUNT (DISTINCT A.cname) = (SELECT COUNT (cname) FROM course WHERE tname = 'Khani')
```

	sname	(No column name)
1	Ali	3
2	Sara	3

جدول مجازی virtual table یا view

فرض کنید جدول زیر با نام Deposit حسابهای مشتریان در شعب مختلف یک بانک را نشان می دهد.

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Sara	Sarv	8500	5151

از نظر امنیتی و سادگی کار بهتر است که کارمندان شعبه Kaj تنها حسابهای شعبه خود را مشاهده کنند. برای این کار جدول مجازی virtual table ایجاد می‌کنیم.

```
CREATE VIEW KajDep1 AS
       SELECT *
       FROM Deposit
       WHERE bname = 'Kaj';
```

جدول پایه

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Sara	Sarv	8500	5151

جدول مجازی

	cname	bname	balance	ano
1	Negin	Kaj	3700	1045
2	Amir	Kaj	7800	1198
3	Ali	Kaj	1700	1734
4	Reza	Kaj	2500	2333
5	Ali	Kaj	7500	3945

مثال) مطلوبست حسابهای Ali در شعبه Kaj

```
SELECT *
FROM KajDep1
WHERE cname = 'Ali'
```

	cname	bname	balance	ano
1	Ali	Kaj	1700	1734
2	Ali	Kaj	7500	3945

مثال) یک حساب جدید برای Reza در شعبه kaj باز کنید.

```
INSERT INTO KajDep1 VALUES ('Reza', 'Kaj' ,  5300, 8123)
```

دقت کنید که هم جدول مجازی و هم جدول پایه (اصلی) تغییر می‌کند.

	cname	bname	balance	ano
1	Negin	Kaj	3700	1045
2	Amir	Kaj	7800	1198
3	Ali	Kaj	1700	1734
4	Reza	Kaj	2500	2333
5	Ali	Kaj	7500	3945
6	Reza	Kaj	5300	8123

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Sara	Sarv	8500	5151
10	Reza	Kaj	5300	8123

سوال) دستور زیر را اجرا کنید. چه تاثیری روی KajDep1 می گذارد، چه تاثیری روی Deposit می‌گذارد؟

```
INSERT INTO Deposit VALUES ('Nazi', 'Kaj' ,  7500, 4243)

SELECT * FROM KajDep1

SELECT * FROM Deposit
```

	cname	bname	balance	ano
1	Negin	Kaj	3700	1045
2	Amir	Kaj	7800	1198
3	Ali	Kaj	1700	1734
4	Reza	Kaj	2500	2333
5	Ali	Kaj	7500	3945
6	Nazi	Kaj	7500	4243
7	Reza	Kaj	5300	8123

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Nazi	Kaj	7500	4243
10	Sara	Sarv	8500	5151
11	Reza	Kaj	5300	8123

نکته) تغییرات روی جدول پایه روی جدول مجازی هم منعکس می‌شود.

نکته) با اینکه با INSERT، UPDATE، DELETE می‌توان محتوای جدول مجازی را تغییرداد ولی این‌کار زیاد توصیه نمی‌شود.

مثال زیر دردسرهای تغییر جدول مجازی را در عمل نشان می‌دهد.!!

مثال) فرض کنید که جدول مجازی به شکل زیر تعریف شود.

```
CREATE VIEW KajDep AS
       SELECT cname, balance, ano
       FROM Deposit
       WHERE bname = 'Kaj';
```

جدول مجازی

	cname	balance	ano
1	Negin	3700	1045
2	Amir	7800	1198
3	Ali	1700	1734
4	Reza	2500	2333
5	Ali	7500	3945

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Sara	Sarv	8500	5151

در گام بعدی

```
INSERT INTO KajDep VALUES ('Hadi', 5300, 8123)
```

```
INSERT INTO KajDep VALUES ('Hadi', 5300, 8123)
```

خروجی دستور زیر عبارت است از

```
SELECT * FROM KajDep
```

	cname	balance	ano
1	Negin	3700	1045
2	Amir	7800	1198
3	Ali	1700	1734
4	Reza	2500	2333
5	Ali	7500	3945

خروجی دستور زیر عبارت است از

```
SELECT * FROM Deposit
```

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	NULL	7500	1717
5	Ali	Kaj	1700	1734
6	Reza	Kaj	2500	2333
7	Ali	Kaj	7500	3945
8	Ali	Afra	1050	4101
9	Mitra	Afra	5200	4172
10	Sara	Sarv	8500	5151
11	Hadi	NULL	5300	8123

با اینکه حساب جدید Ali و Hadi به جدول پایه اضافه شده است. ولی در جدول مجازی دیده نمی‌شود. چرا؟

پاسخ) زیرا ستون bname ، null است.

مثال) خروجی دستورهای زیر چیست؟

```
CREATE VIEW CNTACNT AS
       SELECT cname, COUNT(*) AS cnt
       FROM Deposit
       GROUP BY cname
INSERT INTO CNTACNT VALUES ('Azar', 2)
```

نکته) نامگذاری ستون در view با استفاده از AS

پاسخ) دستور insert خطا تولید می‌کند. زیرا در اعمال این تغییر در جدول پایه غیر ممکن است.

	cname	cnt
1	Ala	1
2	Ali	3
3	Amir	1
4	Mitra	1
5	Nazi	1
6	Negin	1
7	Reza	2
8	Sara	1

نکته) از دستور GROUP BY در تعریف view استفاده شده باشد. نمی‌توان محتوای view را تغییر داد.

مثال) خروجی دستورهای زیر چیست؟

```
CREATE VIEW AVGBalanc AS
SELECT AVG(balance) AS mean
       FROM Deposit
INSERT INTO AVGBalanc VALUES (7500)
```

پاسخ) دستور insert خطا تولید می‌کند. زیرا در اعمال این تغییر در جدول پایه غیر ممکن است.

	mean
1	5298

نکته) اگر در view ستونی داشته باشیم که در جدول پایه نباشد. View را نمی‌توان تغییر داد.

شروط چهارگانه تغییر جدول مجازی

شروط چهارگانه لازم برای انجام delete، update و insert روی جدول مجازی

1. جدول مجازی فقط روی یک جدول پایه تعریف شده‌باشد.
2. از دستور GROUP BY در تعریف view استفاده نشده باشد.
3. ستون‌هایی غایب جدول پایه از نوع not null نباشند.
4. در تعریف view فقط از ستونهای جدول پایه استفاده شده‌باشد.

نکته) همانطور که پیشتر گفته شد، با اینکه با INSERT، UPDATE, DELETE می‌توان محتوای جدول مجازی را تغییرداد ولی این‌کار زیاد توصیه نمی‌شود.

انواع جدول در SQL

1. جدول پایه Base Table: با create ایجاد شده و همه کار می‌توان کرد.
2. جدول میانی Intermediate Table : در حین اجرای پرس‌وجو به شکل موقت ایجاد شده و نتایج میانی را در خود نگه می‌دارد. فقط خواندنی
3. جدول مجازیVirtual Table : با create view ایجاد می‌شود. علاوه بر خواندن می‌توان آن را تغییر داد ولی توصیه نمی‌شود.

نکته) برای حذف جدول مجازی می‌توان نوشت `DROP VIEW KajDep1`

مثال) دستورهای زیر چه کار می‌کند؟

```
CREATE VIEW AliKaj AS
       SELECT *
       FROM KajDep1
       WHERE cname = 'Ali'
```

	cname	bname	balance	ano
1	Ali	Kaj	1700	1734
2	Ali	Kaj	7500	3945

نکته) روی یک جدول مجازی می‌توان جدول مجازی جدید تعریف کرد.

شاخص Index

با تعریف شاخص در بانک اطلاعاتی سرعت بازیابی اطلاعات در پایگاه داده افزایش می‌یابد.

ایده اصلی index بر اساس B-Tree است.

```
CREATE TABLE Customer(
NationalID INT,
Name char(20),
City char (20),
PRIMARY KEY (NationalID)
);
```

```
SELECT * FROM Customer WHERE NationalID = 40

SELECT * FROM Customer WHERE NationalID = 65

SELECT * FROM Customer WHERE NationalID = 120
```

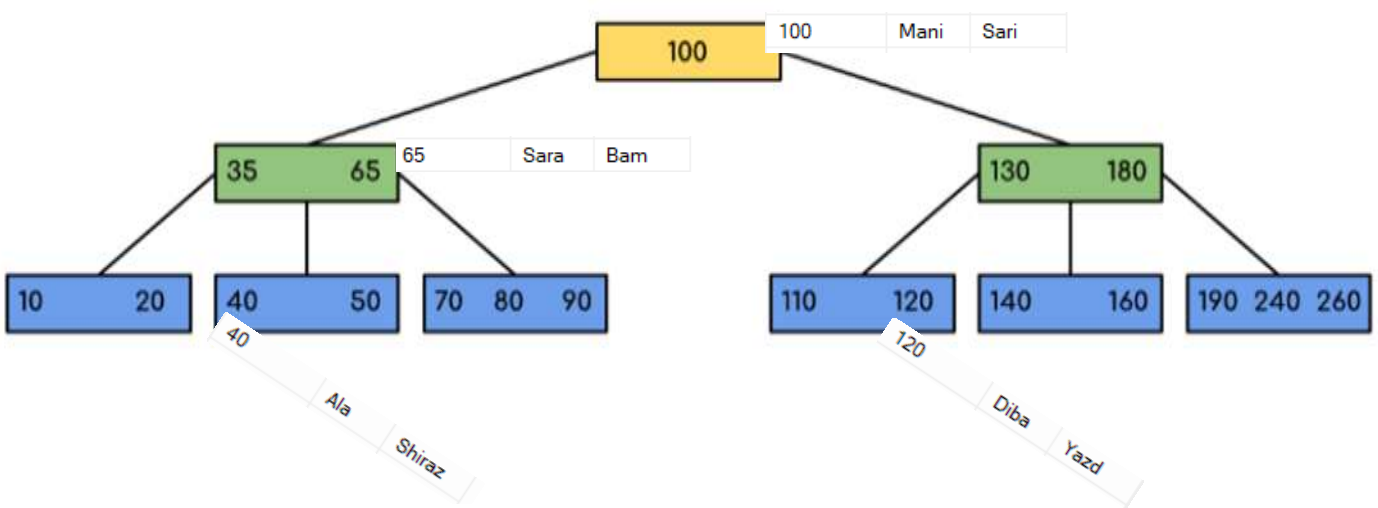
→

	NationalID	Name	City
1	110	Ali	Ahvaz
2	140	Shima	Bam
3	180	Reza	Bam
4	10	Reza	Bam
5	65	Sara	Bam
6	260	Eli	Ilam
7	80	Neda	Kerman
8	130	Nima	Minab
9	70	Nazi	Qom
10	20	Ali	Qom
11	35	Hadi	Sari
12	100	Mani	Sari
13	190	Amir	Sari
14	40	Ala	Shiraz
15	50	Ali	Tehran
16	240	Laleh	Theran
17	160	Neda	Yazd
18	120	Diba	Yazd
19	90	Dara	Zahed...

فرض کنید جدول فوق به شکل یک آرایه خطی در هارد کامپیوتر ذخیره شده است.

سوال) اجرای هر کدام از پرس‌وجوهای زیر چقدر طول می‌کشد؟

```
CREATE INDEX NId ON Customer(NationalID)
```




```
SELECT * FROM Customer WHERE NationalID = 40
SELECT * FROM Customer WHERE NationalID = 65
SELECT * FROM Customer WHERE NationalID = 120
```

با index زدن روی NationalID، جدول به شکل یک درخت باینریB-Tree روی هارد ذخیره می‌شود. این ترفند زمان اجرای بازیابی بر اساس NationalID را به شکل لگاریتمی کاهش می‌دهد.

- وقتی زمان لگاریتمی کاهش می‌یابد اصطلاحا گفته‌می‌شود که سرعت نمایی زیاد می‌شود.

سوال) سرعت اجرای پرس‌وجوهای زیر چقدر است؟

```
SELECT * FROM Customer WHERE Name = ‘Reza’
```

- سرعت بازیابی اطلاعات براساس ستون ایندکس نمایی زیاد می‌شود. زمان بازیابی برای دیگر ستونها همچنان خطی است.

سوال) دستور زیر چه کار می‌کند؟

```
CREATE INDEX NameX ON Customer(Name)
```

این دستور روی Name ایندکس می‌زند. یعنی در کنار ساختار درختی قبلی بر اساس Name هم یک ساختار درختی ایجاد می‌کند. پرس‌وجوها بر اساس Name هم لگاریتمی می‌شود.

سوال) چرا روی همه ستونها ایندکس نزنیم؟

- ایندکس زدن با اینکه زمان بازیابی را کم می‌کند ولی هر ایندکس ساختار درختی خودش را دارد. به خاطر این هزینه‌ی ذخیره‌سازی، معمولا فقط روی کلید اصلی ایندکس می‌زنند.

سوال) دستور زیر چه کار می‌کند؟

```
CREATE UNIQUE INDEX CityX ON Customer(City)
```

پاسخ) این دستور می‌خواهد روی City ایندکس یکتا بزند. با توجه به اینکه در City مقادیر تکراری داریم، این تولید خطا می‌کند. اگر City مقادیر تکراری نداشته‌باشد، این دستور ایندکس یکتا می‌زند. و بعد از این دستور دیگر نمی‌توان مقادیر تکراری برای ستون City در جدول درج کرد.

سوال) دستور زیر چه کار می‌کند؟

```
DROP INDEX NId
```

پاسخ) ایندکس NId از جدول حذف می‌شود.

تعریف محدودیتهای جامعیتی در SQL

دیدیم که به هنگام تعریف جدولهای پایگاه‌داده می‌توانیم محدودیتهایی را اعمال کنیم.

```
CREATE TABLE course(
cname CHAR(20),
tname CHAR(20),
PRIMARY KEY (cname)
)
```

	cname	tname
1	AI	Razavi
2	AW	Khani
3	DB	Khani
4	GS	Khani

```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meet int,
hour float,
attno int,
PRIMARY KEY (attno),
FOREIGN KEY (cname) REFERENCES course ON UPDATE CASCADE ,
FOREIGN KEY (sname) REFERENCES student ON UPDATE CASCADE
)
```

مثال) وضعیت اجرای این دستورات در پایگاه داده بالا به چه شکل خواهد بود؟

```
INSERT INTO att (sname, cname, meet, hour, attno) VALUES  ('ALI','AA', 1, 2.2, 5050);

INSERT INTO course (cname, tname) VALUES  ('GS','Razavi');
```

گاهی نیاز داریم علاوه بر محدودیت‌های فوق محدودیتهای پیچیده‌تری به پایگاه‌داده اضافه کنیم. با استفاده از create assertion می‌توان شرط‌هایی را تعریف کرد. بعد از تعریف این شرط‌ها DBMS موظف به رعایت این شرط‌ها به هنگام درج یا بروزرسانی جدول‌هاست.

مثال) با اجرای دستور زیر دیگر نمی‌توان درسی را در جدول course وارد کرد که نام استاد نداشته باشد.

```
CREATE ASSERTION teacher_notnull CHECK( NOT EXISTS
    ( SELECT * FROM course
    WHERE  tname is NULL));
```

البته قدرت اصلی create assertion در مثالهای بعدی مشخص می‌شود. در مثال قبلی به راحتی به هنگام ایجاد جدول می‌توان محدودیت را تعریف کرد.

```
CREATE TABLE course(
cname CHAR(20),
tname CHAR(20) NOT NULL,
PRIMARY KEY (cname)
)
```

مثال) می‌خواهیم اجازه ندهیم که یک شعبه بانک بیشتر از سپرده‌هایش وام بدهد.

	cname	bname	amount	lno
1	Sara	Kaj	15000	1745
2	Ali	Kaj	7500	3435
3	Ali	Afra	7500	3945

	cname	bname	balance	ano
1	Ala	Sarv	7530	1020
2	Negin	Kaj	3700	1045
3	Amir	Kaj	7800	1198
4	Ali	Kaj	1700	1734
5	Reza	Kaj	2500	2333
6	Ali	Kaj	7500	3945
7	Ali	Afra	1050	4101
8	Mitra	Afra	5200	4172
9	Nazi	Kaj	7500	4243
10	Sara	Sarv	8500	5151
11	Reza	Kaj	5300	8123

```
SELECT SUM(amount) FROM Loan
WHERE Loan.bname = 'Kaj'

SELECT SUM(amount) FROM Loan
WHERE Loan.bname = 'Afra'

SELECT SUM(balance) FROM Deposit
WHERE Deposit.bname = 'Kaj'

SELECT SUM(balance) FROM Deposit
WHERE Deposit.bname = 'Afra'
```

```
CREATE ASSERTION sum-constraint CHECK
     (NOT EXISTS (SELECT * FROM branch
                        WHERE (SELECT SUM(amount) FROM loan
                                 WHERE loan.bname = branch.bname)
                               >= (SELECT SUM(balance) FROM   Deposit
                                 WHERE loan.bname = branch.bname)))
```

نکته) خیلی از DBMS های اصلی بازار مثل MS SQL Server، PostgreSQL create assertion را پیاده‌سازی نکرده‌اند.

تعریف محدودیتهای امنیتی

انواع مجوزها در SQL

- مجوزهای جدول Object Privileges
  - خواندن اطلاعات Select
  - درج Insert
  - بروزرسانی Update
  - حذف Delete
- مجوزهای پایگاه داده System Privileges
  - ایجاد جدول Create

- حذف جدول Drop
- بروزرسانی جدول Alter

اعطای مجوز Grant

```
GRANT SELECT, UPDATE ON employees TO user2;
GRANT SELECT(sname, cname, hour), update (hour) ON att TO Ali, Sina
GRANT SELECT(sname, cname, hour) ON Buy TO PUBLIC
GRANT CREATE TABLE TO Hadi;
```

بازپس گیری مجوز

```
REVOKE SELECT, UPDATE ON employees FROM user2
REVOKE SELECT(sname, cname, hour) ON Buy FROM Ali
REVOKE DROP TABLE FROM reza
```

نکته) اگر کاربری جدولی ایجاد کند 4 مجوز آن جدول را خواهد داشت.

نکته) در SQL اعطای مجوز زنجیره‌ای امکان‌پذیر بوده ولی به خاطر دردسرهای آن توصیه نمی‌شود. مجوز زنجیره‌ای یعنی مدیر به Ali مجوز بدهد و بعد Ali این مجوز را به نفر بعدی مثلا Hadiبدهد.

1 طراحی مفهومی به روش ER

1.1 نمودار ER Entity-Relation Diagram

این نمودار، پایگاه داده را نمایش می‌دهد. مثل فلوچارت که ساختار یک الگوریتم را نمایش می‌دهد.

این نمودار، در طراحی پایگاه‌داده استفاده می‌شود. مثل فلوچارت که در طراحی الگوریتم و برنامه‌سازی نمایش می‌شود.

این نمودار، از دو جز اصلی Entity موجودیت و Relation رابطه تشکیل شده است.

1.2 موجودیت Entity

1.2.1 به زبان ساده

به هر جدول پایگاه‌داده موجودیت Entity می‌گوییم.

به هر ستون جدول یک خصیصه Attribute می‌گویند.

به هر سطر جدول یک نمونه موجودیت Entity Instance گفته می‌شود.

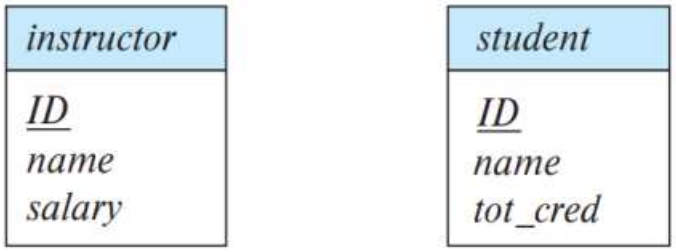


Figure 6.1 E-R diagram showing entity sets *instructor* and *student*.

Instructor		
ID	name	salary
76766	Crick	7500
45565	Katz	6000
98345	Kim	7000

Student		
ID	name	Tot_cred
00128	Zhang	12.23
76653	Aoi	14.5
76543	Brown	15
023121	Chavez	17.2

1.2.2 تعریف عام موجودیت

هر آنچه در دنیای واقعی وجود دارد (موجود است)

1.2.2.1 موجودیت محسوس (فیزیکی)

مثل دانشجو، شعبه بانک، کتاب..

1.2.2.2 موجودیت نامحسوس (منطقی)(قراردادی)

مثل خرید، حضور در کلاس، وام، مسابقه فوتبال

1.2.3 تعریف خاص موجودیت

هر آنچه پایگاه‌داده (بانک‌اطلاعاتی) موظف است تا اطلاعاتی درباره آن ذخیره کند. موجودیت نامیده می‌شود.

نکته) تصمیم درباره تعریف موجودیت بر عهده طراح پایگاه‌داده است.

1.2.3.1 نمونه موجودیت Entity instance

هر موجودیت دارای نمونه‌های متعددی است که به آنها نمونه موجودیت Entity Instance گفته می‌شود.

به زبان ساده به هر سطر جدول پایگاه داده نمونه موجودیت گفته می‌شود.

نکته) در بعضی از منابع به موجودیت مجموعه موجودیت Entity Set گفته می‌شود.

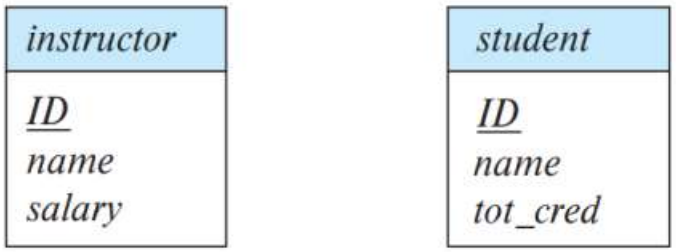


Figure 6.1 E-R diagram showing entity sets *instructor* and *student*.

Instructor		
ID	name	salary
76766	Crick	7500
45565	Katz	6000
98345	Kim	7000

Student		
ID	name	Tot_cred
00128	Zhang	12.23
76653	Aoi	14.5
76543	Brown	15
023121	Chavez	17.2

1.2.4 انواع خصیصه‌ها

به زبان ساده به هر ستون جدول پایگاه داده، خصیصه Attribute های آن موجودیت گفته می‌شود.

نکته) تصمیم گیری تصمیم‌گیری درباره خصیصه‌های موجودیت یا ستون‌های جدول با طراح پایگاه‌داده است.

مثال) مثلا ذخیره اطلاعات قد یا وزن یک دانشجو ضرورتی ندارد. ولی برای موجودیت یک ورزشکار ذخیره سن، قد و وزن مهم است.

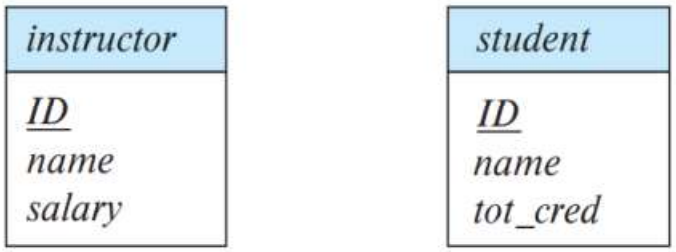


Figure 6.1 E-R diagram showing entity sets *instructor* and *student*.

Instructor		
ID	name	salary

Student		
ID	name	Tot_cred

1.2.4.1 انواع خصیصه‌ها از منظر ساختاری

1.2.4.1.1 خصیصه ساده simple

قابل تجزیه به اجزای کوچکتر نیستند. مثل نام، نام خانوادگی، سال تولد

1.2.4.1.2 خصیصه مرکب composite

قابل تجزیه به اجزای کوچکتر هستند. مثل آدرس که از سه مولفه شهر، خیابان و پلاک تشکیل شده‌است.

نکته) خصیصه مرکب در پایگاه‌داده رابطه‌ای پیاده‌سازی نمی‌شود.

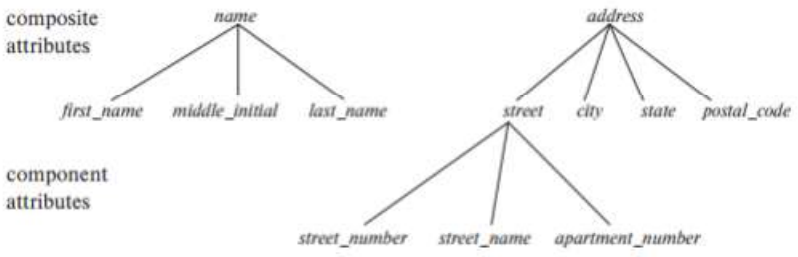


Figure 6.7 Composite attributes instructor *name* and *address*.

1.2.4.2 انواع خصیصه‌ها از دیدگاه مقدار

1.2.4.2.1 خصیصه تک مقداری single-value

در هر زمان فقط یک مقدار می‌تواند داشته باشد. مثل نام‌خانوادگی ، نام یا سال تولد

1.2.4.2.2 خصیصه چند مقداری multivalued

در یک زمان می‌تواند چند مقدار داشته باشد. مثل شماره تلفن

1.2.4.3 انواع خصیصه‌ها از دیدگاه وابستگی

1.2.4.3.1 خصیصه پایه

خصیصه پایه به هیچ خصیصه‌ای وابستگی ندارند. مثل نام، نام خانوادگی، تاریخ تولد

1.2.4.3.2 خصیصه مشتق derived

خصیصه مشتق به دیگر خصیصه‌ها وابستگی داشته و بر اساس آنها قابل محاسبه است. مثل سن دانشجو که بر اساس تاریخ تولد قابل محاسبه است یا معدل دانشجو که بر اساس واحد و نمرات دانشجو قابل محاسبه است.

نکته) طراح پایگاه‌داده درباره ذخیره‌سازی یا عدم‌ذخیره‌سازی خصیصه مشتق تصمیم می‌گیرد.

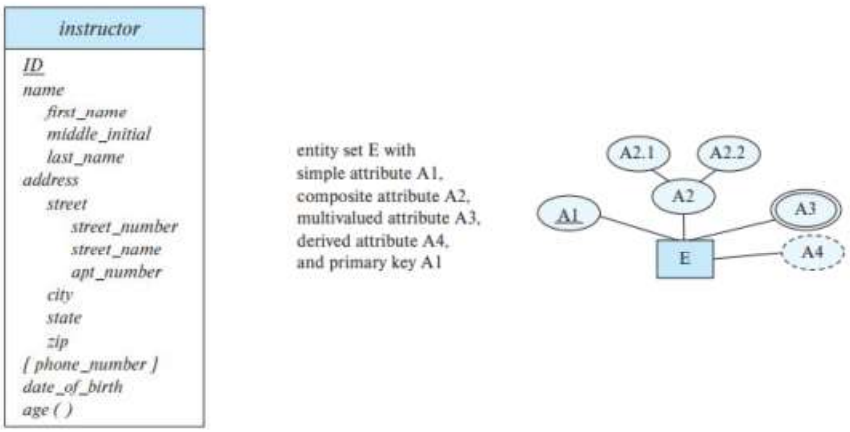


Figure 6.8 E-R diagram with composite, multivalued, and derived attributes.

سوال) مزایا و معایب ذخیره‌سازی خصیصه مشتق چیست؟

1.2.5 انواع موجودیت‌ها

1.2.5.1 موجودیت معمولی

موجودیتی که استقلال وجودی دارد، یعنی حضور و وجودش وابسته به هیچ موجودیت دیگر نیست

1.2.5.2 موجودیت ضعیف Weak

موجودیتی که استقلال وجودی ندارد، یعنی حضور و وجودش وابسته به موجودیت‌های دیگر هست.

Figure 6.14 E-R diagram with a weak entity set.

1.2.6 نشانه‌های موجودیت

طراح به هنگام شناسایی موجودیت‌ها (جدول‌ها) باید به دنبال نشانه‌های زیر باشد.

- تعریف خاص: برای یک موجودیت باید اطلاعات ذخیره شده باشد.
- چند نمونه‌گی: چند نمونه از موجودیت باید قابل شناسایی باشد. به ندرت (1% مواقع) ممکن است موجودیت تک نمونه‌ای داشته باشیم.
- چندخصیصه‌ای: هر موجودیت باید حداقل چند خصیصه داشته‌باشد. به ندرت (1% مواقع) ممکن است موجودیت تک خصیصه‌ای داشته باشیم.
- کلید اصلی: هر موجودیت باید کلید اصلی داشته باشد.

1.3 رابطه Relationship

مثال) رابطه استاد راهنمایی

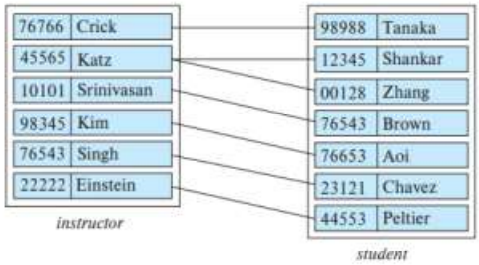


Figure 6.2 Relationship set *advisor* (only some attributes of *instructor* and *student* are shown).

Figure 6.3 E-R diagram showing relationship set *advisor*.

1.3.1 درجه رابطه Degree

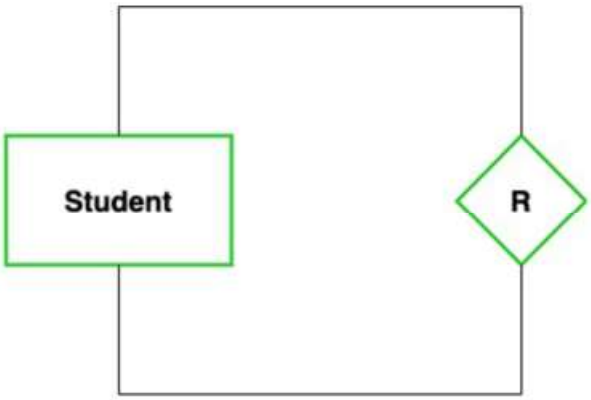
تعریف) درجه رابطه مشخص می‌کند چند نوع موجودیت از طریق رابطه با هم رابطه دارند.

درجه 2 (degree 2) binary

Figure 6.3 E-R diagram showing relationship set *advisor*.

درجه 1 (degree 1) unary

مثال) در یک کلاس دانشجویان مختلفی داریم. هر کلاس تعدادی نماینده(ارشد) که به کار دانشجویان نظارت می‌کنند. چون نمایندگان هم دانشجو هستند. رابط R نمایندگی بین نمونه‌های مختلف موجودیت دانشجو ارتباط برقرار می‌کند. به این رابطه درجه 1 یا unary گفته می‌شود.



سوال) یک نمونه دیگر از رابطه درجه 1 مثال بزنید؟

درجه 3 (degree 3) ternary

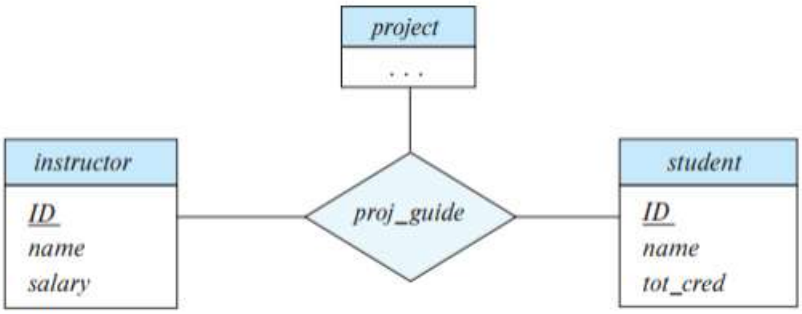
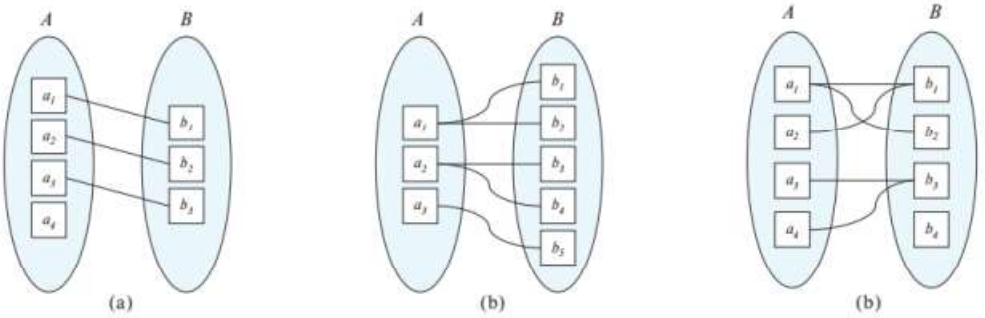
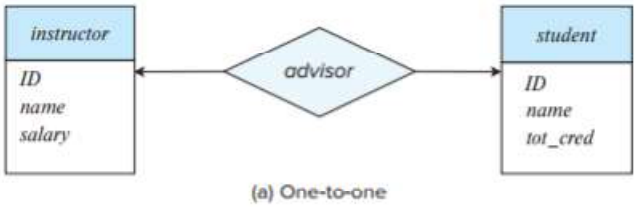


Figure 6.6 E-R diagram with a ternary relationship *proj_guide*.

1.3.2 کاردینالتی رابطه Cardinality

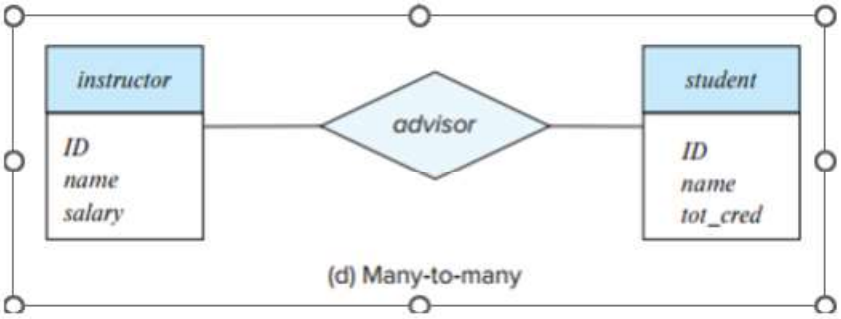




(a) One-to-one

(b) One-to-many

(d) Many-to-many



هر دانشجو می تواند چند استاد داشته باشد؟ صفر تا n

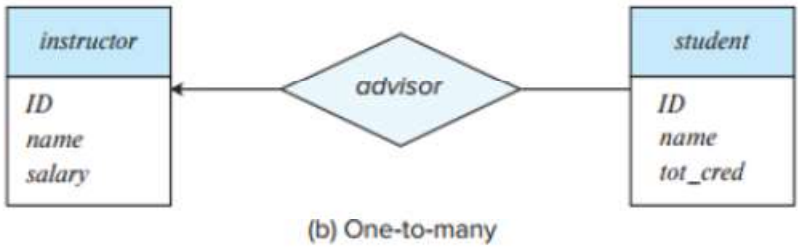
آیا استاد بی دانشجو می‌توانیم داشته باشیم؟ بله

آیا یک دانشجو می‌تواند دو تا استاد راهنما داشته باشد؟ بله

آیا یک دانشجو می‌تواند 100 استاد داشته‌باشد؟ بلی

1.3.2.1 رابطه یک به چند و نقرین جهت

در رابطه یک به چند جهت خیلی مهم است. در رابطه زیر یک استاد می تواند چند دانشجو داشته باشد. ولی یک دانشجو فقط می‌تواند یک استاد داشته باشد. اصطلاحا گفته می‌شود رابطه یک‌به‌چند داریم از استاد به دانشجو. یک استاد چند دانشجو می تواند داشته باشد. دقت کنید در نوتیشن جهت فلش از دانشجو به استاد است ولی در نوشتن می گوییم یک به چند از استاد به دانشجو



هر دانشجو می تواند چند استاد داشته باشد؟ حداکثر 1

آیا استاد بی دانشجو می‌توانیم داشته باشیم؟ بله

آیا یک دانشجو می‌تواند دو تا استاد راهنما داشته باشد؟ خیر

آیا یک استاد می‌تواند 100 دانشجو داشته‌باشد؟ بلی

	کاردینالتی		
	یک به یک	یک به چند	چند به چند

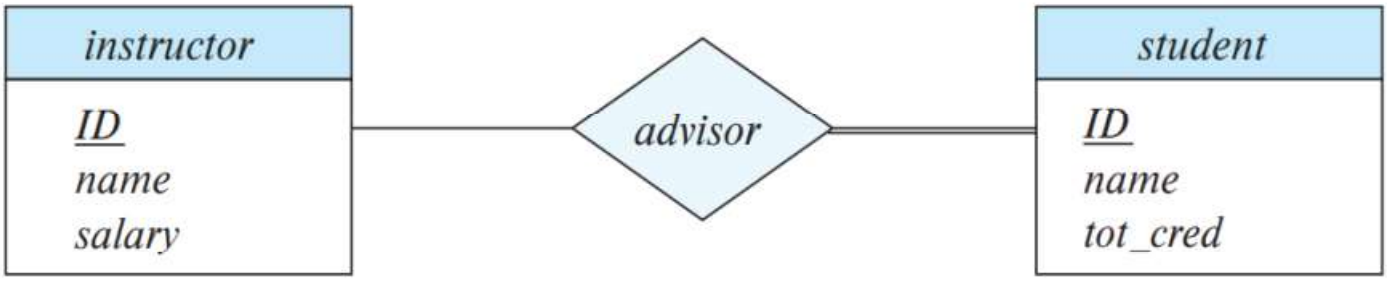
درجه 1			
درجه 2	رایج		
درجه 3			

نکته) در بین انواع رابطه با درجه و کاردینالیتی های مختلف 95% رابطه‌ها از درجه 2 با کاردینالیتی یک به چند هستند.

1.3.2.2 محدویت‌ها در کاردینالیتی

اجباری: وقتی که همه نمونه موجودیت‌ها باید در رابطه حضور داشته باشد.

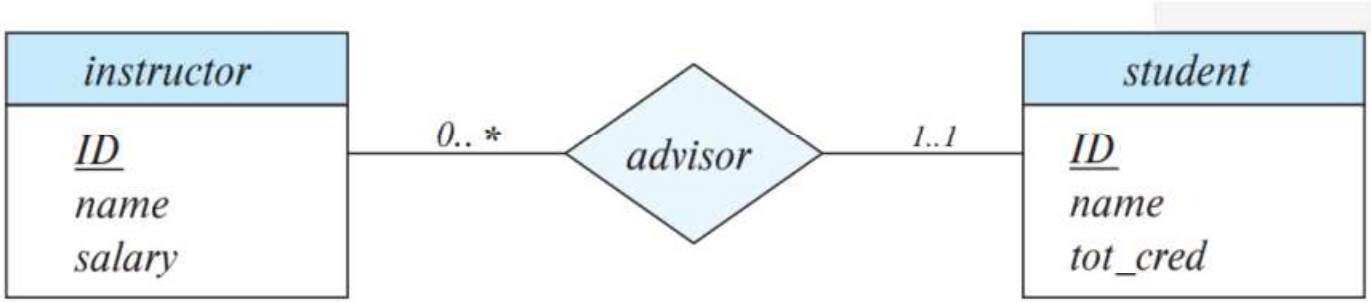
اختیاری: وقتی که یک نمونه موجودیت می‌تواند در رابطه حضور نداشته‌باشد.



در رابطه بالا حضور دانشجوها در رابطه advisor اجباری است. پس یک دانشجو حتما باید استاد راهنما داشته‌باشد. یا به عبارت دیگر دانشجوی بدون استاد راهنما نمی‌توانیم داشته‌باشیم.

برعکس حضور استاد در رابطه advisor اختیاری است. یعنی می‌توان استادی را پیدا کرد که هیچ دانشجویی نداشته‌باشد.

در نمودار ER راههای بیشتر و دقیقتری هم برای نمایش کاردینالیتی وجود دارد.

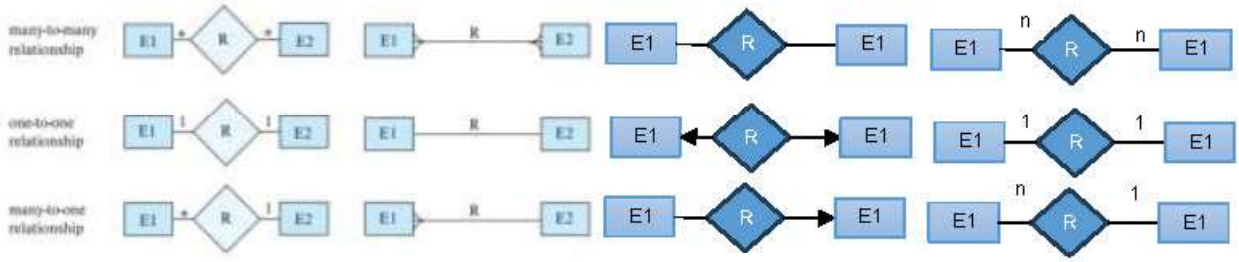


در نمودار بالا یک دانشجو حداقل 1 و حداکثر 1 بار می تواند در رابطه باشد. یک استاد می‌تواند حداقل 0 و تا بی نهایت در رابطه باشد. پس این رابطه یک به چند از استاد به دانشجو است که برای دانشجو رابطه اجباری است.

نکته) اگر هر دو طرف بیشینه 1 داشتند رابطه یک به یک بود.

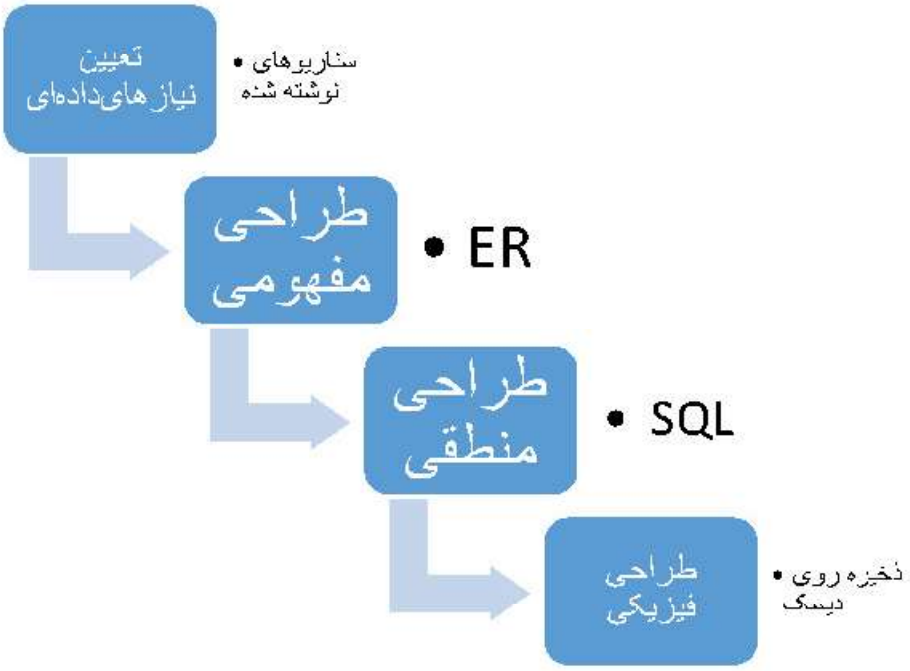
نکته) اگر در سمت استاد *..1 داشتیم. رابطه برای استاد هم اجباری می‌شد و هر استاد باید حتما دانشجویی را راهنمایی می‌کرد.

نکته) در کتابها و منابع مختلف نوتیشن‌های مختلفی برای رابطه استفاده شده است.



2 مراحل طراحی پایگاه‌داده

- طراحی پایگاه‌داده بخش مهمی از طراحی یک سامانه نرم‌افزاری است.
- نکته) برای کنکور زیاد درگیر طراحی فیزیکی و تعیین نیازمندی‌ها نمی‌شویم.



2.1 تعیین نیازهای داده‌ای

این بخشی از مهندسی نرم‌افزار است.

2.2 طراحی مفهومی

طراح پایگاه‌داده نیازمندی‌های متنی را مدل می‌کند.

رایج‌ترین روش برای مدل‌سازی نمایش گرافیکی با نمودار ER می‌باشد.

طراح در این مرحله باید بررسی کند آیا مدل طراحی شده نیازمندی‌های پرس‌وجویی سامانه نرم‌افزاری را پاسخگو هست؟

2.3 طراحی منطقی

طراحی مفهومی به یکی از مدل‌های پایگاه داده تبدیل می‌شود.

رایج‌ترین مدل پایگاه‌داده رابطه‌ای بود. پس موجودیت به جدول، خصوصیت به ستون جدول و رابطه به کلید خارجی تبدیل (ترجمه) می‌شود.

مرحله طراحی	گام اول	گام دوم	نمایش

طراحی مفهومی	شناسایی موجودیت‌ها	شناسایی خصیصه‌ها	درک روابط بین آنها	ER
طراحی منطقی	شناسایی جدول‌ها	شناسایی ستون‌ها	تعریف کلید خارجی برای رابطه بین جدول	SQL
	رابطه	خصیصه		جبر رابطه‌ای

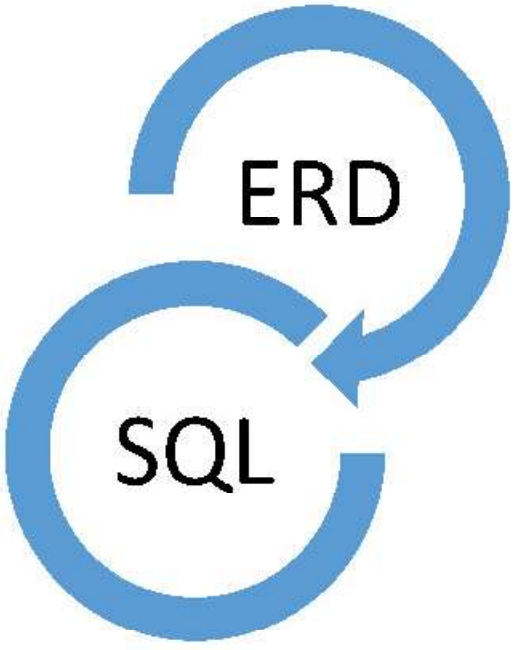
نکته: دقت کنید که Relation رابطه در جبر رابطه‌ای Relational Algebra با Relationship رابطه در طراحی مفهومی یکی نیست. در انگلیسی واژه‌های متمایز داریم ولی در فارسی گاهی این دو مفهوم با هم به اشتباه گرفته می‌شوند.

2.4 طراحی فیزیکی

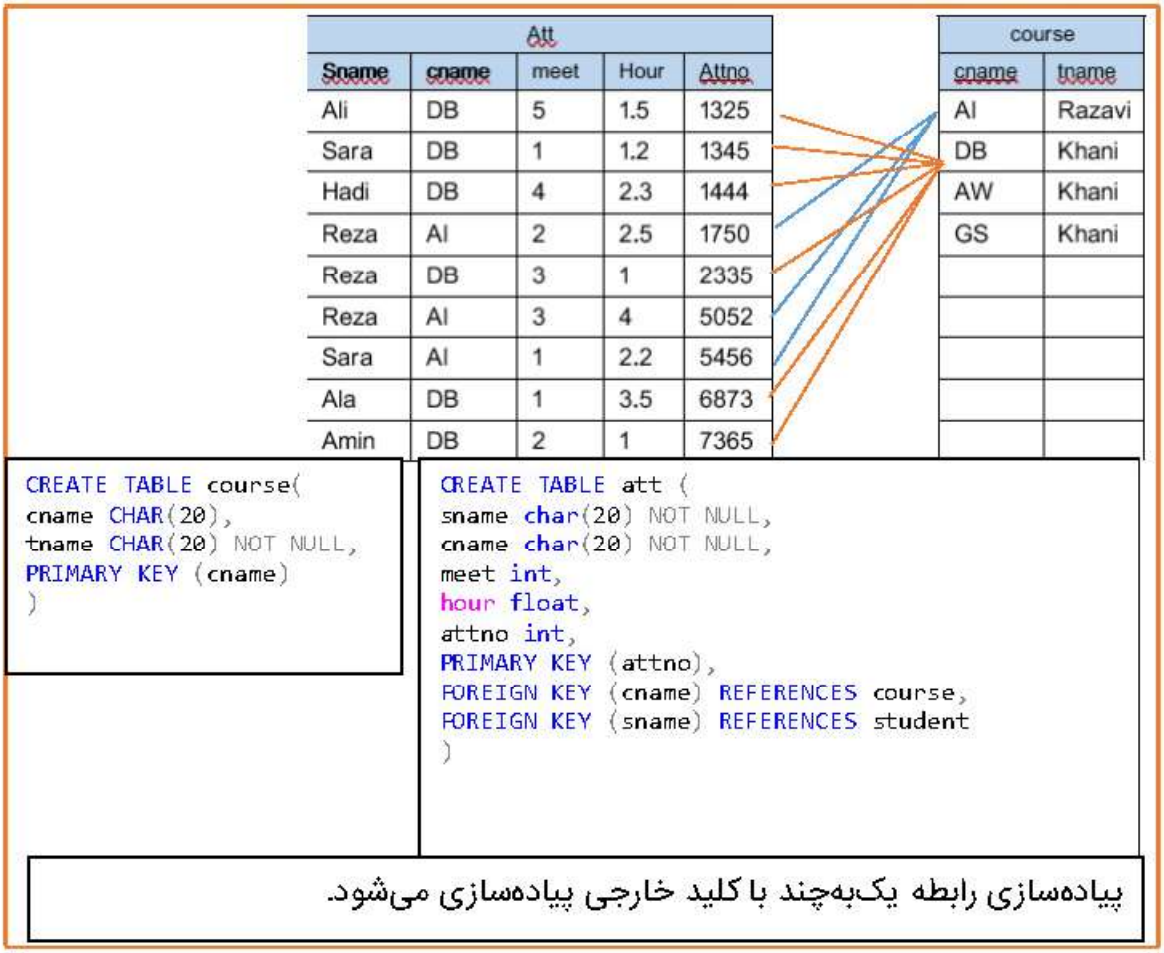
درباره جزییات پیاده‌سازی پایگاه‌داده روی دیسک تصمیم‌گیری می‌شود.

3 طراحی منطقی (پیاده‌سازی)

- طراحی منطقی به زبان ساده: تبدیل (پیاده‌سازی) ERD به (با) SQL



3.1 پیاده‌سازی رابطه یک‌به‌چند



Att

Sname	cname	meet	Hour	Attno
Ali	DB	5	1.5	1325
Sara	DB	1	1.2	1345
Hadi	DB	4	2.3	1444
Reza	AI	2	2.5	1750
Reza	DB	3	1	2335
Reza	AI	3	4	5052
Sara	AI	1	2.2	5456
Ala	DB	1	3.5	6873
Amin	DB	2	1	7365

course

cname	tname
AI	Razavi
DB	Khani
AW	Khani
GS	Khani

```
CREATE TABLE course(
cname CHAR(20),
tname CHAR(20) NOT NULL,
PRIMARY KEY (cname)
)
```

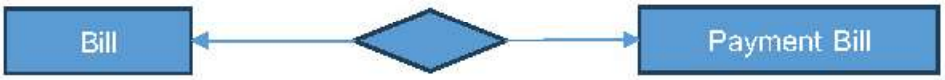
```
CREATE TABLE att (
sname char(20) NOT NULL,
cname char(20) NOT NULL,
meet int,
hour float,
attno int,
PRIMARY KEY (attno),
FOREIGN KEY (cname) REFERENCES course,
FOREIGN KEY (sname) REFERENCES student
)
```

پیاده‌سازی رابطه یک‌به‌چند با کلید خارجی پیاده‌سازی می‌شود.

3.2 پیاده‌سازی رابطه یک‌به‌یک

هر برگه جریمه Bill یک فیش پرداخت Payment Bill دارد. و هر فیش پرداخت دو تا Payment Bill ندارد.

هیچکس یک برگه جریمه را دوبار پرداخت نمی‌کند. هیچ فیشی دو تا جریمه را تسویه نمی‌کند.



Bill No	Amount	Type	PB No
29	100	Parking	12
45	70	Belt	
33	55	Speed	76

PB No	Date	Bill No
76	2 oct	33
12	10 nov	29

دو موجودیتی که با هم رابطه یک‌به‌یک دارند باید با هم ادغام شوند.

برخی از منابع باید ادغام از جنس should (توصیه) است. در برخی از منابع از جنس must (الزام) است.

Bill No	Amount	Type	PB No	Date
29	100	Parking	12	12 nov
45	70	Belt	NULL	NULL
33	55	Speed	76	2 oct

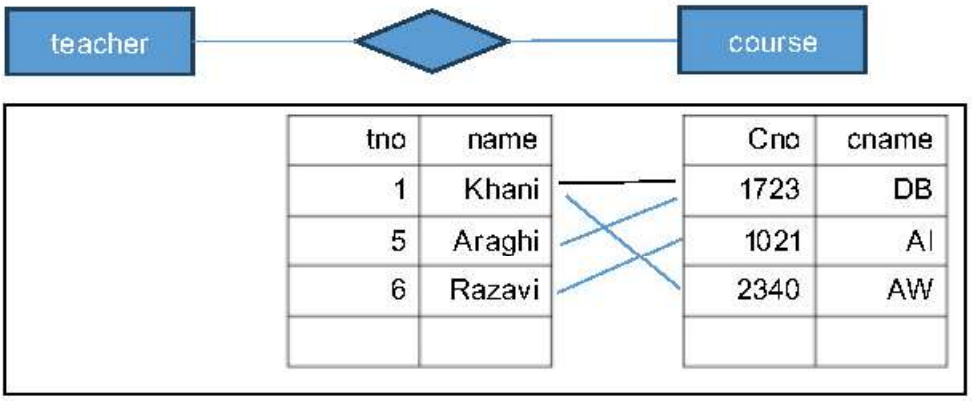
نکته) ایرادی که در ادغام وجود دارد این است که تعداد زیادی null خواهیم داشت. داشتن تعداد زیادی null مطلوب نیست. به همین خاطر بعضی از طراح‌ها ادغام می‌کنند.

3.3 پیاده‌سازی رابطه چندبه‌چند

رابطه چندبه‌چند در مدل رابطه‌ای قابل پیاده‌سازی نیست. باید به یکی از دو روش زیر رابطه چندبه‌چند را به دو رابطه یک‌به‌چند تبدیل می‌کنیم.

1. تحلیل دقیق‌تر و پیداکردن موجودیت از قلم افتاده
2. جعل (ساخت) موجودیت (کاذب) جدید

سوال) در دیاگرام زیر چه موجودیتی از از قلم افتاده است؟



پاسخ) موجودی گروه، دکتر خانی و دکتر عراقی با اینکه هر دو درس DB را می‌دهند. منتها در دو گروه یا اصطلاحا کد مجزا. مثلا دکتر خانی سه‌شنبه‌ها درس می‌دهد. دکتر عراقی جمعه‌ها. با اضافه کردن Group به رابطه به جای یک رابطه چندبه‌چند دو رابطه یک‌به‌چند داریم که به راحتی با استفاده از کلید خارجی قابل پیاده‌سازی است.

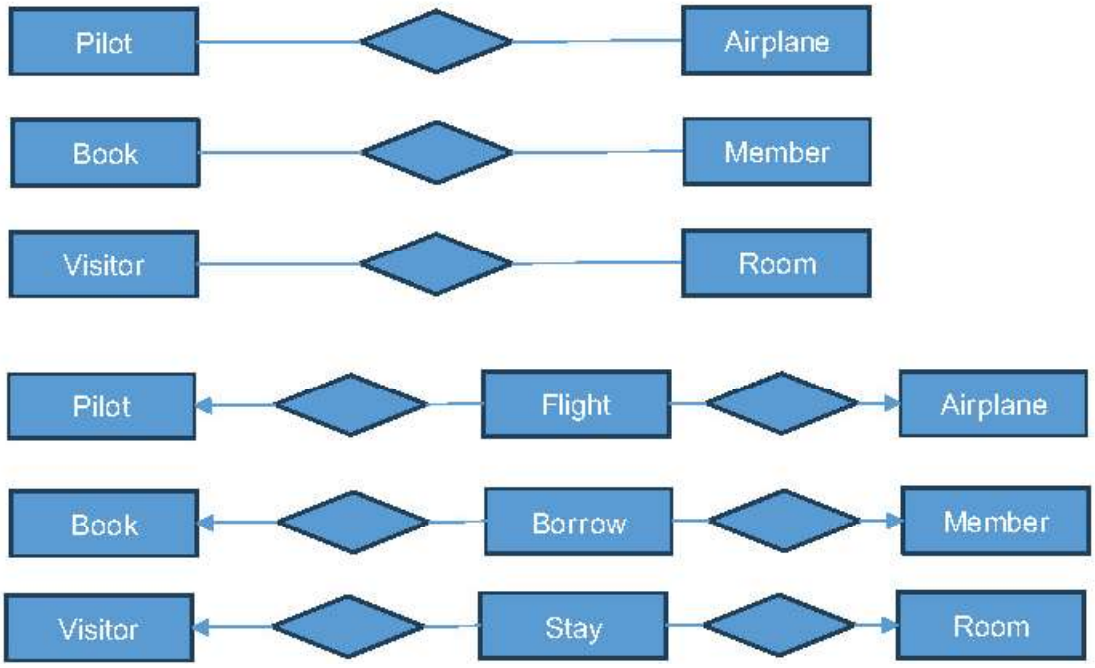
tno	name
1	Khani
5	Araghi
6	Razavi

name	Gno	*cno*	*tno*
1	23	1723	1
2	45	1723	5
1	70	2340	1
1	17	1021	6

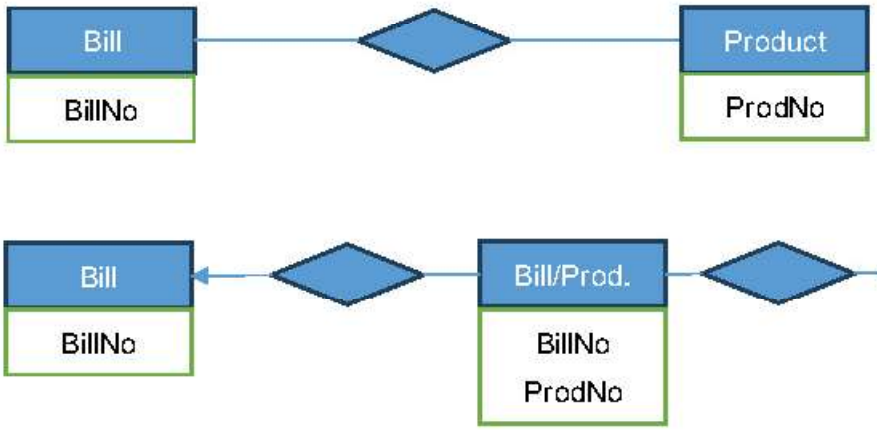
Cno	cname
1723	DB
1021	AI
2340	AW

نکته) در این مثال این تعبیر که رابطه چندبه‌چند را به دو رابطه یک‌به‌چند شکستیم درست نیست. زیرا این رابطه چندبه‌چند اصلا وجود ندارد. با تحلیل دقیق‌تر دیدیم که یک موجودیت Group از قلم افتاده و teacher و course به واسطه این موجودیت از قلم افتاده در ارتباط هستند.

سوال) مثالهای دیگری از موجودیت‌های از قلم افتاده بزنید.



مثال) هر Product کالا می‌تواند در چند Bill فاکتور باشد. هر فاکتورBill می‌تواند شامل چند کالا باشد.



نکته) در اینجا موجودیت .Bill/Prod جعلی یا ساختگی است. یک همچین موجودی از تحلیل به دست نمی‌آید.

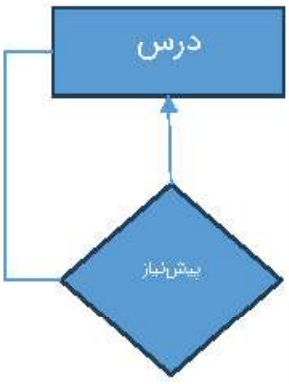
نکته) ترکیب دو کلید خارجی BillNo و ProdNo یکتاست و می‌تواند به عنوان کلید اصلی استفاده شود.

نکته) در مثالهایی که یک موجودیت جعل شده است. تعبیر شکست یک رابطه چندبه‌چند به دو رابطه یک‌به‌چند مناسب‌تر به نظر می‌رسد.

3.4 پیاده‌سازی رابطه درجه یک

پیاده‌سازی رابطه درجه 1 در مدل رابطه‌ای امکان‌پذیر است.

به خاطر ابهام زیاد استفاده نمی‌شود.

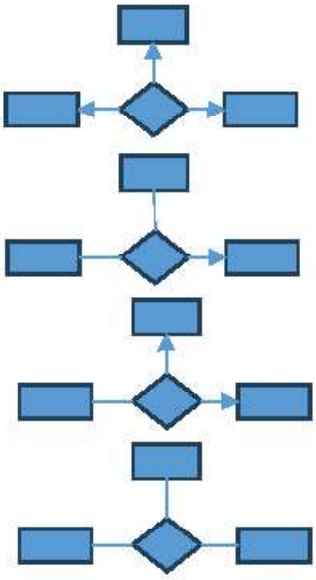


تفسیر1: هر درس چند پیش‌نیاز دارد اما خودش فقط پیش‌نیاز یک درس بوده‌است.

تفسیر2: هر درس فقط یک پیش‌نیاز دارد، ولی خودش می‌تواند پیش‌نیاز چند درس باشد.

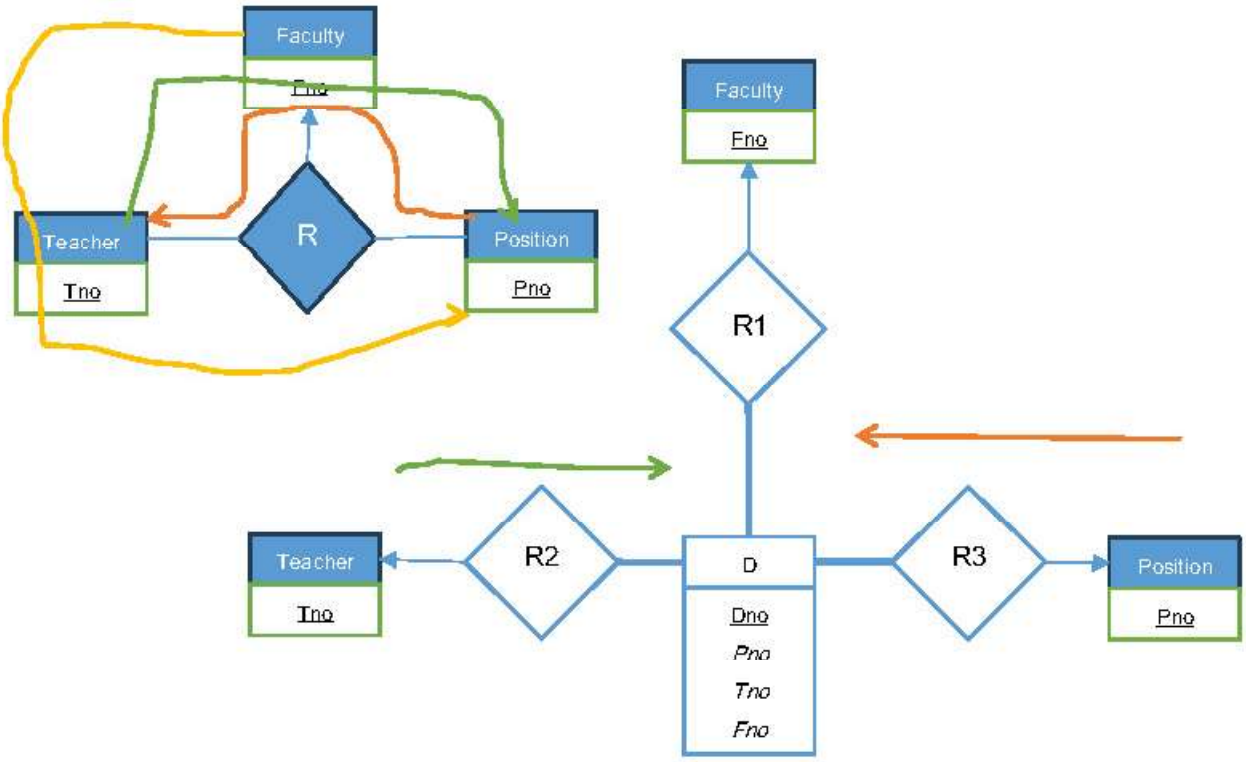
```
CREATE TABLE lesson(
              lno INT,
              pno INT,
              PRIMARY KEY (lno),
  FOREIGN KEY (pno) REFERENCES lesson )
```

3.5 پیاده‌سازی درجه سه



1:1:1 واضح، مشابه رابطه درجه 2، 1:1 امکان ادغام سه موجودیت وجود دارد.
N:n:1 واضح، قابل تبدیل، بعد از تبدیل قابل پیاده‌سازی
N:1:1 مبهم، بعد از تفسیر قابل تبدیل، بعد از تبدیل قابل پیاده‌سازی
N:N:N مبهم

N:N:1 3.5.1



مسیر1) هر استاد در یک دانشکده ممکن است چند تا پست داشته باشد. مثلا خانی می‌تواند هیات علمی باشد، عضو شورای پژوهشی باشد و معاون دانشجویی

مسیر2) هر پست در یک دانشکده ممکن است توسط چند تا استاد اشغال شده باشد. مثلا پست هیات علمی دانشکده کامپیوتر توسط خانی، رضوی، ده‌یادگاری اشغال می‌شود.

مسیر3) هر دانشکده چند تا پست دارد و چند استاد. مثلا دانشکده کامپیوتر پست‌های هیالت علمی، عضو شورای آموزشی، عضو شورای پژوهشی، عضو کمیته انضباطی، معاون آموزشی .. دادر که توسط چند تا از اعضای هیات علمی اشغال شود.

اصلاح دیاگرام) موجودیت D در واقع از قلم افتاده‌است. ما در ساختار اداری دانشکده مفهومی به نام حکم (ابلاغ) داریم که در آن پست Position هر استاد ابلاغ می‌شود. و در نمودار اول در نظر گرفته نشده بود.

نکته) هر حکم (نمونه موجودیت D) باید یک و فقط یک (استاد/ دانشکده/ پست) داشته باشد.

حکم بدون نام دانشکده، پست، نام استاد نمی‌توانیم داشته‌باشیم.

حکم گروهی نمی‌توانیم داشته‌باشیم.

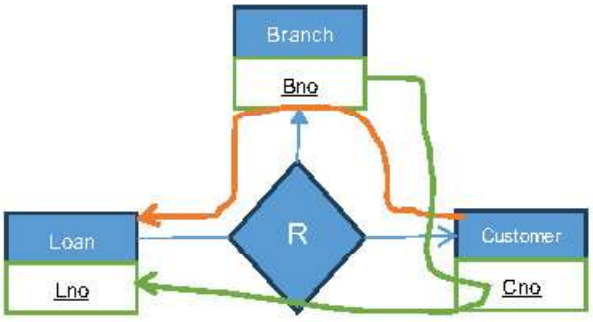
حکم چند پستی نمی‌تونیم داشته باشیم.

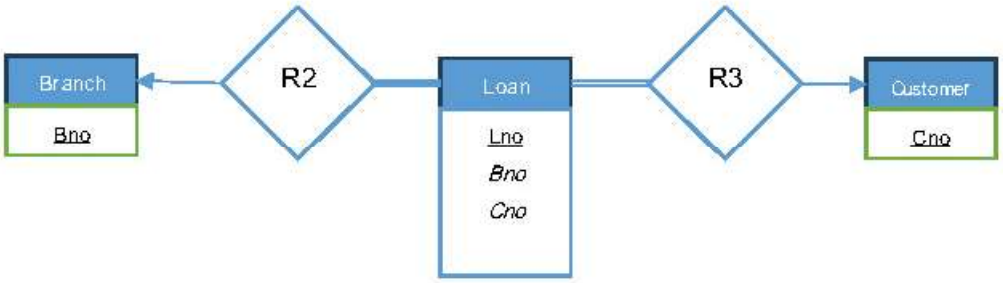
برای هر پست چند حکم صادر شده است.

برای هر استاد چند حکم صادر شده است.

برای هر دانشکده چند حکم صادر شده است.

N:1:1 3.5.2





این رابطه درجه 3 ابهام دارد.

تفسیر 1) هر مشتری فقط می‌تواند از یک شعبه چند وام بگیرد.

تفسیر2) هر شعبه می‌تواند فقط به یک مشتری چند وام بدهد.

تفسیر3) هر مشتری می‌تواند در هر شعبه چند وام بگیرد.

توصیه) در رابطه درجه سه فقط و فقط یک عدد 1 وجود داشته باشد.

نکته) برای رفع ابهام بهتر است طراح پایگاه‌داده تفسیر (منظور) خود را به رابطه درجه به وضوح نمایش دهد.

3.6 پیاده‌سازی موجودیت ضعیف

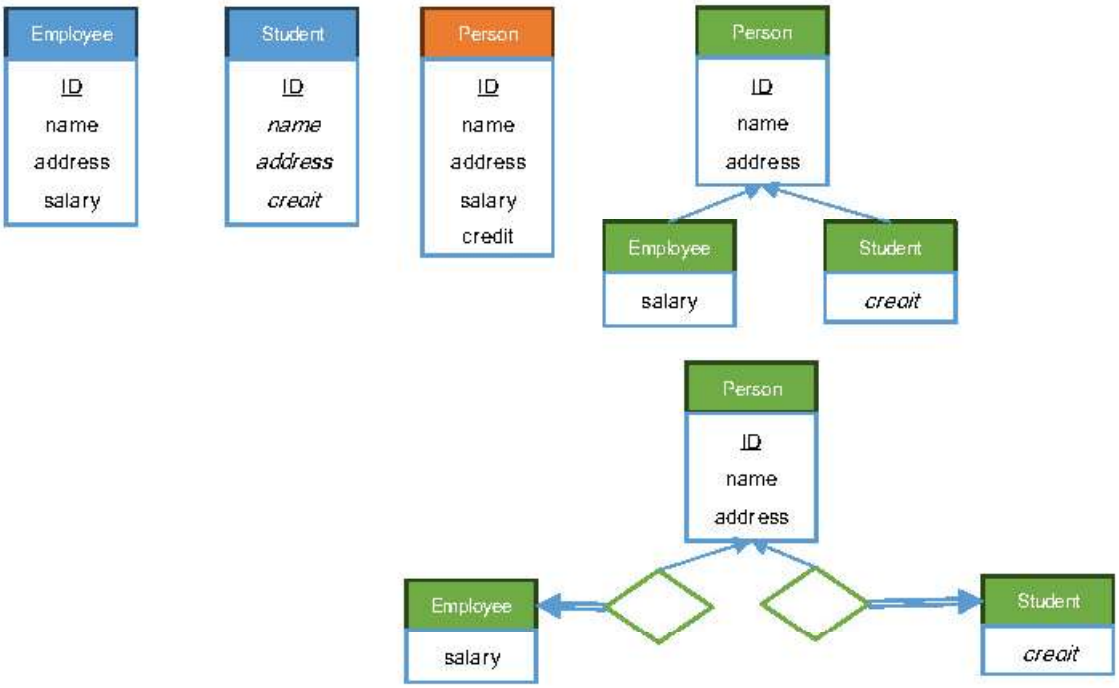
افراد تحت تکلف کارمند (مانند فرزندان و همسران) کاملا وابسته به کارمند هستند و هر تغییر کارمند باید روی آنها منعکس‌شود.

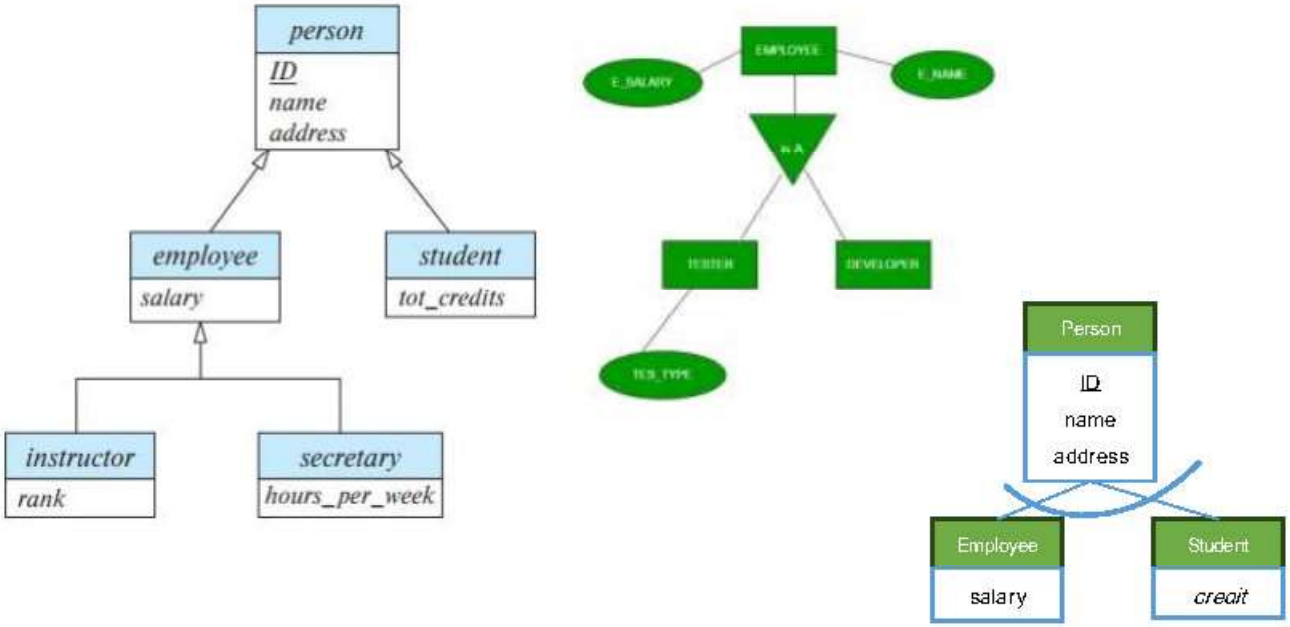

```
CREATE TABLE dependent(...
                FOREIGN KEY (eno) REFERENCES Employee
                ON DELETE CASCADE,
                  ON UPDATE CASCADE)
```

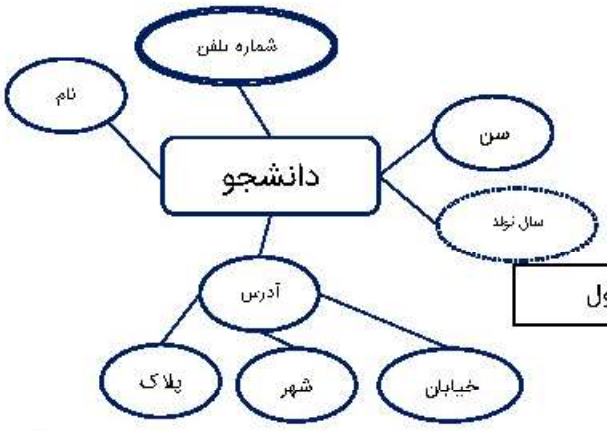
3.7 پیاده‌سازی رابطه کلی-اختصاصی Generalization-Specialization

نکته) رابطه‌هایی که تا حالا دیدیم، رابطه بین نمونه موجودیت‌ها بود و با کلید اصلی خارجی قابل پیاده‌سازی است. در حالی‌که رابطه کلی-اختصاصی رابطه بین موجودیت‌هاست و ماهیتا متفاوت است.

نکته) این رابطه خیلی شبیه وراثت inheritance پدر-پسری در object oriented است.





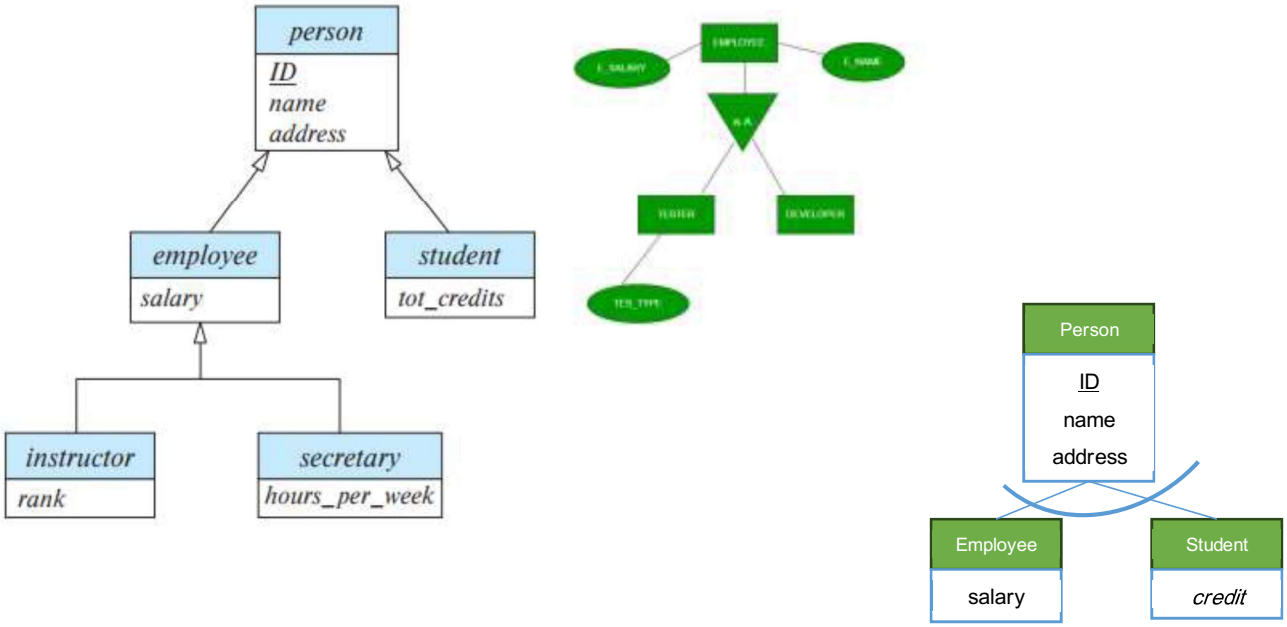


نمایش موجودیت دانشجو با نمودار ER

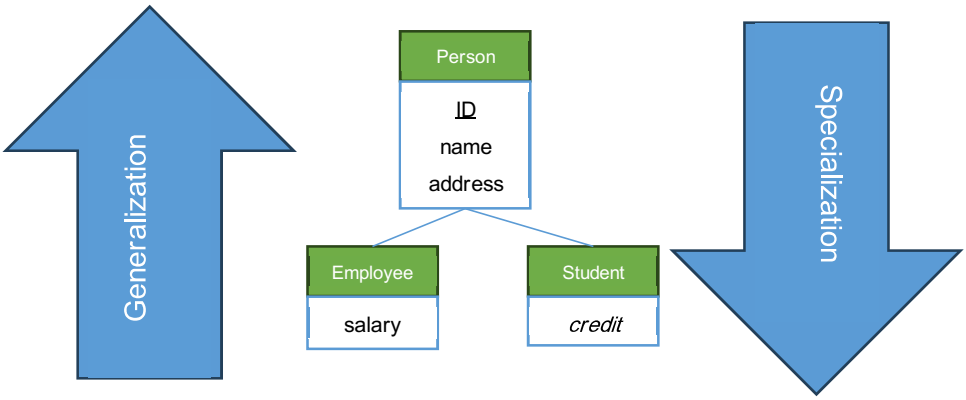
Student				
name	Address	Year	Age	Phone
Hadi	Qom	1374	28	09128303030, 09157387431
Ali	Bam	1372	30	09128303030
Sara	Yazd	1375	27	09156783254

نمایش موجودیت دانشجو و نمونه‌های آن با جدول

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید. استفاده از این جزوه بدون خرید فیلم‌ها غیرقانونی و غیرشرعی است.

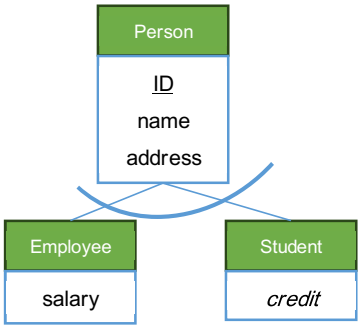


3.7.1 وجه تسمیه



3.8 انواع رابطه specialization و generalization

سوال) در نمودار زیر آیا یک نفر می‌تواند هم کارمند باشد هم دانشجو؟



پاسخ) این بستگی به نیازمندی‌ها و قواعد سیستم (دانشگاه) دارد. اگر یک فرد بتواند هم دانشجو باشد هم کارمند، اصطلاحا مجموعه موجودیت‌های کارمند و دانشجو می‌تواند عضو مشترک داشته باشند. یا اصطلاحا همپوشانی overlap داشته باشند.

ممکن است یک قانون یک دانشگاه این اجازه را ندهد. یک کارمند نتواند در دانشگاه تحصیل کند. در این صورت مجموعه موجودیت‌های کارمند و دانشجو همپوشانی ندارند یا اصطلاحا disjoint هستند.

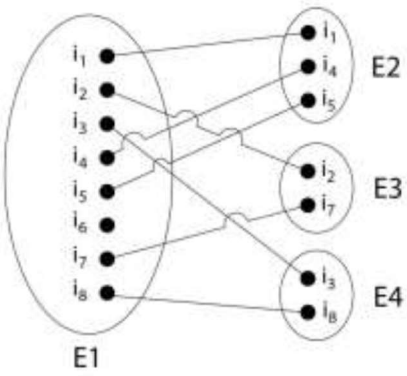
3.8.1 انواع رابطه generalization-specialization از منظر اشتراک

1. تعریف رابطه overlapping) اگر موجودیت‌های فرزند بتوانند عضو مشترک داشته باشند. رابطه از نوع overlapping است.
2. تعریف رابطه disjoint) اگر در موجودیت‌های فرزند عضو مشترک مجاز نباشد، رابطه از نوع disjoint است.

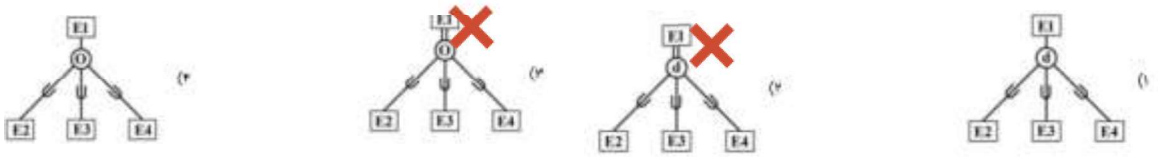
نکته) به طور پیش فرض رابطه از نوع overlapping است. یعنی اگر نوع رابطه مشخص نباید آن را می‌توان overlapping در نظر گرفت.

3.8.1.1.1. تست سراسری مهندسی 1398 4 ER ارثبری disjoint

نمونه هایی از چهار نوع موجودی E3، E2، E1و E4در شکل زیر نمایش داده شده است. کدام مورد بهترین نمودارEER معرف محیط است



(مهندسی کامپیوتر ۱۳۹۸)



پاسخ) گزینه 1صحیح است. با توجه به اینکه هریک از نمونه های موجود در موجودیت E1به حداکثر یک نمونه از نمونه های موجود در موجودیت E3، E2و E4نگاشت شده اند)یعنی نمونه های موجود در این 3موجودیت با یکدیگر اشتراکی ندارند(پس این حالت یک رابطه وراثت از نوع disjointاست) همچنین از یک نمونه از موجودیت E1فلشی خارج نشده، که این نشاندهنده اختیاری بودن شرکت کردن در رابطه است. بهترین گزینه برای توصیف این حالت گزینه 1 است

3.8.2 انواع رابطه gen-spec از منظر کامل بودن

1. تعریف total) هر نمونه موجودیت پدر (بالادست) باید به یک موجودیت پسر (فرودست) مربوط شود
2. تعریف partial) نمونه موجدیت پدر (بالادست) می‌تواند در رابطه با موجودیت پسر (فرودست) نباشد.

نکته) در رابطه total با اضافه کردن نمونه موجودیت به جدول پدر باید پسر متناظر آن هم به جدول فرودست اضافه شود. در رابطه total با حذف نمونه موجودیت از جدول پسر باید پدر متناظر آن هم از جدول پدر حذف شود.

نکته) به طور پیش فرض رابطه از نوع partial است.

3.9 پیاده‌سازی رابطه generalization

برای پیاده‌سازی رابطه generalization روش‌های مختلفی وجود دارد. در این جا جزییات بیشتری درباره این روش‌ها، مقایسه آنها و اینکه هر کدام برای کدام نوع مناسب تر است می‌پردازیم.

3.9.1 جدولهای جدا

در این روش از ارثبری صرفنظر کرده و فقط برای موجودیتهای پسر (فرودست) یک جدول جدا ایجاد می کنیم. به این خاطر مجبوریم که خصوصیات یکسان در هر دو جدول تکرار کنیم.

Student(ID, name, address, credit)

Employee (ID, name, address, salary)

- ID کلید اصلی پدر بوده و یک خصیصه یکتا در محیط است مثلا شماره ملی
- در این روش هیچ نوع ارتباطی بین دو جدول نداریم.
- این پیاده‌سازی برای generalization disjoint complete بهترین است.
- استفاده از این پیاده‌سازی برای overlapping باعث ذخیره مقادیر تکراری خواهد شد. مثلا فرض کنید hadi هم دانشجو باشد هم کارمند. مجبوریم نام و آدرس hadi را دوبار ذخیره کنیم.
- به این پیاده‌سازی برای disjoint partial باید person(ID, name, address) اضافه شود تا افرادی که نه دانشجو هستند نه کارمند را ذخیره کنیم.

3.9.2 ادغام

در این روش هم از ارثبری صرفنظر شده و تمامی خصیصه‌ها در یک جدول ادغام می‌شود.

Person(ID, name, address, credit, salary)

بدیهی است که در سطر دانشجویان مقادیر salary و در سطر اساتید credit ، null می‌ماند. تعداد زیادهای null مشکل این روش است.

3.9.3 جدول‌های مجزا برای پدر و پسر

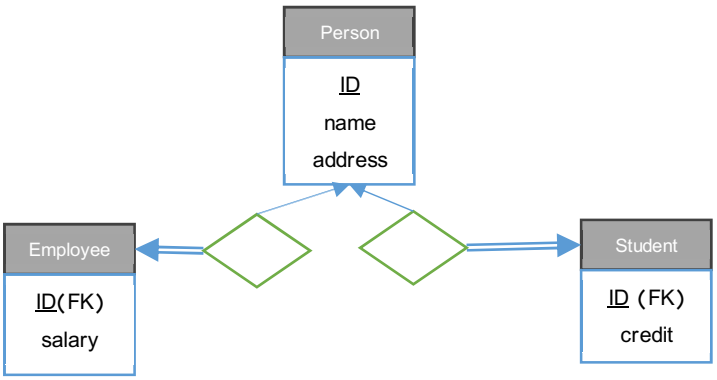
در این روش سعی می‌شود با کلید خارجی مفهوم ارثبری پیاده شود.

person(ID, name, address)

Student(ID F.K, credit)

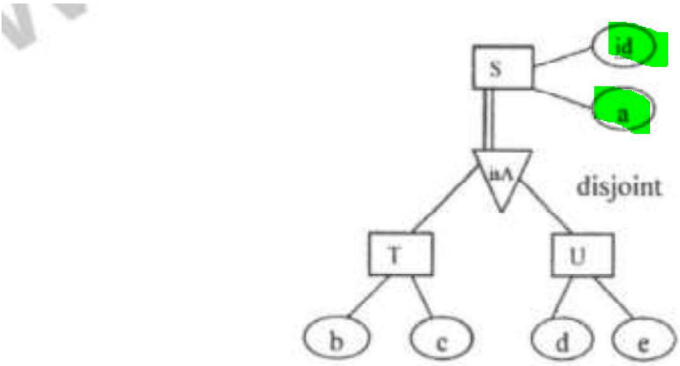
Employee (ID F.K, salary)

- ID در هر سه جدول کلید اصلی است.
- ID در جدول‌های فرودست به عنوان کلید خارجی ارجاع‌دهنده به person ID تعریف می کنیم.



3.9.3.1.1.1 تست سراسری آی تی 1392 78 4 ER ارثبری disjoint

قطعه زیر را از یک نمودار ER را در نظر بگیرید.



کدام یک از مجموعه رابطه‌های زیر برای تبدیل این قطعه به مدل رابطه‌ای مناسب‌ترین است؟

۱) T(id, a, b, c), U(id, a, d, e)

۲) S(id,a), T(id, b, c), U(id, d, e)

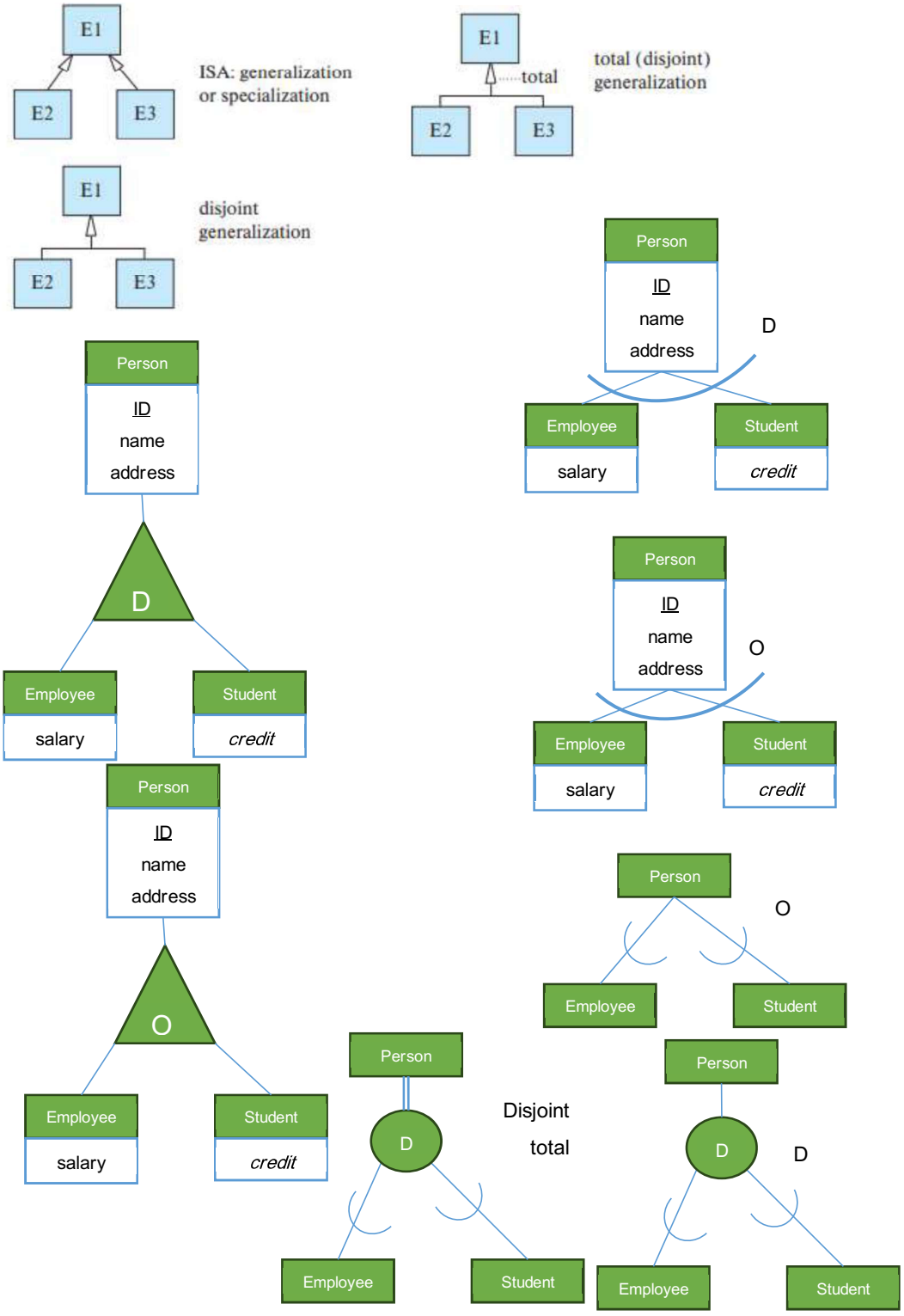
۳) R(id, a, b, c, d, e, Type) . مقدار Type یکی از سه مقدار S, T یا U می باشد.

۴) S(id, a, b, c, d, e, isT, isU). دو مقدار isT و isU مقادیر False یا True را می پذیرند.

پاسخ) گزینه 1. رابطه disjoint complete است. بهترین پیاده سازی جدولهای مجزا و صرفنظر از ارثبری است.

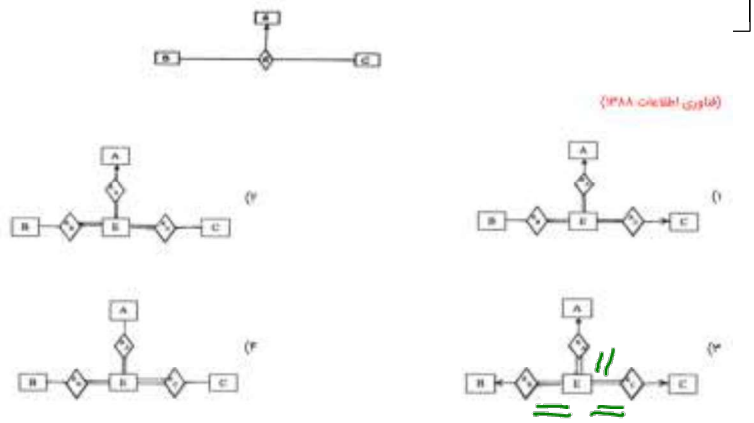
3.9.4 نمادهای گرافیکی generalization

شوربختانه – البته طبق معمول- برای نمایش overlapping و disjoint نمادهای مختلفی داریم.

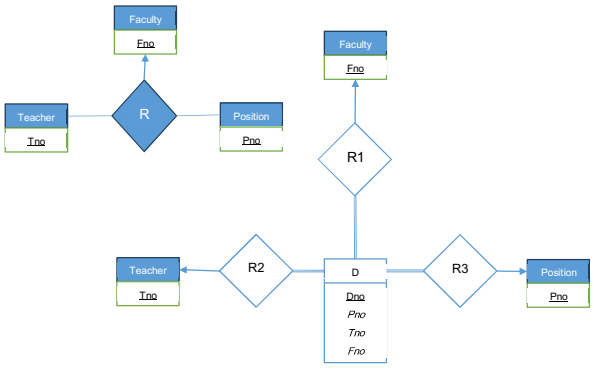


3.9.4.1.1.1 تست سراسری آی تی 1388 4

فرض کنید که رابطه سه تایی زیر بین موجودیتهای Cو Bو Aوجود دارد، حال اگر بخواهیم این رابطه سه تایی را با رابطه‌های دودویی نمایش دهیم، کدام یک از نمودارهای موجودیت – رابطهERD) (زیر دقیقًا معادل با این رابطه سه تایی میباشد؟



پاسخ) گزینه 3 به فصل پیاده درجه N:N:1 3 مراجعه کنید.



3.9.4.1.1.2 تست آزاد آی تی 1385

در نمودر ER تعریف زیر معرف کدام نوع خصیصه Attribute است؟ (آی تی- آزاد 1385)

خصیصه‌ای که در موجودیت و در نهایت در پایگاه داده برای آن فیلدی تعریف نشده است

1) چند مقداری 2) مشتق 3) ساده 4) مرکب

پاسخ) گزینه 2

خصیصه مشتق به دیگر خصیصه‌ها وابستگی داشته و بر اساس آنها قابل محاسبه است. مثل سن دانشجو که بر اساس تاریخ تولد قابل محاسبه است یا معدل دانشجو که بر اساس واحد و نمرات دانشجو قابل محاسبه

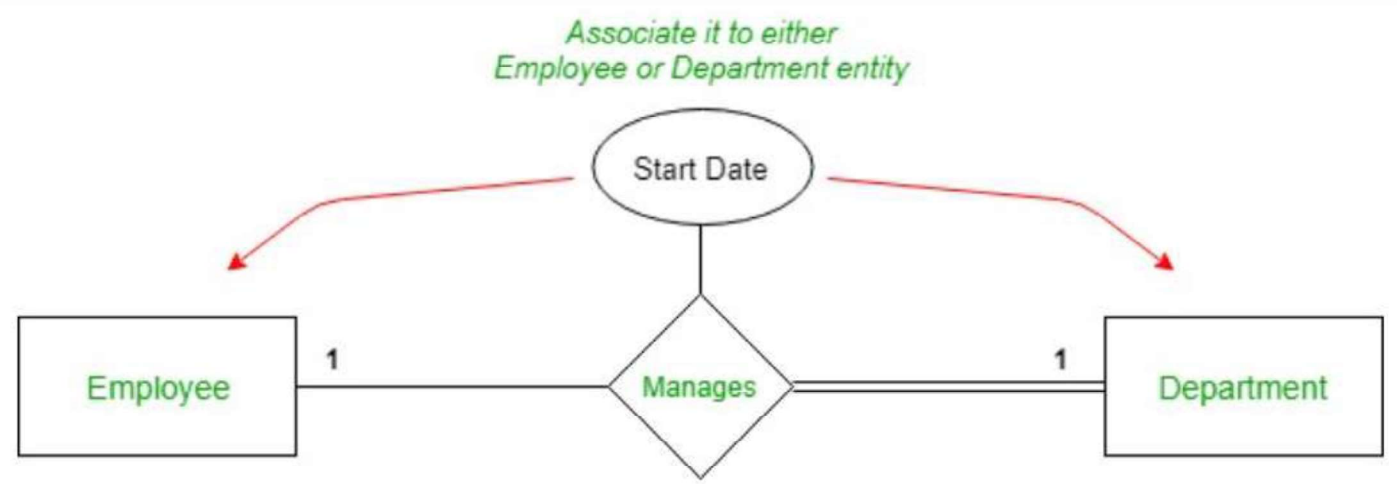
است. طراح پایگاه‌داده درباره ذخیره‌سازی یا عدم‌ذخیره‌سازی خصیصه مشتق تصمیم می‌گیرد. یعنی ممکن است هیچ ستونی برای آن در نظر نگیرد. دقت کنید خصیصه مرکب با اینکه عینا در پایگاه داده رابطه‌ای پیاده‌سازی نمی‌شود ولی اجزای آن به عنوان ستون در جدول تعریف می‌شود. پس بهترین گزینه همان 2 است.

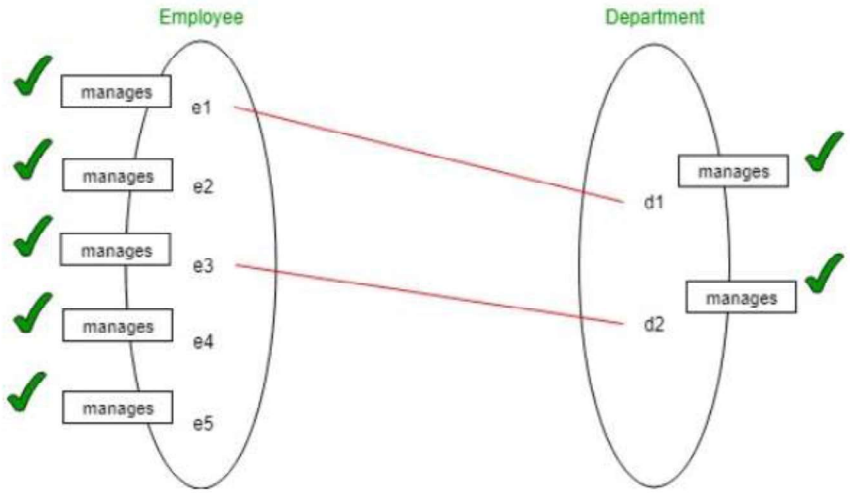
3.10 پیاده سازی خصیصه رابطه

گاهی طراح ER برای یک رابطه خصیصه تعیین می‌کند. به خاطر پیچیدگی و ابهام این کار توصیه نمی‌شود.

3.10.1 رابطه 1 به 1

مثال) در یک شرکت، هر بخش Department توسط یک کارمند Employee اداره می‌شود. و هر نفر Employee می‌تواند یک بخش Department را اداره کند. طراح تاریخ شروع مدیریت Start Date را در رابطه نمایش داده‌است. در پایگاه داده رابطه‌ای، رابطه با کلید خارجی پیاده‌سازی می‌شود، و دادن خصوصیت به رابطه غیر ممکن است. ما می‌توانیم Start Date را به هر کدام از موجودیت‌ها بدهیم.





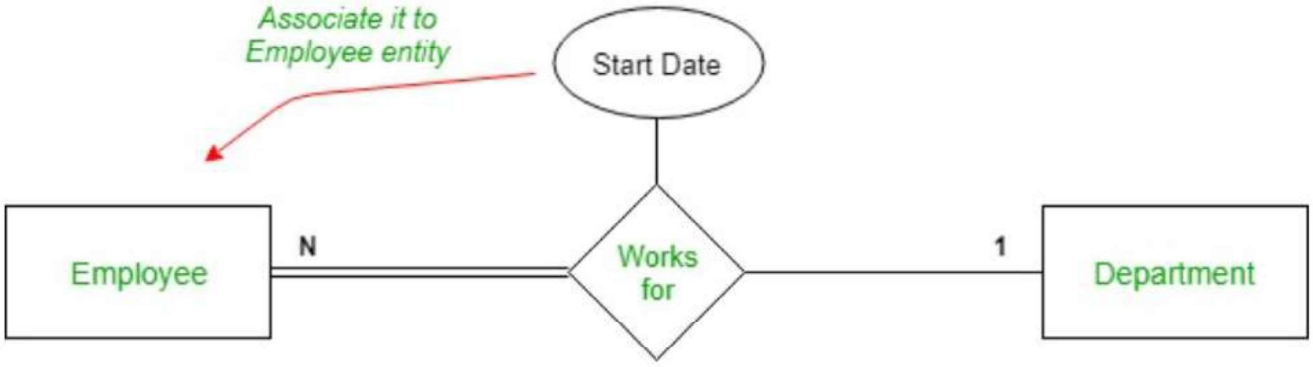
نکته) در رابطه 1 به 1 ، خصیصه رابطه را به هر دو موجودیت می‌توانیم اضافه کنیم.

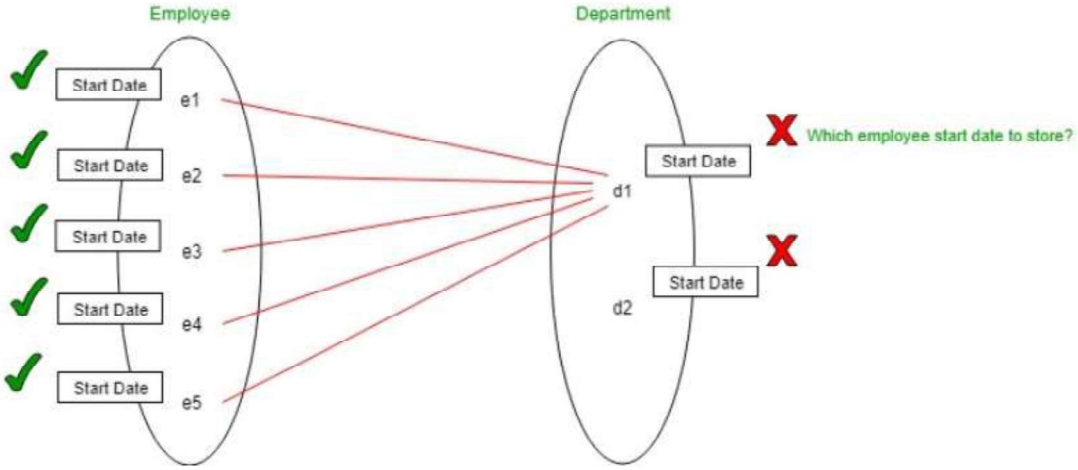
3.10.2 رابطه یک به چند

مثال) در این شرکت هر کارمند Employee می‌تواند برای یک بخشDepartment کار کند. ولی هر بخش Department می‌تواند کارمندان زیادی داشته باشد. Start Date در رابطه زمان شروع به کار را نشان می‌دهد. اگر Start Date را در موجودیت بخش پیاده‌سازی کنیم، نمی‌تواند تاریخ به شروع همه کارمندهای آن بخش را ذخیره کند. پس بدیهی است که Start Date باید در Employee پیاده شود.

نکته) در رابطه 1 به N ، خصیصه رابطه را در موجودیتی که با N به رابطه وصل شده باید پیاده‌سازی کرد.

نکته) یادآوری می کنیم که در رابطه 1:N، کلید اصلی موجودیت متصل به خط 1 باید به موجودیت دیگر اضافه می‌شد و آنجا به عنوان کلید خارجی به کلید اصلی موجودیت متصل به خط 1 ارجاع می‌داد.

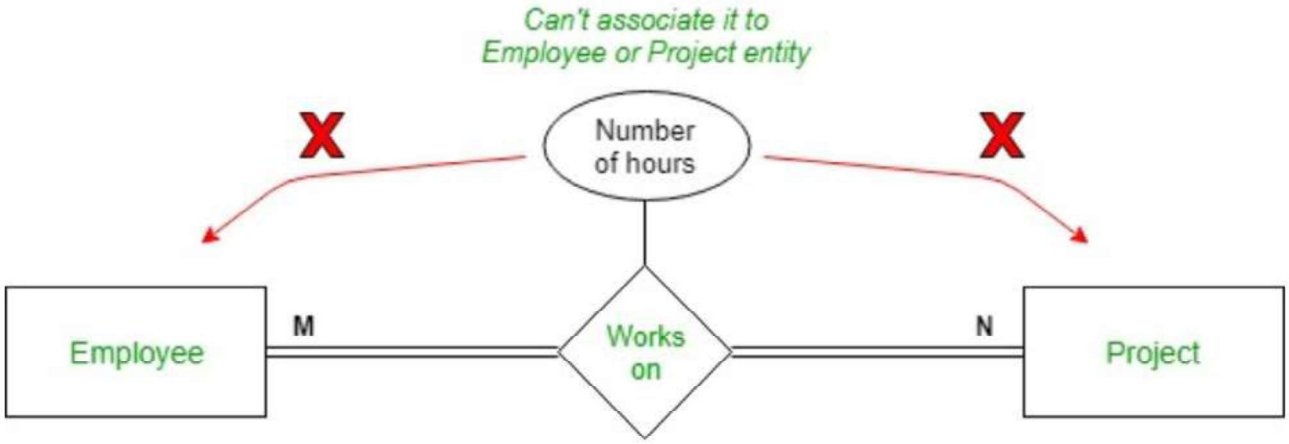


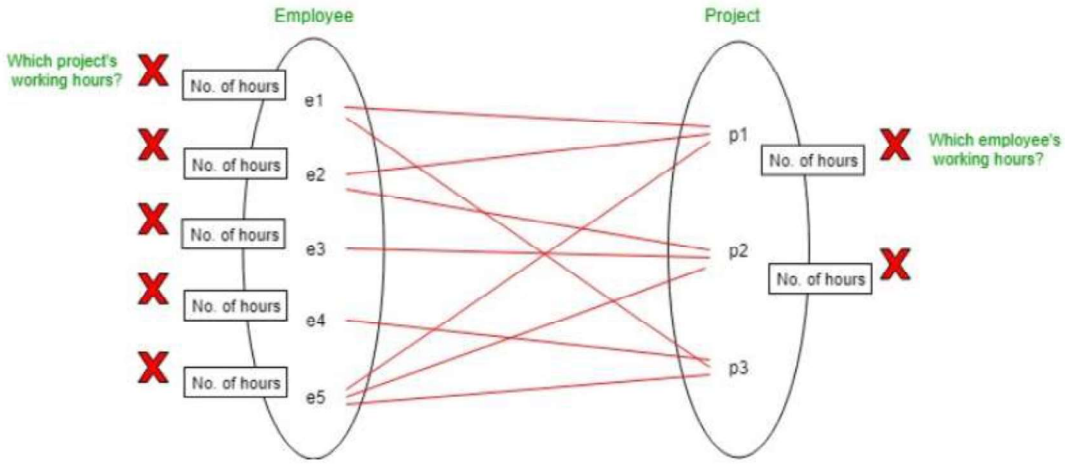


3.10.3 رابطه چند به چند

مثال) در شرکت یک کارمند Employee باید روی یک یا چند (N) پروژه Project کار کند. و هر پروژه هم باید یک یا چند M کارمند داشته باشد. طراح خصیصه ساعت کاری Number of hours را به رابطه مربوط کرده‌است. این خصیصه را نه می‌توان به موجودیت کارمند و نه موجودیت پروژه اضافه کرد. زیرا برای هر کارمند می‌توان N پروژه داشت. اگر Number of hours را به جدول کارمند اضافه کنیم. ساعت کاری کدام پروژه را در آنجا باید ذخیره کنیم؟! به همین شکل اگر Number of hours را به جدول پروژه اضافه کنیم به مشکل می‌خوریم.

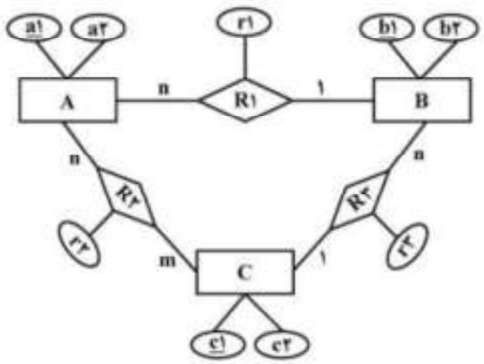
نکته) به یاد بیاورید که در رابطه N به M باید یک موجودیت جدید برای رابطه جعل می‌کردیم. به همین موجودیت جعلی می‌توان خصوصیت Number of Hours را هم اضافه کرد.





3.10.3.1.1.1 تست سراسری آی تی 1398 82

مدل رابطه‌ای معادل کدام گزینه است؟



۱) $A(\underline{a_1,a_2})\ B(\underline{b_1,b_2})\ C(\underline{c_1,c_2})\ R_1(\underline{a_1,b_1,r_1})\ R_2(\underline{a_1,c_1,r_2})\ R_3(\underline{b_1,c_1,r_3})$

۲) $A(\underline{a_1,a_2})\ B(\underline{b_1,b_2},a_1,r_1)\ C(\underline{c_1,c_2},b_1,r_2)\ R_1(\underline{a_1,b_1,r_1})\ R_2(\underline{b_1,c_1,r_2})$

۳) $A(\underline{a_1,a_2},b_1)\ B(\underline{b_1,b_2},c_1,r_1)\ C(\underline{c_1,c_2},r_2)\ R_2(\underline{a_1,c_1,r_2})$

۴) $A(\underline{a_1,a_2},b_1,r_1)\ B(\underline{b_1,b_2},c_1,r_2)\ C(\underline{c_1,c_2})\ R_2(\underline{a_1,c_1,r_2})$

پاسخ) گزینه 4

راه‌حل تستی) رابطه R1 و R3 با اینکه خصیصه دارند ولی چون 1:N هستند لازم نیست به عنوان موجودیت مستقل پیاده شود. پس گزینه 1 و 2 غلط است. خصیصه r1 رابطه R1 بهتر است به موجودیتی که با n وصل است یعنی A منتقل شود. پس گزینه 3 غلط است. و تنها گزینه 4 باقی می‌ماند.

راه حل کامل) رابطهR2 M:N است و باید به موجودیت R3 مانند شکل بالا تبدیل شود. R3 علاوه بر خصوصیت اصلی خود باید کلید اصلی C ، c1 و کلید اصلی A، a1 را به خصیصت داشته باشد. اینها کلید خارجی هستند تا بتوان ارتباط ایجاد کنند. ترکیب این دو یکتا بوده و می‌تواند به عنوان کلید اصلی استفاده شود. پس R3(r3, a1, c1)

موجودیت A باید خصوصیت‌های a1 و a2 را داشته باشد. به خاطر رابطه رابطه 1:N با B ، باید کلید اصلی B یعنی b1 کلید خارجی بشود. پس خصیصه b1 هم باید به B اضافه شود. خصیصه r1 رابطه R1 بهتر است به موجودیتی که با n وصل است یعنی A منتقل شود. موجودیت N با موجودیت جعلی R3 با خط 1 وصل است پس لازم نیست خصوصیتی از آن را داشته باشد. پس A(a1,a2,b1,r1)

موجودیت C خصوصیات c1 و c2 را دارد. دقت کنید C با خط 1 با دو موجودیت R3 و B ارتباط دارد. لازم نیست که کلید اصلی این دو موجودیت در B ظاهر شود. پس B(c1, c2)

موجودیت B باید خصوصیت‌های b1 و b2 را داشته باشد. به خاطر رابطه رابطه 1:N با C ، باید کلید اصلی C یعنی c1 کلید خارجی بشود. پس خصیصه c1 هم باید به C اضافه شود. خصیصه r3 رابطه R3 بهتر است به موجودیتی که با n وصل است یعنی B منتقل شود. پس B(b1,b2,c1,r3)

مثال) در ER-Diagram زیر چه نتایجی می‌توان گرفت؟

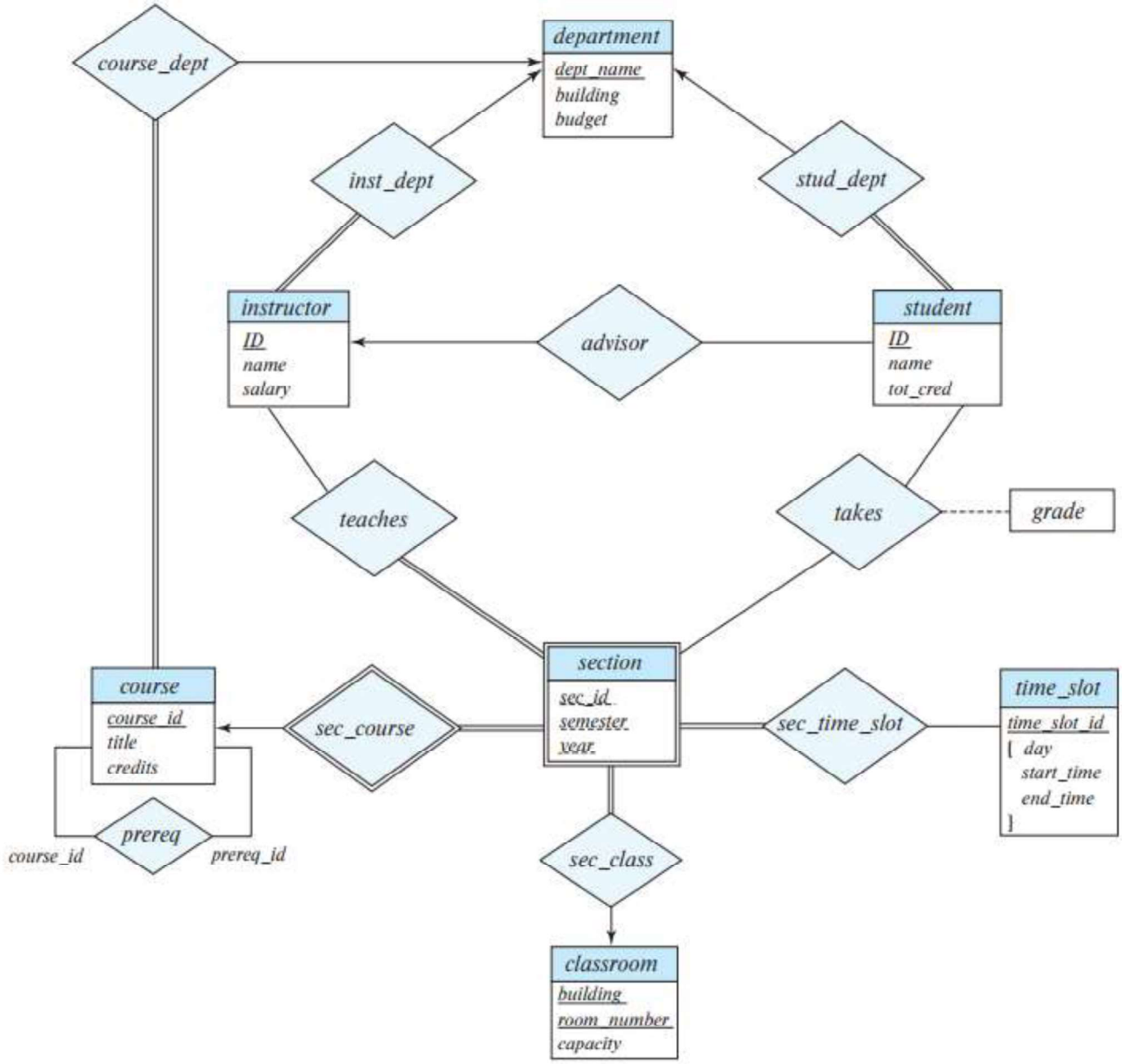
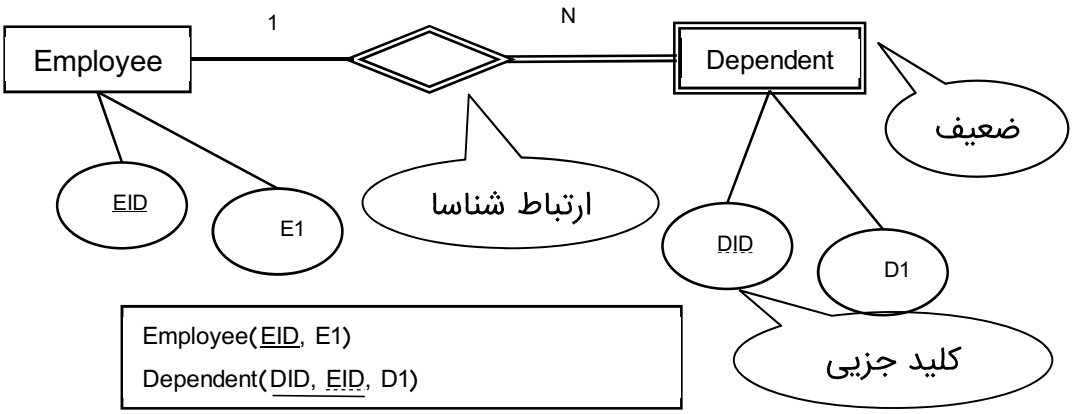


Figure 6.15 E-R diagram for a university enterprise.

3.10 پیاده‌سازی موجودیت ضعیف

- موجودیتی Y ضعیف weak است اگر وجودش وابسته به یک نوع موجودیت دیگر X باشد. یعنی اگر X از مدلسازی حذف شود Y هم حذف شود. به X موجودیت قوی گفته می‌شود.
- موجودیت ضعیف را با مستطیل دو خط در نمودار ER نمایش می‌دهیم.
- مثلاً افراد تحت تکلف کارمند (مانند فرزندان و همسران) کاملا وابسته به کارمند هستند. اگر کارمند از پایگاه داده سازمان حذف شود. افراد تحت تکلف هم حذف خواهند شد.
- نوع موجودیت ضعیف کلید ندارد. بلکه یک صفت ممیز دارد که به آن کلید جزیی هم گفته می‌شود. کلید جزیی قدرت تمایز ندارد. نمی‌توان از کلید جزیی به عنوان کلید اصلی موجودیت ضعیف استفاده کرد. کلید جزیی را معمولا با زیرخط نقطه چین نمایش می‌دهند. دقت کنید برای کلید خارجی هم از همین نماد استفاده می‌شود.
- ارتباط بین موجودیت ضعیف و قوی معمولا درجه ۲ 1:N است. در برخی از منابع به این ارتباط شناسا گفته می‌شود. این ارتباط با لوزی دو خط نمایش داده می‌شود.
- در پیاده‌سازی رابطه کلید موجودیت ضعیف از ترکیب کلید اصلی موجودیت قوی و کلید جزیی موجودیت ضعیف ساخته می‌شود.
- مثلاً نام کوچک اعضای خانواده یک کارمند می‌تواند کلید جزیی باشد. و ترکیب کلید اصلی موجودیت قوی Eno و نام وابسته کلید اصلی موجودیت ضعیف است. دقت کنید که نام کوچک قدرت تمایز

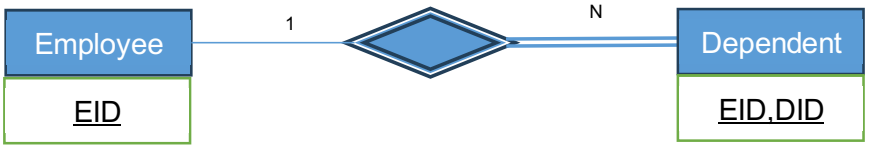
نداشته و به تنهایی نمی‌تواند کلید باشد مثلا دو کارمند ممکن است پسر به نام علی داشته باشند. ولی فرزندان یک کارمند هم نام نیستند.



- یک موجودیت ضعیف می‌تواند همزمان به دو موجودیت قوی وابسته باشد. در این حالت رابطه شناسا درجه سه می‌شود. به خاطر ابهام این گونه از موجودیت ضعیف توصیه نمی‌شود.
- برای حذف یک ارتباط درجه سه می‌توان از یک موجودیت ضعیف استفاده کرد. به عبارت دیگر موجودیت میانی که برای تبدیل ارتباط درجه سه N:N:1 به سه ارتباط درجه دو N:1 استفاده می‌شود می‌تواند موجودیت ضعیف باشد.

3.10.1 پیاده‌سازی SQL

در پیاده‌سازی SQL با استفاده ON DELETET CASCADE و ON UPDATE CASCADE وابستگی را پیاده‌سازی می کنیم. ضمن این تعریف کلید اصلی موجودیت قوی به عنوان کلید خارجی در موجودیت ضعیف رابطه 1:N را به ما می‌دهد.

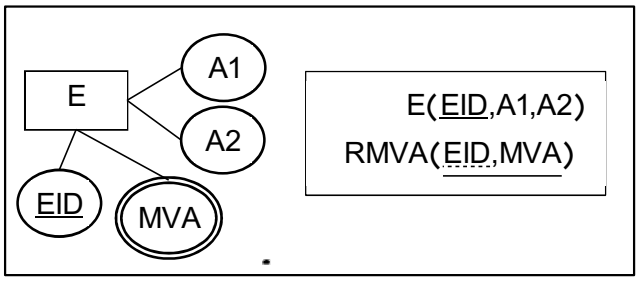



```
CREATE TABLE dependent(...
              FOREIGN KEY (EID) REFERENCES Employee
              ON DELETE CASCADE,
                ON UPDATE CASCADE)
```

3.11 صفت چند مقداری

برای پیاده‌سازی صفت چندمقداری سه روش داریم.

3.11.1 روش اول

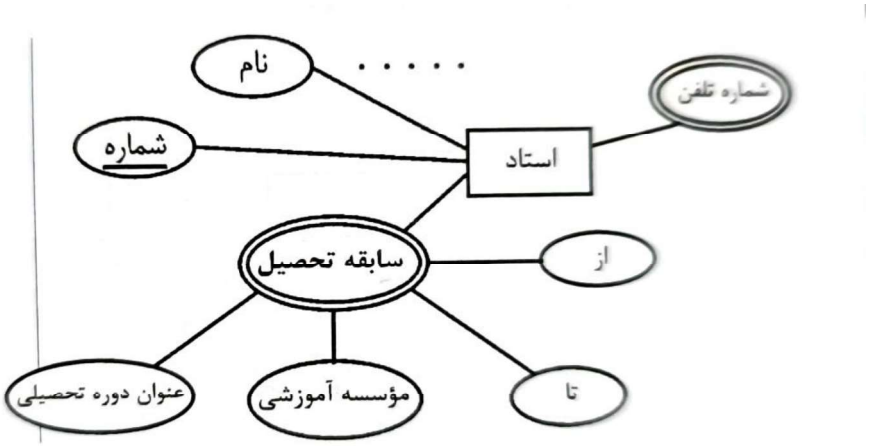


•برای هر صفت چندمقداری یک رابطه (جدول) ایجاد می‌کنیم.

•عیب: در این روش برای رسیدن به همه اطلاعات موجودیت E باید رابطه R را با رابطه RMVA الحاق کنیم.

•این روش در همه صفات چند مقداری قابل استفاده است و در تستهای کنکور عموما از همین روش استفاده می شود.

مثال) پیاده‌سازی صفت مرکب چندمقداری



PROF(PRID , PRNAME , ...)
C.K.

PREDUCHIS(PRID , EDTITLE , INST , FROM , TO)
F.K.
C.K.

PRTEL(PRID , TELN)
C.K.

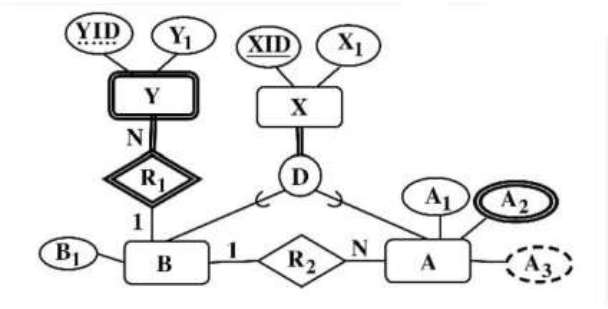
3.11.2 روش دوم

- اگر صفت چند مقداری به ازای هر نمونه موجودیت ***حداقل یک*** مقدار داشته باشد. می‌توان صفت چندمقداری را در رابطه موجودیت وارد کرد.
- فرض کنید استاد فقط صفات ساده شماره و نام و صفت چند مقداری شماره تلفن را داشته باشد.
  Prof(PRID, Tel, PrName)
- عیب این روش افزونگی داده است. به ازی هر شماره تلفن نام و شماره استادی تکرار می‌شود.

3.11.3 روش سوم

- به تعداد حداکثر تعداد مقداری که صفت چندمقداری می‌تواند مقدار داشته باشد صفت ساده به رابطه موجودیت اضافه می‌کنیم. Prof(PRID, PrName, Tel1, Tel2, Tel3)
- عیب این روش NULL های زیاد است.

3.11.3.1.1.1 تست سراسری فناوری اطلاعات ۱۴۰۱



(فناوری اطلاعات ۱۴۰۱)

۱) $Y(\underline{XID, YID}, Y_1)$ $A(\underline{XID}, \underline{BXID}, A_1, A_2, A_3)$

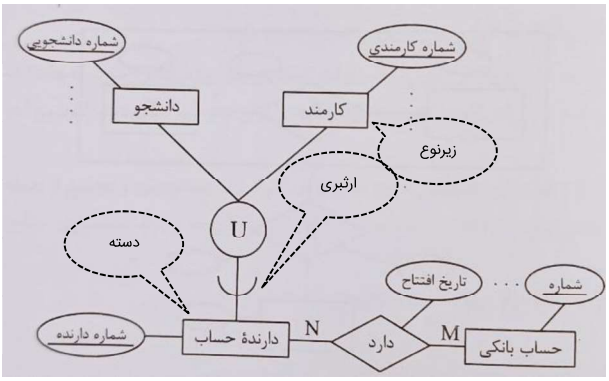
۲) $Y(\underline{YID}, Y_1, \underline{XID})$ $AA2(\underline{XID}, A_2)$ $A(\underline{XID}, A_1, A_3, \underline{BXID})$

۳) $Y(\underline{YID}, Y_1)$ $A(\underline{XID}, A_1, A_2, A_3)$

۴) $Y(\underline{XID}, \underline{YID}, Y_1)$ $AA_2(\underline{XID}, A_2)$ $A(\underline{XID}, A_1, \underline{BXID})$

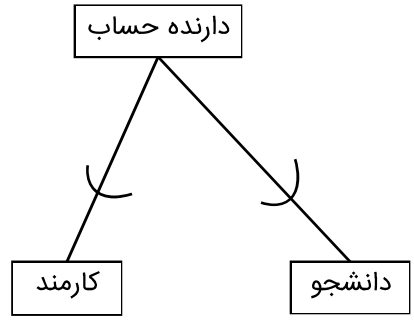
- پاسخ گزینه ۴ صفت A2 با بیضی دو خط نمایش داده شد پس صفت مرکب است اضافه کردن آن به عنوان یک صفت ساده به رابطه A نادرست است. گزینه ۱ و۳ نادرست و رد می‌شود. صفت با بیضی خطچین نمایش داده شده پس A3 صفت مشتق است. لازم نیست آن را به عنوان یک صفت ساده به رابطه A اضافه کنیم. گزینه ۲ و ۳ رد می‌شود. تنها گزینه ۴ باقی می‌ماند که پاسخ است و باید انتخاب شود.
- بحث)گزینه ۲و ۳ به دلیل دیگری هم رد می‌شود. B در یک رابطه پدر پسری با X است از نوع complete disjoint و کلید هم ندارد باید از پدرش کلید بگیرد یعنی B(XID, B1,..). Y یک موجودیت ضعیف است که با B رابطه دارد. YIDZ کلید جزیی است به تنهایی نمی‌تواند کلید باشد. XIDو YID باید با هم کلید Y باشند. Y(XID, YID, Y1). این یک دلیل دیگر است برای رد گزینه ۲ و ۳.

3.12 ارتباط U-Type

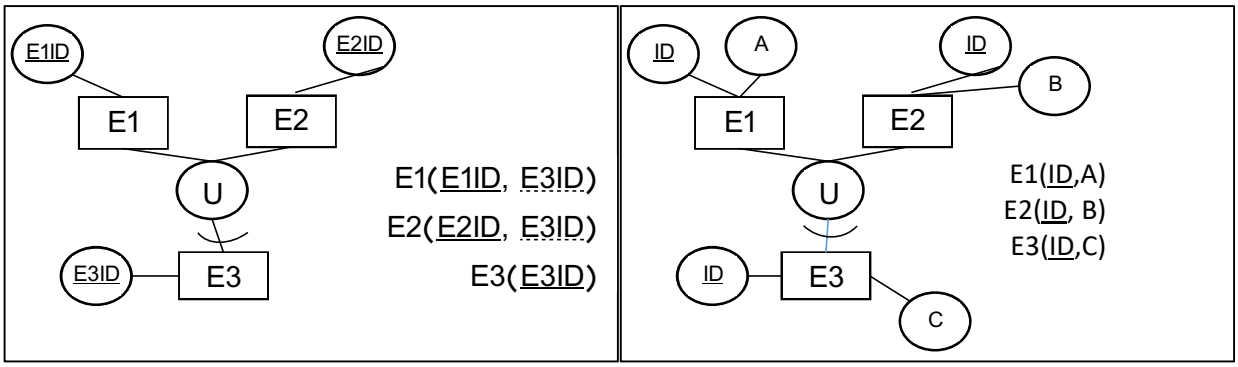


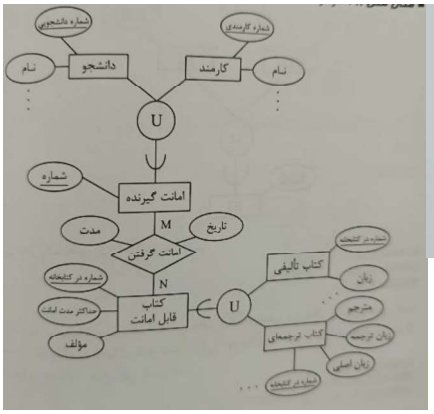
- دارنده حساب می‌تواند از نوع کارمند باشد یا از نوع دانشجو.
- اگر دارنده حساب دانشجو باشد صفات آن را به ارث می‌برد.
- اگر دارنده حساب کارمند باشد صفات آن را به ارث می‌برد.
- به پدرها در این ارتباط زیرنوع گفته می‌شود.
- به پسرها در این ارتباط دسته گفته می‌شود.

- چون کلید اصلی دانشجو شماره دانشجویی با کلید اصلی کارمند یعنی شماره کارمندی تفاوت دارد. دسته باید کلید اصلی خودش را داشته باشد.
- اگر کلید اصلی دانشجو و کارمند یکی باشند -مثلا کدملی- دسته می‌توانست همان کلید اصلی را به ارث ببرد.
- اگر یکی از کارمندان یا یکی از دانشجویان حساب نداشته باشد گفته می‌شود دسته ناقص است.
- اگر همه دانشجوها و همه کارمندان حساب داشته باشند. گفته می‌شود که دسته کامل است.



- در ارتباط u-type اگر دسته کامل باشد آن را می‌توان با ISA نمایش داد. به خصوص وقتی که کلید اصلی زیر نوع‌ها یکی باشد.





رابطه دانشجو : STUD(STID , STNAME , ... , BOID)

رابطه کارمند : EMPL(EMID , ENAME , ... , BOID)

رابطه امانت‌گیرنده : BORP(BOID)

رابطه کتاب قابل امانت : BOBK(BBID , MDUR , AUTR)

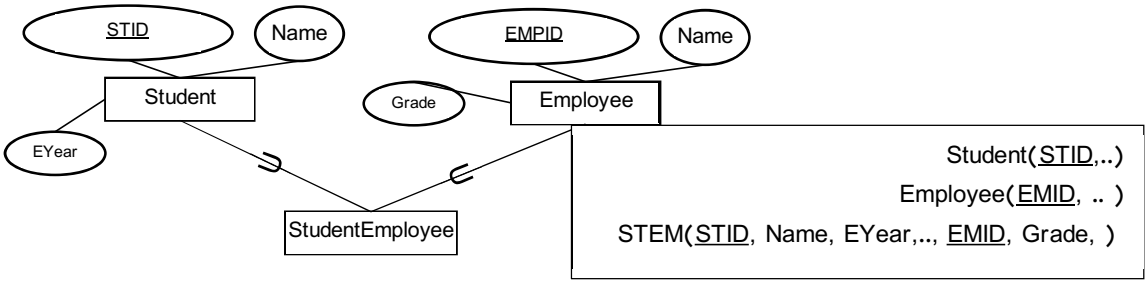
رابطه امانت گرفتن : BWER(BOID , BIDD , DATE , BDUR)

رابطه کتاب تألیفی : ABOK(BBID , LANG , ...)

رابطه کتاب ترجمه‌ای : TBOK(BBID , OLAN , TLAN , TRAN , ...)

3.13 ارثبری چندگانه

- وقتی یک پسر بیش از یک پدر داشته باشد.



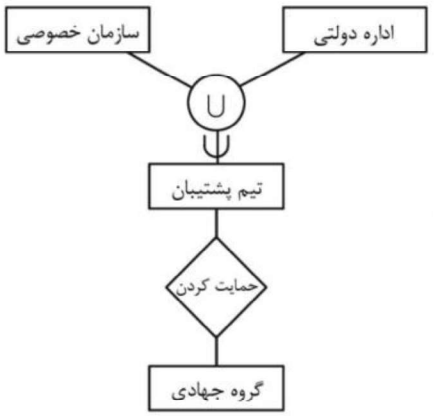
3.13.1.1.1.1 تست سراسری فناوری اطلاعات ۱۴۰۲

۹۹. می‌خواهیم برای عبارت زیر یک EER مناسب رسم کنیم. کدام مورد مناسب‌ترین گزینه است؟

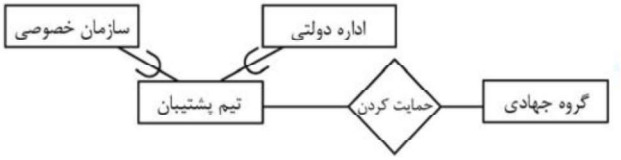
«در هنگام بروز زلزله، تعدادی تیم پشتیبان تشکیل می‌شود که هر تیم پشتیبان از یک سازمان خصوصی یا یک اداره دولتی انتخاب می‌شود. این تیم کار حمایت از گروه‌های جهادی را برعهده دارد. بعضی سازمان‌های خصوصی علاقه‌ای به شرکت در این طرح ندارند.»

(فناوری اطلاعات ۱۴۰۲)

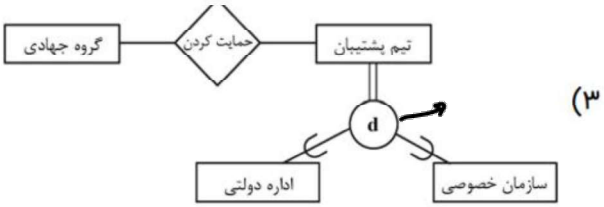
۱)



۲)



۳)



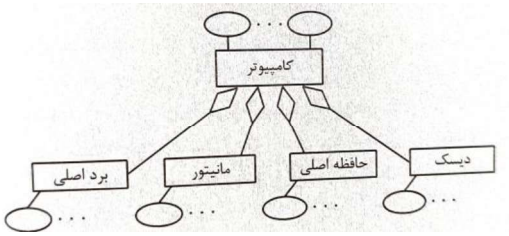
۴) موارد ۱ و ۳ درست است.

پاسخ گزینه ۱ اگر سازمان خصوصی پسر باشد و تیم پشتیبانی پدر. پسر بی پدر نمی‌تواند وجود داشته باشد. یعنی سازمان خصوصی حتماً باید یک تیم پشتیبانی باشد. گزینه ۳ نادرست رد می‌شود. گزینه ۴ هم شامل گزینه ۳ می‌شود و رد می‌شود.

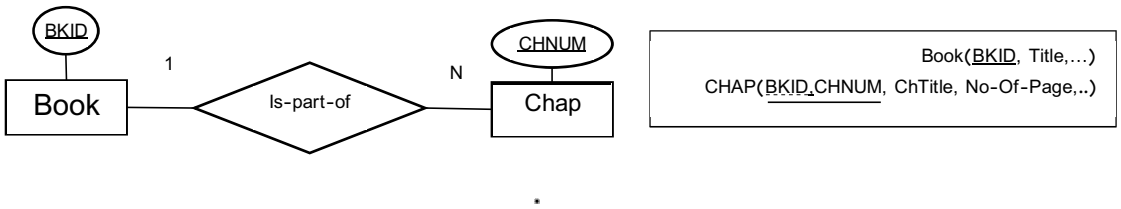
در گزینه ۱ از ارتباط u-type استفاده است. تیم پشتیبان می‌تواند از نوع اداره دولتی باشد یا سازمان خصوصی. و می‌توان نمونه موجودیت سازمان خصوصی داشت که تیم پشتیبان نباشد. این دقیقا همین نکته‌ای است که صورت سوال روی آن تاکید کرده‌است. پس بهترین گزینه همین گزینه است.

در گزینه ۲ تیم پشتیبانی از اداره دولتی و سازمان خصوصی ارثبری چندگانه دارد. یعنی تیم پشتیبانی می‌تواند هم خصوصیات اداره دولتی و هم خصوصیات سازمان خصوصی را داشته باشد. گزینه ۱ چون u-type است پیشنهاد بهتری است.

3.14 ارتباط is-part-of



موجودیت جزء با نوع موجودیت ضعیف از دو لحاظ فرق دارد
۱) نوع جزء از خود شناسه دارد، ولی نوع ضعیف ندارد۲)با حذف نوع کل ، لزوماً نوع جزء از مدلسازی حذف نمی شود به بیان دیگر نوع جزء لزوماً وابستگی وجودی به نوع کل ندارد.



1 وابستگی تابعی Functional Dependency (FD)

1.1 چرا FD؟

چرا باید FD را یاد بگیریم. با کاهش وابستگی داده می‌توانیم افزونگی داده را کاهش دهیم.

مثال) از افزونگی داده (تکرار زاید اطلاعات) در یک جدول

Student_id	Name	Course	Department	Fee	Cno
101	Devi	EL	EE	80,000	1012
101	Devi	EM	EE	70,000	3240
101	Devi	LABEM	GE	10,000	4430
101	Devi	DB	CS	90,000	2022
102	Sona	DB	CS	90,000	2022
103	Varun	DB	CS	90,000	2022
104	Satish	DB	CS	90,000	2022
105	Devi	DB	CS	90,000	2022

سه نوع وابستگی داریم

1. وابستگی تابعی (FD) Functional Dependency ←←←←← مهمترین
2. وابستگی چندمقداری (MVD) Multi-Valued Dependency
3. وابستگی الحاقی (JD) Join Dependency

انواع FD: 1. بد باعث افزونگی 2. بی‌ضرر افزونگی ایجاد نمی‌کند.

دلایل مطالعه FD: 1. انواع FD را بشناسیم 2. FD بد را از بی ضرر تشخیص دهیم. 3. با حذف FD های بد افزونگی را کاهش دهیم.

4.2 FD چیست؟

101	Devi	EL	EE	80,000	1012
101	Devi	EM	EE	70,000	3240
101	Devi	LABEM	GE	10,000	4430
101	Devi	DB	CS	90,000	2022
102	Sona	DB	CS	90,000	2022
103	Varun	DB	CS	90,000	2022
104	Satish	DB	CS	90,000	2022
105	Devi	DB	CS	90,000	2022

از روی student_id می توان name را فهمید. می گوییم. Name وابستگی تابعی دارد به student و نشان می‌دهیم. Student_id → Name

از روی Name نمی‌توان student_id را فهمید.

تعریف FD) اگر R رابطه باشد و a و b دوزیر مجموعه از R.

اگر از a به b برسیم. یا از مقدار a بتوانیم مقدار b را نتیجه بگیریم.

- b وابستگی تابعی دارد به a
- a → b
- به a determinant و به dependent گفته می‌شود.

سوال) آیا در رابطه R (A, B, C, D, E, F) می‌توانیم CD→A ؟

پاسخ) بلی دقت کنید در تعریف FD گفتیم زیرمجموعه از خصیصه‌ها نه یک خصیصه. پس در هر دو طرف علامت پیکان می‌توان مجموعه داشته باشیم.

تعریف FD بدیهی) بعضی از FD ها درباره هر خصیصه‌ای صادق است و به آنها بدیهیTrivial گفته می‌شود.

سوال) چند نمونه FD بدیهی مثال بزنید؟

مثلا اگر نام یک نفر هادی باشد می‌توانیم بگوییم نام او هادی است. پس از روی نام به نام میرسیم.

نام → نام

مثلا اگر نام یک نفر هادی و نام خانوادگی‌اش خانی باشد می توان نتیجه گرفت که نامش هادی است.

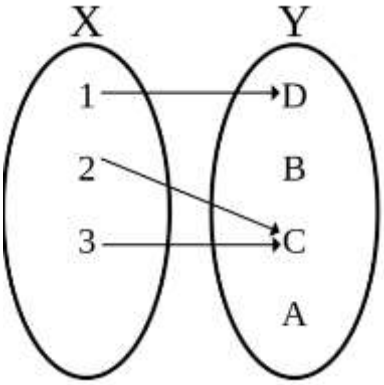
نام → نام و نام خانوادگی

تعریف دقیق FD بدیهی) $A \subseteq B \Rightarrow B \rightarrow A$

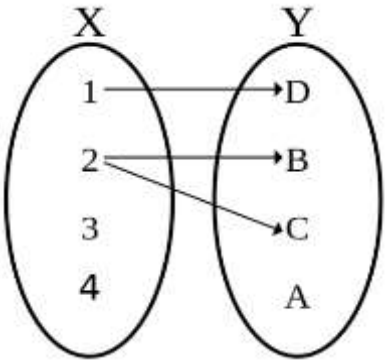
نکته) دقت کنید که همیشه $A \subseteq A \Rightarrow A \rightarrow A$

4.2.1 وجه تسمیه

یادآوری تعریف تابع از ریاضی.



تابع هست



تابع نیست

4.3 بستار یک مجموعه FD

سوال) R(NationalID , studentID ,Faculty, Name) با مجموعه وابستگی تابعی F={NationalID→StudentID, StudentID→Faculty} چه وابستگی تابعی دیگری را می‌توان نتیجه گرفت؟

پاسخ) NationalID→Faculty

نکته) سوال بالا را می‌توان این گونه نوشت.

R(A, B, C, D) با مجموعه وابستگی تابعی F={A→B, B→C} چه وابستگی تابعی دیگری را می‌توان نتیجه گرفت؟ A→C

به مجموعه $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ بستار مجموعه F گفته می‌شود.

تعریف بستار یک مجموعه تابع وابستگی) اگر یک مجموعه وابستگی تابعی F را داشته‌باشیم. به مجموعه تمامی وابستگی‌های تابعی که از F می‌توان نتیجه گرفت، بستار F گفته می‌شود و با F^+ نمایش داده می‌شود.

نکته) بدست آوردن بستار یک مجموعه زمانبر است. معمولا در سوالات کنکور به شما یک F داده می‌شود و می‌پرسند کدام گزینه در بستار F است، یا نیست.

4.3.1 اصول و قوانین آرمسترانگ

برای بدست آوردن بستار از مجموعه اصول و قوانین آرمسترانگ استفاده می‌شود.

	ورودی	نتیجه	انگلیسی	فارسی	اهمیت	
اصول	$Y \subseteq X$	$X \rightarrow Y$	Reflective	بازتابی		به این بدیهی Trivial هم گفته می شود
	$X \rightarrow Y$	$XZ \rightarrow YZ$	augmentation	افزایشی	مهم	برعکسش درست نیست. نمی‌توانیم ساده کنیم.
	$X \rightarrow Y, Y \rightarrow Z$	$X \rightarrow Z$	Transitivity	انتقالی	مهم	
قواعد	$X \rightarrow Y, X \rightarrow Z$	$X \rightarrow YZ$	Union	اجتماع	مهم	اجتماع و تجزیه عکس هم هستند
	$X \rightarrow Y, WY \rightarrow Z$	$WX \rightarrow Z$	Pseudo-trans	شبه‌انتقالی	مهم	
	$X \rightarrow YZ$	$X \rightarrow Y, X \rightarrow Z$	decomposition	تجزیه		
	$X \rightarrow Y, Z \rightarrow W$	$XZ \rightarrow YW$	composition	ترکیب		
	$X \rightarrow Y, Z \rightarrow W$	$X\ U\ (Z-Y) \rightarrow YW$	General Unification	همسان‌سازی کلی		این قانون ارتقا یافته ترکیب است. سعی می‌کند طرف چپ را کوچک کند.

- اصول آرمسترانگ sound درست است. یعنی از این اصول نمی‌توان، FD نادرست نتیجه گرفت.
- اصول آرمستراگ complete کامل است. یعنی برای مجموعه FD های F با این سه اصل می‌توان تمامی اعضای F^+ را تولید کرد.
- قواعد آرمسترانگ از اصول نتیجه گرفته می‌شود. و در محاسبه +F به ما کمک می‌کند.

سوال) برای قوانین آرمسترانگ در پایگاه داده ثبت احوال مثال بزنید؟

قاعده بازتابی Ref) از روی نام و نام خانوادگی می‌توان به نام رسید.

قاعده افزایشی Aug)

نام و نام خانوادگی → کدملی و نام خانوادگی => نام →کد ملی

دقت کنید که برعکس افزایشی (به تعبیری) فاکتور گرفتن درست نیست. فرض کنید در ایران ترکیب شماره شناسنامه و محل صدور منحصر بفرد است. یعنی در یک اداره ثبت‌احوال شماره شناسنامه تکراری نداریم.

کد ملی و محل صدور →شماره شناسنامه و محل صدور

نمی توان نتیجه گرفت

کد ملی → شماره شناسنامه

نکته) از قانون افزایش به راحتی می‌توان نتیجه گرفت به سمت چپ هر خصیصه‌ی که دوست دارید را می‌توانید اضافه کنید. $X \rightarrow Y \Rightarrow XAB..Z \rightarrow Y$

قاعده اجتماع Union)

شماره شناسنامه → کدملی

نام →کدملی

می توان گفت

شماره شناسنامه و نام → کدملی

قاعد شبه انتقالی Pseudo)

نام →شماره شناسنامه و محل صدور

محل صدور→کدملی

می‌توان نتیجه گرفت

نام → شماره شناسنامه و کدملی

تجزیه Deco)

شماره شناسنامه → کدملی و نام → کدملی => شماره شناسنامه و نام → کدملی

دقت کنید که فقط سمت راست را می‌توان تجزیه کرد

نام → شماره شناسنامه و کدملی

نمی‌توان نتیجه گرفت

نام →کدملی

نام → شماره شناسنامه

مثال) قاعده شبه انتقالی pseudo را اثبات کنید.

پاسخ) قاعده می گوید X→Y WY→Z => WX→Z

1. با استفاده از قاعده افزایشی Augm به دو طرف X→Y، W اضافه می کنیم خواهیم داشت. WX→WY
2. با استفاده از انتقالی Trans WX→WY , WY→Z خواهیم داشت WX→Z

راه حل بالا به جای فارسی نوشتن به شکل زیر خلاصه نوشته می‌شود.

X→Y Aug => WX→WY , WY→Z Trans. => WX→Z

مثال) رابطه R(A, B, C, D, E) با مجموعه وابستگی‌های تابعی F={C→E, BC→AD, E→A} را در نظر بگیرید، آیا می‌توان نتیجه گرفت C→AE؟

بلی C→AE Union => C→A, C→E Tran. => C→E, E→A (راه حل

مثال) با فرض رابطه R(A, B, C, D, E, G) و F={A→BC, B→E, CD→EG} ، آیا می‌توان گفت AD→G عضو بستار F هست؟

بلی AD → G Decomp. => AD→EG Pseud => CD→EG, A→C A→B Decomp. => A→BC (راه حل

مثال) برای R(A,B,C,D,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} را داریم.

سوال) آیا می‌توان نتیجه گرفت A→H؟

پاسخ) بلی. A→B, B→H Trans => A→H

سوال) آیا می‌توان نتیجه گرفت CG→HI؟

پاسخ) بلی. CG→I, CG→H Union => CG→HI

سوال) آیا می‌توان نتیجه گرفت AG→I؟

پاسخ) بلی. CG→I, A→C Pseudo => AG→I

4.3.1.1.1.1 تست سراسری آی تی 1388 86 4

کدام مورد، درباره وابستگیهای تابعی(Functional Dependencies) غلط میباشد؟

کدام مورد، درباره وابستگیهای تابعی(Functional Dependencies) غلط میباشد؟

۱) $AB \to C, D \to AE, AF \to C, EF \to GA \Rightarrow AD \to C$

۲) $AB \to C, D \to AE, AE \to C, EF \to GA \Rightarrow DF \to G$

۳) $X \to Y \wedge YZ \to W \Rightarrow XZ \to W$

۴) $X \to Y \wedge Y \to Z \wedge X \to W \Rightarrow XZ \to W$

پاسخ) گزینه 1

گزینه3) X→Y YZ→W Psedo => XZ→W

گزینه 4) X→W => XZ→W

اگر رابطه بالا بدیهی نیست. X→W Z→Z Comp => XZ→WZ Decomp => XZ→W

گزینه 2) D→AE Decomp => D→A D→E EF→GA => DF→GA Decomp => DF→G

استنتاج گزینه 1 صحیح نیست. D→AE Decomp => D→A AB→C Pseudo => DB→C

4.3.1.1.1.2 تست سراسری آی تی 1391 4

برای رابطه R(A,B,C,D,E) و وابستگی های تابعی زیر A→B, BC→D, E→C کدام یک از وابستگی‌های تابعی زیر لزوماً در R برقرار نیست؟ (ارشد IT سراسری 1399)

1) CE→D 2) AE→C 3) BC→B 4) AC→D

پاسخ) گزینه 1

گزینه 2) E→C => AE→C اصولا به طرف چپ هر چیزی را می‌توان اضافه کرد.

گزینه 3) بدیهی است زیر B زیر مجموعه BC است.

گزینه 4) A→B BC→D Pseudo => AC→D

4.3.2 الگوریتم تولید بستار

```
F+ = F
apply the reflexivity rule /* Generates all trivial dependencies */
repeat
    for each functional dependency f in F+
        apply the augmentation rule on f
        add the resulting functional dependencies to F+
    for each pair of functional dependencies f1 and f2 in F+
        if f1 and f2 can be combined using transitivity
            add the resulting functional dependency to F+
until F+ does not change any further
```

Figure 7.7 A procedure to compute F^+.

الگوریتم بالا تمامی اعضای F^+ را حساب می‌کند. سمت راست و سمت چپ همه FD ها، زیر مجموعه R هستند. اگر R، n عضو داشته باشد 2^n زیرمجموعه خواهد داشت. تمام FD های ممکن $2^n x 2^n = 2^{2n}$ خواهد بود. هر تکرار الگوریتم بالا حداقل یک FD جدید به F^+ اضافه خواهد کرد، به جز آخرین تکرار. پس الگوریتم بالا بالاخره تمام می‌شود.

سوال) الگوریتم procedure to compute F^+ تمامی FD های قابل رسیدن را تنها با استفاده از سه اصلی آمسترانگ reflective، transitivity و augmentation به ما می‌دهد. پس چرا باید از قواعد آرمسترانگ استفاده کنیم.

پاسخ) اجرای الگوریتم زمانبر است. در کنکور و امتحان کتبی باید از قواعد استفاده شود تا به سرعت به جواب برسیم.

4.4 بستار مجموعه‌ای از خصوصیت‌ها

اگر R یک رابطه، F مجموعه‌ای از FDهای این رابطه و α زیرمجموعه‌ای از خصوصیتهای R باشد. بستار α تحت F عبارت است از تمامی خصوصیت‌های قابل استنتاج (رسیدن) از مجموعه α با استفاده از F. بستار α را با α^+ نمایش می‌دهند.

4.4.1 الگوریتم محاسبه بستار مجموعه خصوصیت

```
result := α;
repeat
    for each functional dependency β → γ in F do
        begin
            if β ⊆ result then result := result ∪ γ;
        end
until (result does not change)
```

Figure 7.8 An algorithm to compute α^+, the closure of α under F.

مثال) برای R(A,B,C,D,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} را داریم. با استفاده از الگوریتم محاسبه بستار خصوصیت، {AG}$^+$ را حساب کنید؟

پاسخ)

1. Result = {AG}
2. A→B, A⊂ result => result = {AGB}
3. A→C, A ⊂ result => result = {ABCG}
4. CG→H, CG ⊂ result => result = {ABCGH}
5. CG→I, CG⊂ result => result = {ABCGHI}
6. {AG}$^+$={ABCGHI}

4.4.2 کاربردهای الگوریتم محاسبه بستار مجموعه خصوصیت

- پیدا کردن ابرکلید: آیا α ابر کلید super-key است؟ باید α^+ را محاسبه کنید و بررسی کنید که آیا α^+ تمامی خصوصیتهای R را شامل می‌شود.
- تحقیق درستی یک FD: آیا $\alpha \rightarrow \beta$ درست است؟ باید α^+ را محاسبه کنید و بررسی کنید که آیا $\beta \subset \alpha^+$ هست؟ اگر بود آنگاه می‌توان نتیجه گرفت $\alpha \rightarrow \beta$
- محاسبه بستار مجموعه وابستگی‌های تابعی یعنی F^+ : با الگوریتم محاسبه بستار مجموعه خصوصیت‌ها می‌توان بستار مجموعه FD ها را حساب کرد. برای هر $\gamma \subset R$ ، بستار γ^+ را حساب کنید. برای هر $S \subset \gamma^+$ میتوانیم بگوییم $\gamma \rightarrow S \subset F^+$ است.

4.4.3 تعریف کلید با FD بستار

تعریف ابرکلید: اگر R یک رابطه، α زیرمجموعه‌ای از خصوصیت‌های R باشد. اگر α^+ شامل تمامی خصوصیت‌های R باشد، آنگاه α یک ابرکلید super-key است.

یادآوری تعریف ابرکلید در جبر رابطه‌ای: ترکیبی از یک یا چند attribute یا مشخصه (ستون) که منحصر بفرد باشد. یعنی اگر مقادیر ابرکلید را بدانیم ما به یک سطر به خصوص از جدول می‌رسیم و از آنجا می‌توانیم مقادیر دیگر ستون‌ها را نتیجه‌گیری کنیم.

مثال) آیا برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} آیا ابرکلید {AG} است؟

پاسخ) خیر. وقتی بستار را محاسبه کردیم D را شامل نشد پس نمی‌تواند بستار باشد.

1. Result = {AG}
2. A→B, A⊂ result => result = {AGB}
3. A→C, A ⊂ result => result = {ABCG}
4. CG→H, CG ⊂ result => result = {ABCGH}
5. CG→I, CG⊂ result => result = {ABCGHI}
6. $\{AG\}^+$={ABCGHI}

مثال) در مثال بالا برای این داشتن ابر کلید به مجموعه {AG} چه خصوصیتی را اضافه کنیم؟

پاسخ)

1. {ADG}
2. {ABDG}
3. {ABCDGHI}

نکته) مجموعه تمامی خصوصیتهای رابطه ابرکلید بدیهی است.

نکته) در طراحی هدف ما پیدا کردن کلید کمینه است. یعنی کلیدی که خصوصیت اضافه نداشته باشد.

سوال) نام ابرکلید کمینه چه بود؟

پاسخ) کلید کاندید Candidate Key

تعریف کلید کاندید) اگر R یک رابطه، α زیرمجموعه‌ای از خصوصیتهای R باشد. α کلید کاندید است اگر و فقط اگر

1. α^+ شامل تمامی خصوصیت‌های R باشد.
2. همه اعضای α مستقل باشند. هیچ عضوی از α قابل استنتاج از روی بقیه اعضا نباشد.

نکته) شرط استقلال به این معناست که اگر هر کدام از اعضا را حذف کنیم یکی از خصوصیت‌ها از α^+ حذف خواهد شد.

سوال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} کدام مورد کلید کاندید است؟

1. {AG}
2. {ADG}
3. {ABDG}
4. {ABCDGHI}

پاسخ) {AG} کلید کاندید نیست. زیرا $\{AG\}^+=\{ABCGHI\}$. در بستار D غایب است. پس نمی‌تواند کلید کاندید باشد. این مجموعه حتی ابر کلید هم نیست.

{ADG} هست زیرا $\{AG\}^+=\{ABCGHI\}$. و با اضافه کردن D همه خصوصیت‌ها را در بستار خواهیم داشت. دقت کنید هیچ کدام از اعضا نباید اضافه باشد. اگر A را حذف کنیم خواهیم داشت $\{DG\}^+=\{DG\}$، پس A اضافه نیست. اگر G را حذف کنیم $\{AD\}^+=\{ABCDH\}$ پس G اضافه نیست.

{ABDG} کلید کاندید نیست. زیرا B اضافه است. داریم A→B پس B وابسته به A است. دقت کنید چون $\{ABDG\}^+=\{ABCDGHI\}$ پس ابر کلید است.

{ABCDGHI} کلید کاندید نیست. دقت کنید چون این مجموعه خودش همه خصوصیت‌ها را دارد بدون نیاز به FD ابرکلید است. اصطلاحا ابرکلید بدیهی است.

4.5 پیدا کردن کلید کاندید با FD

نکته) خصوصیتی که در سمت راست هیچ FD نیست، یعنی هیچ خصوصیت دیگری نمی‌تواند آن را تولید کند. پس حتما باید در کلید کاندید وجود داشته باشد.

تعریف خصوصیت جنسیس) به خصوصیتی که در سمت راست هیچ FD نیست، خصوصیت جنسیس یا اصیل می‌گوییم.

مثال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} D سمت راست هیچ FD نیست پس جنسیس است. A سمت راست هیچ FD نیست پس جنسیس است. G سمت راست هیچ FD نیست پس جنسیس است. این سه خصوصیت یعنی A, D, G تنها جنسیس‌ها هستند.

تعریف خصوصیت غیر مولد) خصیصه‌ای که در سمت چپ هیچ FD نباشد.

مثال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→ , B→H} H و I سمت چپ هیچ FD نیستند پس غیر مولد هستند.

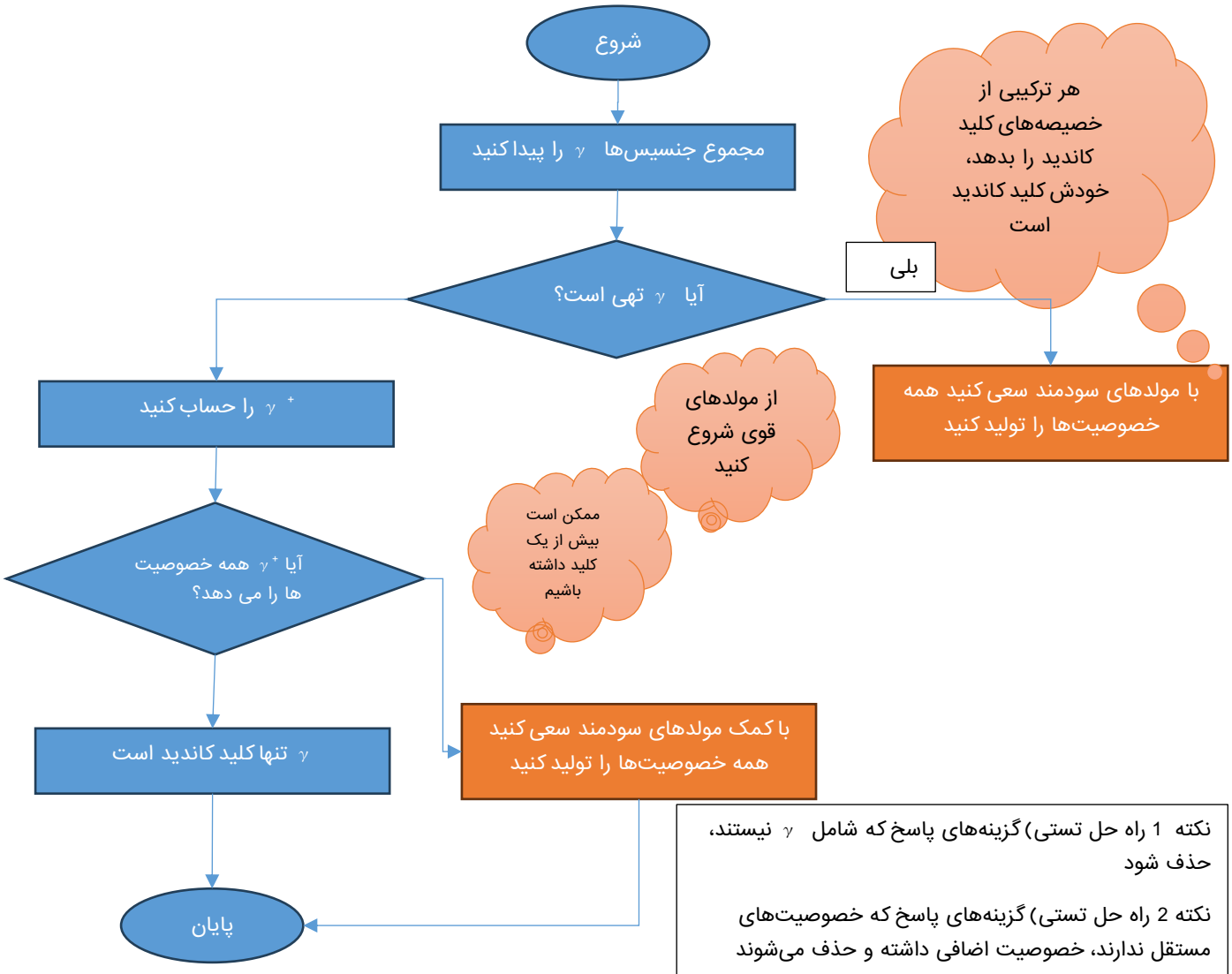
تعریف خصوصیت مولد) خصوصیتی که غیرمولد نباشد. یا در سمت چپ رابطه ظاهر شده باشد.

مثال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H}. خصوصیتهای A, B, C, D, G مولد هستند. دقت کنید برای مولد بودن کافی است که یا جنسیس باشید مثل A, D, G یا سمت راست یک FD باشید مثل A, B, C, G

تعریف خصوصیت مولد قوی‌تر) خصوصیت که در سمت چپ FD های بیشتر ظاهر شده باشد، یا سمت راست شلوغ‌تری داشته باشد مولد قوی تری است.

مثال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H}. مثلا بر اساس FD می‌توان گفت که A مولد قوی است زیر در سمت راست دو FD ظاهر شده است.

خصوصیت‌های مستقل) خصوصیت‌هایی که از هم قابل استنتاج نباشد مستقل هستند. مثلا از A می‌توان به B رسید پس مستقل نیستند. اما از G نمی‌توان به B رسید پس مستقل هستند.



مثال) برای R(A,B,C,D,G,H,I) مجموعه FD های F={A→B, A→C, CG→H, CG→I, B→H} کلید یا کلیدهای کاندید را به دست آورید.

پاسخ) اول باید جنسیس‌ها را پیدا کنیم. A, D, G در سمت راست هیچ FD نیستند پس جنسیس هستند.

دوم باید بستار مجموعه جنسیس‌ها را حساب کنید.

Result = {ADG}

A→B => result = {ABDG}

, B→H => result ={ABDGH},

A→C => result ={ABCDGH},

CG→I => result={ABCDGHI}

مجموعه جنسیس‌ها همه خصوصیتها را تولید کرد. پس ADG تنها کلید کاندید است. پایان.

نکته) در این مثال جنسیس داشتیم و مجموعه جنسیس به تنهایی قدرت تولید همه را داشتند.

مثال) برای R={A, B, C, D, E} و F={CE → B, A→D, B→C, D→E} کلید کاندید چیست؟

پاسخ) اول) دنبال جنسیس‌ها می‌گردیم. A سمت راست هیچ FD نیست. پس جنسیس است.

دوم) بستار مجموعه جنسیس را حساب می‌کنیم.

خود اعضای مجموعه در بستار است پس result = {A}

A→D => result ={A,D}

D→E => result ={A,D,E}

بیش از این تولید نمی‌شود. پس باید از مولدهای سودمند کمک بگیریم.

سوم) در بین مولدها B و C در result نیست. پس کمک گرفتن از آنها سودمند است. B از C قوی تر است. چون به تنهایی قدرت تولید دارد. با اضافه کردن B خواهیم داشت

Result = {A, B, D, E}

B→C => result ={A, B, C, D, E}

همه خصوصیتها را تولید کردیم پس AB کلید کاندید است. دقت کنید که A, B مستقل هستند. کار نباید متوقف شود. ممکن است کلیدهای کاندید دیگر هم داشته باشیم.

چهارم) ادعا می‌کنیم AC کلید کاندید است. زیرا

چهارم-اول) A و C مستقل هستند. به هنگام محاسبه بستار A دیدیم که به C نمی‌رسیم. یعنی C مستقل از A است. از طرف دیگر A جنسیس است و هیچ مولدی از جمله C نمی‌تواند آن را تولید کند، پس A هم مستقل از C است.

چهارم-دوم) A-->D, D-->E, Trans => A-->E CE-->B Pseudo => AC-->B AC-->A Union --> AC -->AB

چون AB کلید کاندید است و با استفاده از چهارم-اول و چهارم-دوم می‌توان نتیجه گرفت که AC هم کلید کاندید است.

پنجم) به یک روش دیگر هم می‌توان نشان داد AC کلید کاندید است. دیدیم که A → ADE برای رسیدن به همه خصوصیت‌ها باید از مولدهای سودمند استفاده کنیم، یعنی B و C. در سوم از B استفاده کردیم، حالا از C کمک می گیریم.

Result = {A, C, D, E}

CE→ B => result ={A, B, C, D, E}

پس AC کلید کاندید است. چون به غیر از B و C مولد سودمند دیگری برای کمک نداریم. با خیال راحت می‌توانیم جستجو برای کلیدهای بیشتر را متوقف کرده و نتیجه بگیریم که AB و AC تنها کلیدهای کاندید ماست.

نکته) در این مثال جنسیس داریم ولی چون به تنهایی همه خصوصیت‌ها را تولید نمی‌کرد مجبور شدیم از مولدهای سودمند کمک بگیریم.

مثال) برای R={X, Y, Z, W, P} و F={Y→ZW, X→YP, W→XZ} کلیدهای کاندید را پیدا کنید.

راه حل اول)

اول) همه خصوصیت‌ها سمت راست ظاهر شده‌اند پس جنسیس نداریم!😔😔😔

دوم) با از مولدها شروع کنیم. سه مولد X, Y, W هم قدرت داریم. از X شروع می‌کنیم. باید بستار X را حساب کنیم.

Result={X}

X→YP => result ={X, Y, P}

Y→ZW => result={X, Y, Z, W, P}

به همه خصوصیت‌ها رسیدیم پس X کلید کاندید است. نباید کار را متوقف کنیم

سوم) اینبار از Y شروع می‌کنیم.

Result = {Y}

Y→ZW => result ={Y, Z, W}

W→XZ => result = {X, Y, Z, P}

X→YP => result = {X, Y, Z, W, P}

به همه خصوصیت‌ها رسیدیم پس Y به تنهایی می‌تواند کلید کاندید باشد.

چهارم) اینبار از W شروع می‌کنیم.

Result = {W}

W→XZ => result={X, Z, W}

X→YP => result = {X, Y, Z, W, P}

به همه خصوصت‌ها رسیدیم.

پنجم) چون هیچ مولد دیگری نداریم می‌توان جستجو را متوقف کرد. کلید بیشتری نداریم. سه کلید کاندید داریم X و Y و Z.

نکته) در این مثال جنسیس نداشتیم با استفاده از خصوصیت‌های مولد کلید کاندید را پیدا کردیم.

نکته) وقتی اولین کلید کاندید را پیدا کردیم به جای بررسی دیگر مولدها می‌توان از تکنیک تولید کلید کاندید استفاد کرد.

مثال) برای R={X, Y, Z, W, P} و F={Y→ZW, X→YP, W→XZ} کلیدهای کاندید را پیدا کنید.

راه حل دوم)

در راه حل اول دیدیم که X کلید کاندید است.

اول) Y را به تنهایی در نظر بگیرید. Y چون ترکیب نیست مستقل است. در زیر نشان می‌دهیم Y→X

Y→ZW Decomp => Y→Z , Y→W

Y→W , W→XZ Trans => Y→XZ

Y→XZ Decomp => Y→X

چون از Y به X رسیدیم و X کلید کاندید است پس Y هم کلید کاندید است.

دوم) W را به تنهایی در نظر بگیرید. W چون ترکیب نیست مستقل است. در زیر نشان می‌دهیم W→X

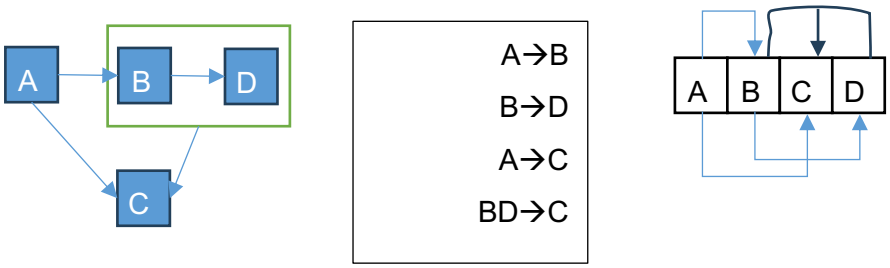
W→XZ Decomp. => W→X

چون از W به X رسیدیم و X کلید کاندید است پس W کلید کاندید است.

سوم) سه تا مولد داشتیم هر سه به تنهایی کلید کاندید شدند پایان جستجو.

4.5.1 نمایش‌های مختلف FD

برای نمایش FD راههای مختلفی داریم.



4.5.1.1.1.1 تست سراسری مهندسی کامپیوتر 1389

رابطه R(A,B, C, D, E) با وابستگی‌های تابعی زیر را در نظر بگیرید:

A→B, AB→CD, D→ABC

کدام گزینه کلید رابطه است؟

1) AD
2) AE
3) AB
4) ABD

پاسخ) گزینه 2

پاسخ تستی) جنسیس‌ها یعنی خصوصیت‌هایی که سمت راست هیچ FD نیست را پیدا کنید. E تنها جنسیس است. E باید در کلید کاندید باشد. تنها جواب گزینه 2 است.

نکته) صورت سوال با اینکه نگفته کلید کاندید منظور از کلید، کلید کاندید است.

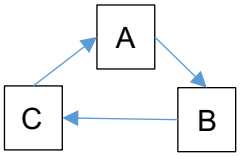
4.5.1.1.1.2 تست آزاد مهندسی کامپیوتر 1389

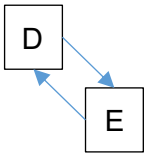
رابطه R(A, B, C, D, E) با وابستگی‌های تابعی زیر را در نظر بگیرید.

A→B, B→C, C→A, D→E, E→D

تعداد ابرکلیدها و کلیدهای کاندید رابطه به ترتیب از راست به چپ چند تاست؟

1) ابرکلید 8 کلید کاندید 2
2) ابرکلید 21 و کلید کاندید 6
3) ابرکلید 21 و کلید کاندید 8
4) ابرکلید 18 و کلید کاندید؟





پاسخ) گزینه 2. نمودار وابستگی را بکشید. دو حلقه دیده می‌شود. از هر حلقه باید یک و فقط یک نماینده در کلید کاندید باشد. اگر بیشتر باشد زاید است.

AD, AE

BD, BE

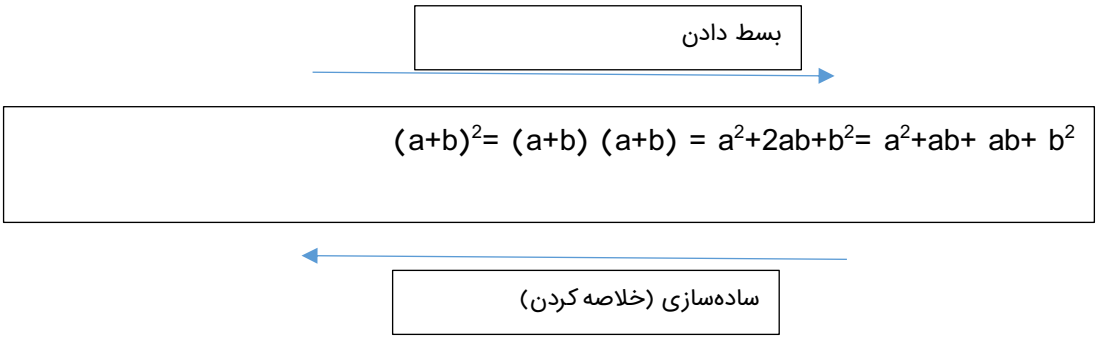
CD, CE

شش کلید کاندید خواهیم داشت.

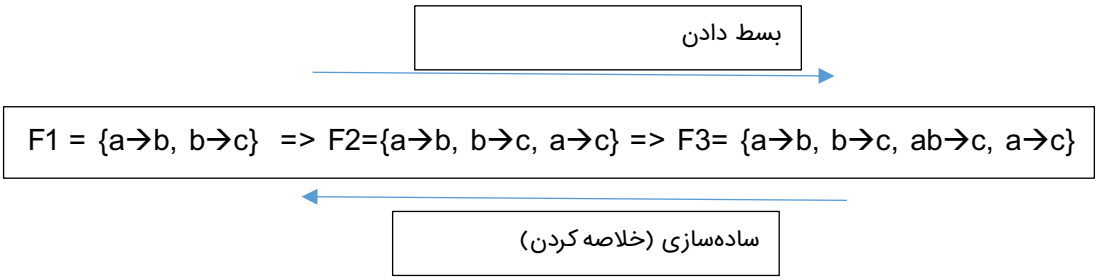
4.6 فرم کاهش‌ناپذیر irreducible

4.6.1 فرم کاهش ناپذیر چیست؟

یادآوری از جبر دبیرستان



عمل کاهش دادن reduce عکس عمل بسط است. کاهش دادن تقریبا همان مفهوم ساده کردن است.



تعریف مجموعه کاهش‌پذیر (دارای افزونگی):

یک مجموعه FD کاهش‌پذیر (دارای افزونگی) است اگر یک از دو شرط زیر را داشته باشد.

1. دارای اعضای تکراری باشد
2. برخی از اعضای آن قابل استنتاج از برخی از اعضای دیگر باشد

مثال) F2 ={a→b, b→c, a→c} کاهش‌پذیر است؟

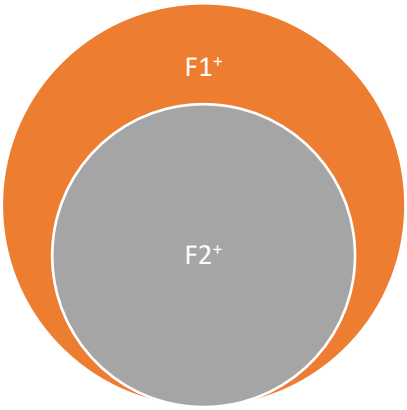
تعریف مجموعه های کاهش‌ناپذیر (فاقد افزونگی):

یک مجموعه FD کاهش‌ناپذیر است اگر دو شرط زیر را داشته باشد.

1. فاقد عضو تکراری باشد
2. اعضای آن از روی هم قابل استنتاج نباشند.

مثال) F1={a→b, b→c} کاهش ناپذیر است. به عبارت دیگر از این ساده‌تر نمی‌شود.

تعریف پوش) فرض کنید F1 و F2 دو مجموعه FD باشد. اگر هر یک از اعضای F2 قابل استنتاج از روی F1 باشد در این صورت گفته می‌شود که F1 یک پوش برای F2 است.



به راحتی می‌توان نشان داد که F1 پوش F2 است اگر $F2^+ \subset F1^+$.

مثال) F1={a→b, b→c} یک پوش برای F2={a→b, b→c, a→c} است.

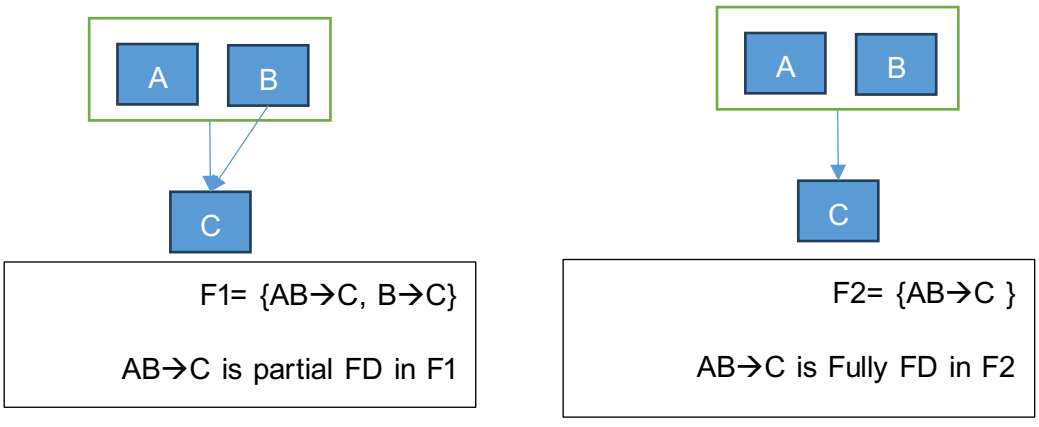
تعریف دو مجموعه معادل)دو مجموعه F1 و F2 معادل هستند اگر و فقط اگر $F2^+ \subset F1^+$ و $F1^+ \subset F2^+$ یا به عبارت دیگر $F1^+ = F2^+$.

نکته) دو مجموعه معادل هستند اگر هر FD که از اولی نتیجه می‌شود از دومی هم نتیجه بگیریم و بالعکس.

مثال) دو مجموعه F1={a→b, b→c} و F2={a→b, b→c, a→c} معادل هستند. پیشتر دیدیم که از F1 می‌توان F2 را نتیجه گرفت. پس F1 پوش F2 است. از طرف دیگر همه اعضای F1 در F2 است. پس F2 پوش F1 است. پس می‌توان F1 معادل F2 است.

نکته) برابر بودن با معادل بودن متفاوت است. مثلا F1={a→b, b→c} و F2={a→b, b→c, a→c} از دیدگاه نظریه مجموعه‌ها، اعضای یکسانی نداشته و برابر نیستند. ولی از دیدگاه FD معادل هستند.

4.6.2 بدست آوردن مجموعه کاهش‌ناپذیر



تعریف وابستگی تابعی کامل FFD Full Functional Dependency: وقتی که سمت راست وابستگی به کل سمت چپ داشته باشد نه بخشی از آن.

نکته: در برخی از منابع به FFD، Left-irreducible FD نیز گفته می‌شود. چون سمت چپ وابستگی کاهش‌پذیر (قابل ساده‌سازی) نیست.

سوال) چگونه می‌توان با حذف افزونگی، از مجموعه وابستگی‌ها F، مجموعه کاهش‌ناپذیر معادل آن را به دست آورد؟

پاسخ کوتاه) به سختی 😁😁😁😁😁

قضیه) یک مجموعه FD کاهش‌ناپذیر است اگر و فقط اگر

1. در سمت راست‌ها فقط یک خصیصه داشته باشیم.
2. FD ها همگی FFD باشند.
3. هیچ عضو مجموعه قابل استنتاج از بقیه اعضا نباشد. به بیان دیگر هیچ عضو مجموعه قابل حذف نباشد.

مثال) مجموعه F={A→B, B→C} کاهش‌ناپذیر است.

مثال) مجموعه F={A→B, B→C, A→C } کاهش‌پذیر است.

مثال) برای R (X Y Z W P) کاهش ناپذیر F= {XZ→YW, Y→WP, X→W, X→Z, YZ→P, W→P, X→P} را حساب کنید.

گام اول) به سراغ شرط اولی می‌رویم سمت راست همه FD ها باید تک خصیصه‌ای باشد. از قانون تجزیه Decomp استفاده کرده و F2 را بر اساس F1 می‌نویسیم.

F2= {XZ→Y, XZ→W, Y→W, Y→P, X→W, X→Z, YZ→P, W→P, X→P}

گام دوم) حالا سعی می‌کنیم تمام FD ها را FFD کنیم. دقت کنید FD ها با سمت چپ مرکب (چند خصیصه‌ای) ممکن است FFD نباشند.

XZ→Y, X→Z Pseudo => XX→Y => X→Y

پس به جای XZ→Y می‌توان X→Y نوشت.

XZ→W, X→W پس Z در XZ→W اضافه بود و بعد از حذف داریم X→Z

YZ→P, Y→P پس Z در YZ→P اضافه بوده و بعد از حذف داریم Y→P

با جایگذری‌های بالا F3 را بر اساس F2 می‌نویسیم.

F3= {X→Y, X→W, Y→W, Y→P, X→W, X→Z, Y→P, W→P, X→P}

گام سوم) تکراری‌های F3 را حذف می‌کنیم.

F4= {X→Y, X→W, Y→W, Y→P, X→Z, W→P, X→P}

گام چهارم) از قانون تراگذری استفاده می‌کنیم تا قابل استنتاج‌ها را حذف کنیم.

Y→W, W→P , trnXs. => Y→P پس می‌توان Y→P را حذف کرد.

X→W, W→P, trXns => X→P پس می‌توان X→P را حذف کرد.

X→Y , Y→W , trnXs => X→W پس می‌توان X→W را حذف کرد.

با حذف FD های بالا از F4 به F5 می‌رسیم.

F5={X→Y, Y→W, X→Z, W→P}

F5 کاهش ناپذیر است و قابل ساده کردن نیست. پایان.

5 نرمال‌سازی

یک از مراحل طراحی پایگاه‌داده نرمال‌سازی است.

نرمال‌سازی یک فرآیند مرحله به مرحله است.

نرمال‌سازی تقریباً معادل تجزیه‌کردن جدول‌ها است.

5.1 اهداف نرمال‌سازی

مثال) فرض کنید پایگاه‌داده آموزش را در یک جدول داشته باشیم.

Education (StID, CoID, Name, Grade, Major, Department)

StID	CoID	Name	Grade	Major	Department
شماره دانشجویی	شماره درس	نام دانشجو	نمره	رشته تحصیلی	دانشکده

مثلا Ali با شماره دانشجویی 77222 در درس شماره 222 نمره 12 گرفته‌است. Ali دانشجوی Comp در دانشکده ECE است.

Education

StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

Normalized

Student

StID	Name	Major
77222	Ali	Comp
78111	Reza	Math
76333	Sara	Phys
77444	Amin	Math
78666	Ali	Comp

GradeList

StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

DepList

Major	Department
Comp	ECE
Math	Science
Phys	Science

5.1.1 کاهش افزونگی‌های داده

وجود برخی از وابستگی‌های باعث بروز افزونگی می‌شود. مثلا در جدول Education زیر افزونگی‌ها با رنگ مشخص شده است.

مثلا Ali سه تا درس گرفته و به ازای هر سه درس شماره دانشجویی و نام وی در جدول تکرار شده است. یک دانشجوی لیسانس ممکن است تا 50 درس بگیرد. در این طراحی به ازای هر کدام از این درس‌ها شماره دانشجویی و نام وی تکرار می‌شود.

به عنوان یک نمونه دیگر به رشته Math در دانشکده Science دقت کنید. ما دو دانشجو به نام Reza و Amin داریم که دو دانشجوی Math در دانشکده Science هستند. هر کدام از این دانشجوها دو درس گرفته‌اند و در مجموع 4 بار این اطلاعات تکرار شده‌است. اگر این رشته 100 دانشجو داشته باشد و هر کدام 50 تا درس داشته باشند. 5000 بار این Math و Science در جدول تکرار می‌شود.

Education

StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

Normalized

Student

StID	Name	Major
77222	Ali	Comp
78111	Reza	Math
76333	Sara	Phys
77444	Amin	Math
78666	Ali	Comp

GradeList

StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

DepList

Major	Department
Comp	ECE
Math	Science
Phys	Science

5.1.2 مدل‌سازی دقیق‌تر

جدول Education اطلاعات چند موجودیت مختلف یعنی Student، GradeList ، DepList را در یک جدول نگهداری می‌کند. در یک طراحی دقیق‌تر بهتر است برای هر موجودیت یک جدول داشته باشیم.

5.1.3 کاهش هزینه اعمال برخی محدودیت‌های جامعیتی

در جدول Education این محدودیت جامعیتی را در نظر بگیرید که یک دانشجو نمی‌تواند دو نام داشته باشد. در این مثال هر موقع اطلاعات درسی یک دانشجو را می‌خواهیم درج کنیم باید کل جدول را جستجو کنیم تا مطمین شویم که این شماره دانشجویی پیش از این با نام دیگری درسی اخذ نکرده باشد.

مثال) فرض کنید (77222, 120, Sara, 15, DB, ECE) را می‌خواهیم در جدول درج کنیم. از نظر محدودیتهای جامعیتی این کار درست نیست زیرا 77222 دانشجویی با نام Ali است. برای جلوگیری از این درج باید اطلاعات این سطر را با تمامی سطرهای جدول مقایسه کنیم که این کار خیلی پر هزینه است. دقت کنید با توجه به این که برای هر درس دانشجو ما یک سطر در جدول Education داریم برای یک دانشگاه با 30000 دانشجو و میانگین 50 درس برای هر دانشجو تعداد سطرهای جدول Education حدود 150000 خواهد بود.

5.1.4 کاهش آنومالی

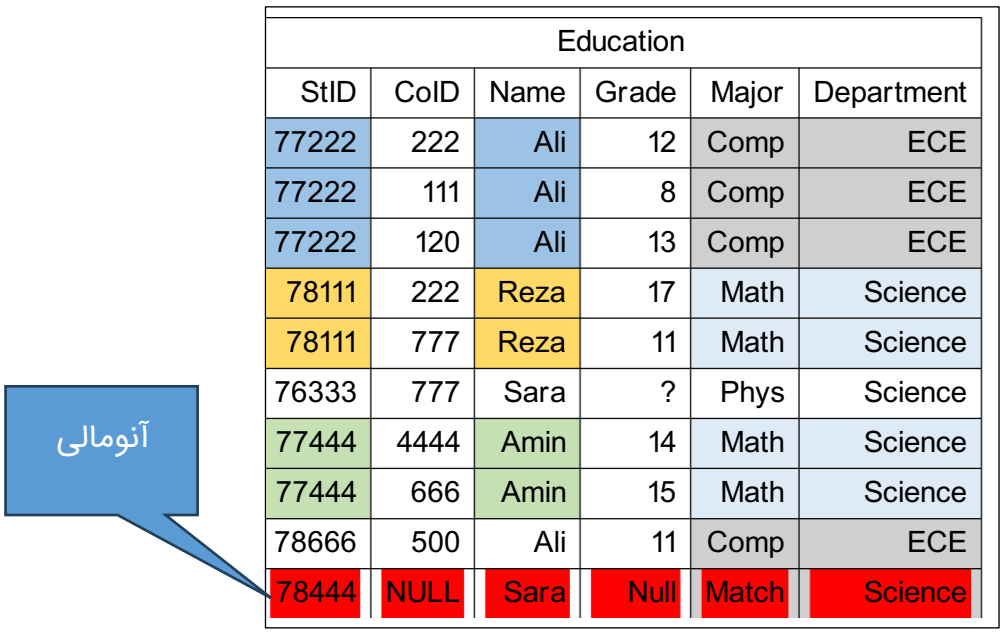
آنومالی Anomaly در لغت به معنای وضع غیرعادی، وضع نامطلوب و نابهنجاری و .. است.

به خاطر افزونگی جدول Education در عملیات‌های درج، حذف و بروزرسانی با آنومالی مواجه می‌شویم.

5.1.4.1 آنومالی به هنگام درج

فرض کنید یک دانشجو جدید ثبت‌نام کرده، بدیهی است که وی هنوز درسی اخذ نگرفته و نمره‌ای ندارد. اگر این دانشجو را در جدول Education درج کنیم مشکلات زیر را خواهیم داشت.

1. برای این دانشجو CoID و Grade مقدار ندارد و Null می‌گیرد.
2. CoID بخشی از کلید اصلی است و null شدن آن قانون دوم جامعیت را نقض می‌کند.



Education					
StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE
78444	NULL	Sara	Null	Match	Science

5.1.4.2 آنومالی به هنگام حذف

مثلا فرض کنید می‌خواهیم نمره Ali به شماره دانشجویی 78666 را حذف کنیم. با حذف نمره کل اطلاعات Ali حذف می‌شود. این آنومالی است.

5.1.4.3 آنومالی به هنگام بروزرسانی

فرض کنید می‌خواهیم اسم Amin به شماره دانشجویی 77444 را به Sadra تغییر دهیم. برای اینکار باید کل جدول پیمایش شود و به ازای تمامی درس‌های Amin به شماره دانشجویی 77444 نام را تغییر دهیم تا سازگاری پایگاه داده حفظ شود.

5.1.5 معایب نرمال‌سازی

5.1.5.1 افزایش افزونگی

نرمال‌سازی با تجزیه جدول افزونگی داده را کاهش می‌دهد. ولی چون همچنان ما باید این جدول‌ها را الحاق کنیم، باید ستونهای مشترک (تکراری) داشته باشیم. این ستونهای تکراری خود یک افزونگی است.

Education

StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

Normalized

Student

StID	Name	Major
77222	Ali	Comp
78111	Reza	Math
76333	Sara	Phys
77444	Amin	Math
78666	Ali	Comp

GradeList

StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

DepList

Major	Department
Comp	ECE
Math	Science
Phys	Science

5.1.5.2 سربار پردازشی

در جدول Education به راحتی می‌توانیم معدل دانشجویان یک دانشکده را حساب کنیم. ولی در فرم نرمال‌شده برای این پرس‌وجو باید سه جدول Student، GradeList و DepList با هم الحاق شوند. الحاق یکی از پرهزینه‌ترین عملیات‌ها در پایگاه داده است.

5.1.5.3 سربار طراحی

در حین نرمال‌سازی (تجزیه کردن) پایگاه‌داده انتخاب‌های زیادی دارد. انتخاب بهترین راه‌حل زمان طراحی را افزایش داده و نیاز به دانش و تجربه دارد. هر چه قدر پایگاه داده بزرگتر و پیچیده‌تر باشد.

5.2 نرمال فرم اول 1NF

نرمال فرم اول First Normal Form شروط زیر را دارد.

1. دارای حداقل یک کلید کاندید باشد
2. خصیصه مرکب نداشته باشیم.
3. خصیصه چند مقداری نداشته باشیم.

1NF

Education

StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

5.3 نرمال فرم دوم 2NF

نرمال فرم اول First Normal Form شروط زیر را دارد.

1. جدول باید در نرمال فرم اول باشد.
2. وابستگی بخشی Partial Dependency نداشته باشیم.

تعریف وابستگی بخشی: خصوصیتی فقط به بخشی از کلید وابسته باشد.

1NF

Education					
StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

افزونگی

2NF

Student			
StID	Name	Major	Department
77222	Ali	Comp	ECE
78111	Reza	Math	Science
76333	Sara	Phys	Science
77444	Amin	Math	Science
78666	Ali	Comp	ECE

GradeList		
StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

5.4 نرمال فرم سوم 3NF

نرمال فرم اول First Normal Form شروط زیر را دارد.

1. جدول باید در نرمال فرم دوم باشد.
2. وابستگی انتقالی Transitive Dependency نداشته باشیم.

تعریف وابستگی انتقالی: خصوصیتی به خصوصیت غیر کلیدی وابسته باشد.

1NF

Education					
StID	CoID	Name	Grade	Major	Department
77222	222	Ali	12	Comp	ECE
77222	111	Ali	8	Comp	ECE
77222	120	Ali	13	Comp	ECE
78111	222	Reza	17	Math	Science
78111	777	Reza	11	Math	Science
76333	777	Sara	?	Phys	Science
77444	4444	Amin	14	Math	Science
77444	666	Amin	15	Math	Science
78666	500	Ali	11	Comp	ECE

2NF

Student			
StID	Name	Major	Department
77222	Ali	Comp	ECE
78111	Reza	Math	Science
76333	Sara	Phys	Science
77444	Amin	Math	Science
78666	Ali	Comp	ECE

GradeList		
StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

3NF

Student		
StID	Name	Major
77222	Ali	Comp
78111	Reza	Math
76333	Sara	Phys
77444	Amin	Math
78666	Ali	Comp

GradeList		
StID	CoID	Grade
77222	222	12
77222	111	8
77222	120	13
78111	222	17
78111	777	11
76333	777	?
77444	4444	14
77444	666	15
78666	500	11

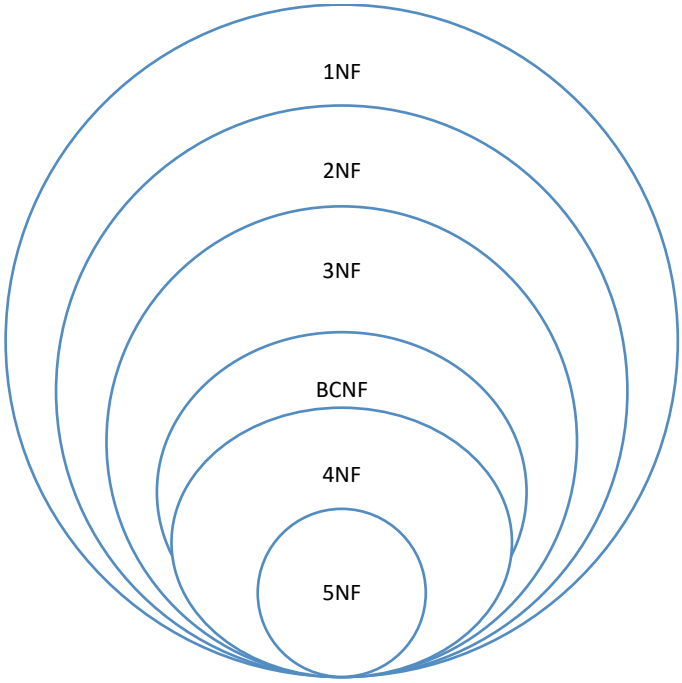
DepList	
Major	Department
Comp	ECE
Math	Science
Phys	Science

5.5 مراحل نرمال‌سازی

نرمال‌سازی مرحله به مرحله است.

هر جدول 3NF باید شرایط 1NF را داشته باشد چون 3NF زیر مجموعه 1NF است.

BCNF زیر مجموعه 3NF است. مرز BCNF و 4NF خیلی نزدیک بوده و در پروژه‌های عملی کمتر با جدولی از نوع BCNF روبرو می‌شویم.



تعریف فرم‌های نرمال با کمک انواع وابستگی		
	شرط	نکته
1NF	دارای حداقل یک کلید کاندید باشد. فاقد خصیصه مرکب باشد فاقد خصیصه‌های چندمقداری باشد.	
2NF	+فاقد وابستگی بخشی باشد	
3NF	+فاقد وابستگی انتقالی باشد	
تعریف وابستگی‌های		
بخشی	خصیصه غیرکلیدی→ بخشی از کلید	خصیصه غیرکلیدی در هیچ کلید کاندیدی ظاهر نشده‌است.
انتقالی	خصیصه غیرکلیدی→ خصیصه غیرکلیدی	

شکل کلی سوال)رابطه R با مجموعه وابستگی‌های F را داریم. مشخص کنید در چه نرمال فرمی است؟

گام اول)تشخیص همه‌ی کلیدها (سخت‌ترین)

گام دوم)شناسایی خصوصیت‌های غیرکلیدی

گام سوم)تشخص نوع وابستگی‌ها

مثال) برای رابطه A(X,Y,Z,P,Q,R) وابستگی‌های F={XYZ→QR, P→Q, P→R, P→X, Y→Z} را داریم.
مشخص کنید این رابطه در کدام نرمال فرم است؟

راه حل)

گام اول تشخیص کلیدها) بر اساس فلوچارت تشخیص کلید اول از همه باید خصوصیت‌های جنسیس را پیدا کنیم. به سمت راست FD ها نگاه کرده و هر خصوصیتی که سمت راست ظاهر شده خط می‌زنیم.

XYZ→QR => A(X,Y,Z,P,~~Q,R~~)

P→X, Y→Z => A(~~X~~,Y,~~Z~~,P,~~Q,R~~)

تنها Y و P باقی می‌ماند. پس این دو جنسیس هستند. حالا بستار {Y,P} را حساب می‌کنیم. باید به پوشش کامل برسیم.

RESULT = {Y,P}

Y→Z, P→X => RESULT ={X,Y,Z,P}

P→Q, P→R => RESULT={X,Y,Z,P,Q,R}

همه خصوصیت‌ها را تولید کردیم. پس YP تنها کلید کاندید ماست و لازم نیست دنبال کلیدهای بیشتر هم بگردیم. در این سوال گام اول خیلی کوتاه بود و از این نظر خوش‌شانس بودیم 😁😁

گام دوم) شناسایی خصیصه‌های غیر کلیدی

خصیصه‌های غیرکلیدی X, Z, Q, R

گام سوم) در وابستگی P→X الگوی خصیصه غیرکلیدی→ بخشی از کلید را داریم. این وابستگی بخشی است. پس 2NF نیست. پس 3NF و ... هم نخواهد بود. این رابطه در فرم 1NF است. پایان.

2.6 نرمال فرم BCNF

این نرمال فرم به نام پیشنهاددهنده آن یعنی Boyce/Codd معروف شده است.

با رسیدن به BCNF تمامی FD ها و افزونگی‌های ناشی از آنها حذف شده است. ممکن است افزونگی داشته باشیم ولی این افزونگی‌ها ناشی از وابستگی تابعی نیست بلکه از دیگر نوع وابستگی‌ها ناشی می‌شود.

2.6.1 تعریف اول BCNF

تعریف) یک رابطه BCNF است اگر و فقط اگر به ازای همه $\alpha \rightarrow \beta$ یکی از دو شرط زیر برقرار باشد.

اول) $\alpha \rightarrow \beta$ بدیهی باشد. یعنی $\beta \subset \alpha$

دوم) α ابرکلید باشد.

نکته)شرط اول در سوالات کنکور معمولاً برقرار نیست ما باید با پیداکردن کلیدهای شرط دوم را بررسی کنیم.

مثال)R=(A, B, C) و F={A→B, B→C} در BCNF است؟

راه حل)

گام اول) پیدا کردن کلید.

A جنسیس است. پس حتما جزیی از کلید کاندید است. به راحتی با A دو خصوصیت دیگر را هم می‌توان تولید کرد. پس A تنها کلیدکاندید است. و کار جستجوی برای کلیدهای بیشتر را متوقف می‌کنیم.

گام دوم)

حالا شرط دوم تعریف BCNF را بررسی می‌کنیم.

در A→B سمت چپ A کلید کاندید است. پس ابرکلید super key هم هست. در نتیجه شرط را نقض نمی‌کند.

در B→C سمت چپ B کلید نیست. پس ابرکلید هم نیست. در نتیجه شرط را نقض می‌کند. کافی است یک FD شرط را نقض کند. بلافاصله نتیجه می‌گیریم که BCNF نیست و سراغ دیگر FD ها نمی‌رویم.

مثال)R=(A, B, C) و F={A→B, B→C} در چه نرمال فرمی است؟

دیدیم که تنها کلید کاندید A است. وابستگی بخشی نداریم پس 2NF است. در B→C الگوی خصیصه غیرکلیدی → خصیصه غیرکلیدی را داریم. این وابستگی انتقالی است پس 3NFنیست. از این که 3NF نیست هم می‌توان نتیجه گرفت که BCNF نیست زیرا BCNF ⊂ 3NF

مثال)R (A, B, C, D) وF={D→A, AC→D} آیا BCNF است؟

گام اول پیدا کردن کلید)

B و C چون سمت راست هیچ FD ظاهر نشده‌اند. پس عضو کلید کاندید هستند. در اینجا لازم نیست راه حل را ادامه دهیم. همه ابرکلیدها باید شامل BC باشد. پس در D→A سمت چپ سوپر کلید نیست. همین مثال نقض کافی است که نتیجه بگیریم BCNF نیست. پایان.

مثال)R (A, B, C, D) وF={D→A, AC→D} در چه نرمال فرمی است؟

B و C جنسیس هستند. ولی چون به تنهایی نمی‌توانند همه خصیصه‌ها را تولید کنند باید از مولدها کمک گرفت.

از A کمک بگیریم $\{A, B, C\}^+=\{A, B, C, D\}$ پس ABC کلید کاندید است. شاید کلید کاندید دیگر هم داشته باشیم.

D→A پس BCD→ABC . از آنجایی که قبلا نشان دادیم که ABC کلید کاندید است پس BCD هم کلید کاندید است.

دو تا کلید کاندید داریم ABC و BCD داریم. خصیصه‌های کلیدی عبارتند از A و B و C و D. و هیچ خصیصه غیرکلیدی نداریم. پس وابستگی بخشی و انتقال غیرممکن است. زیرا نمی‌توان الگوی خصیصه غیرکلیدی→ * را پیدا کرد. وقتی وابستگی بخشی و انتقالی نداشته باشیم 3NF است.

هشدار)D→A مصداق وابستگی بخشی نیست. درست است که D بخشی از کلید است ولی سمت راست A خصیصه غیر کلیدی نیست. دقت کنید که A عضوی از کلیدABC می‌باشد.

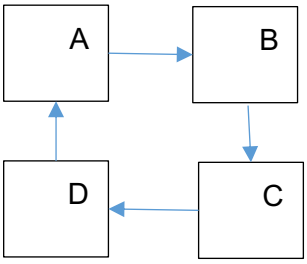
به همین شکل AC→D مصداق وابستگی بخشی نیست. درست است که AC بخشی از کلید است ولی سمت راست D خصیصه غیر کلیدی نست.

مثال)R(A, B, C, D) و F={A→B, C→D, B→C, D→A} آیا BCNF است؟

گام اول)پیدا کردن کلید.

همه خصیصه‌ها سمت راست هستند پس جنسیس نداریم. 😟😟

همه خصیصه‌ها سمت چپ هستند پس همه مولد هستند با قدرت تولید مساوی. پس همه‌ی راه‌های میانبر بسته است.



نمودار FD یک حلقه شده است. از هر حلقه یک نماینده در کلید کاندید کافی است. پس ما چهار کلید کاندید خواهیم داشت. A و B و C و D

به روابط که نگاه می‌کنیم سمت چپ همه آنها یک سوپر کلید داریم پس رابطه BCNF است. پایان.

تعریف جدول تمام کلید) اگر در یک جدول ترکیب همه خصیصه‌ها با یکدیگر کلید کاندید را تشکیل دهد، به آن جدول تمام کلید گفته می‌شود.

نکته) اگر جدولی (رابطه‌ای) تمام کلید باشد قطعاً BCNF است و فاقد وابستگی بخشی، انتقالی و معکوس.

2.6.1.1.1.1 تست سراسری IT ۱۳۸۵ طراحی BCNF با تمام کلید

رابطه xyz به شکل روبرو در نظر بگیرید. این رابطه در چه سطحی از نرمال بودن است؟

1NF(۱ 2NF(۲ 3NF(۳ 4NF(۴

X	Y	Z
X1	Y1	Z2
X2	Y2	Z1
X2	Y1	Z1
X1	Y1	Z1

راه حل) هیچ کدام از خصیصه‌ها یعنی Xو Y و Z به تنهایی تمایز بین سطرها را ایجاد نکرده و نمی‌تواند کلید باشد. ترکیب‌های دوتایی یعنی XYو XZ و YZ هم به همین شکل قدرت تمامیز بین سطرها را ندارند. کلید باید شامل تمام خصوصیت‌ها باشد. پس جدول تمام کلید بوده و جدول تمام کلید BCNF است.

دیدیم که BCNF⊂ 3NF است. یعنی 3NF شروط ضعیفتری دارد. با اضافه کردن یک شرط به BCNF می‌توان شروط آن را ضعیف‌تر کرده و به تعریف جدیدی برای 3NF رسید.

2.6.2 تعریف 3NF بر اساس تعریف BCNF

تعریف) یک رابطه 3NF است اگر و فقط اگر به ازای همه $\alpha \rightarrow \beta$ یکی از سه شرط زیر برقرار باشد.

اول) $\alpha \rightarrow \beta$ بدیهی باشد. یعنی $\beta \subset \alpha$

دوم) α ابرکلید باشد.

سوم) هر خصیصه $\beta - \alpha$ عضو یک کلید کاندید باشد.

نکته) دقت کنید که دو شرط اول عیناً از BCNF آمده است. و شرط سوم چون با OR منطقی آمده است موجب ضعیف‌تر شدن شرط‌ها می‌شود.

نکته)هر خصیصه $\beta - \alpha$ می‌تواند عضو یک کلید کاندید باشد. به عبارت دیگر اگر دو خصیصه A و B در $\beta - \alpha$ باشد لازم نیست که هر دو عضوی یک کلید کاندید باشد.

مثال) با استفاده از تعریف جدید 3NF نشان دهید R (A, B, C, D) و F={D→A, AC→D} در فرم 3NF است.

گام اول)پیشتر نشان دادیم که تنها دو کلید کاندید داریم ABC و DBC. چون قبلاً نشان داده شد که BCNF نیست پس سراغ شرط دوم نمی‌رویم و شرط سوم را چک می‌کنیم.

D→A

α = D , β = A => $\beta - \alpha$ = A

A عضوی از کلید ABC پس برای این FD شرط سوم برقرار است.

AC→D

α = AC , β = D => $\beta - \alpha$ = D

D عضوی از کلید DBC پس برای این FD شرط سوم برقرار است.

برای هر دو FD شرط سوم برقرار است. پس می‌توان گفت که 3NF است.

نکته) منظور از $\beta - \alpha$ تفاضل مجموعه‌ای است.

مثال)R(A, B, C, D) و F={AB → CD, CD→AB} آیا BCNF است؟

گام اول)پیداکردن کلید کاندید. همه خصوصیت‌ها سمت راست هستند پس جنسیس نداریم. همه سمت چپ هستن و مولد هستند.

خوشبختانه با کشید نمودار FD می‌بینیم حلقه داریم. یک نمایند از حلقه در کلید کاندید کافی است. پس ما دو کلید کاندید داریم AB و CD.

شرط دوم را چک می‌کنیم.

AB → CD سمت چپ سوپرکلید است. پس شرط دوم را دارد.

CD→AB سمت چپ سوپر کلید است. پس شرط دوم را دارد.

پس رابطه در فرم BCNF می‌باشد. پایان.

2.6.3 تعریف دوم BCNF

تعریف دوم BCNF) رابطه R در BCNF است اگر و فقط اگر برای هر $\alpha \rightarrow \beta$ نابدیهی (مطرح)(مهم) (nontrivial) و کاهش‌ناپذیر α کلید کاندید رابطه باشد.

2.6.3.1.1.1 تست سراسری مهندسی کامپیوتر 1394

رابطه R(A,B,C,D,E) با F={BC→A, A→D, D→C, D→E} کدام یک از وابستگی‌های تابعی زیر BCNF را نقض نمی‌کند؟

۱) BC→A ۲) D→C ۳) A→D ۴) D→E

پاسخ) گزینه ۱

راه حل تستی)

1. طبق تعریفBCNF، همه FD نابدیهی باید سمت چپ سوپر کلید باشند.
2. تنها B سمت راست ظاهر نشده است. تنها جنسیس است و حتما باید عضوی از کلید باشد. گزینه ۱ تا ۳ ، سمت چپ B نداشته و نمی‌توانند سوپر کلید باشد. حذف می‌شوند و فقط گزینه ۱ باقی می‌ماند. پایان

راه حل تشریحی)

در گام اول باید کلیدها را پیدا کنیم. B جنسیس است. بستار B برابر با B بوده و باید کمک بگیریم. از C کمک می گیریم.

Result={B, C} , BC→A , A→D, D→E => result = {A,B,C,D,E}

پس BC کلید است. برای حل این تست پیدا کردن همین کلید کافی است. طبق تعریف چون سمت چپ BC →A سوپر کلید است. می‌توانیم گزینه ۱ را انتخاب کنیم.

دو تا کلید بعدی هم AB و BD است. که می‌توان دید همه خصیصه‌ها را تولید می‌کند.

2.6.4 شروط BCNF بودن 3NF

نکته)همانطور که گفتیم هر 3NF لزوماً BCNF نیست. ولی در عمل معمولاً همه 3NF ها BCNF هم هستند. هر BCNF همیشه 3NF است.

نکته 1) اگر 3NF یکی از شروط زیر را داشته باشد. قطعاً BCNF هم هست. بهتر است این شروط را به همین ترتیب بررسی کنید.

اول)فقط یک کلید کاندید داشته باشد.

دوم)یکی از کلیدهای کاندید آن ساده باشد.

سوم)دو تا از کلیدهای کاندید مرکب آن خصیصه مشترک نداشته باشند.

مثال)R(A, B, C, D) و F={AB → CD, CD→AB} 3NF است با استفاده از نکته 1 بگویید که BCNF هست یا خیر؟ کلیدها عبارتند از AB و CD
شرط اول برقرار نیست. بیش از یک کلید داریم.
شرط دوم برقرار نیست. هر دو کلید مرکب هستند.
شرط سوم برقرار است. دو کلید خصیصه مشترک ندارند. پس می‌توان نتیجه گرفت که BCNF است. پایان.

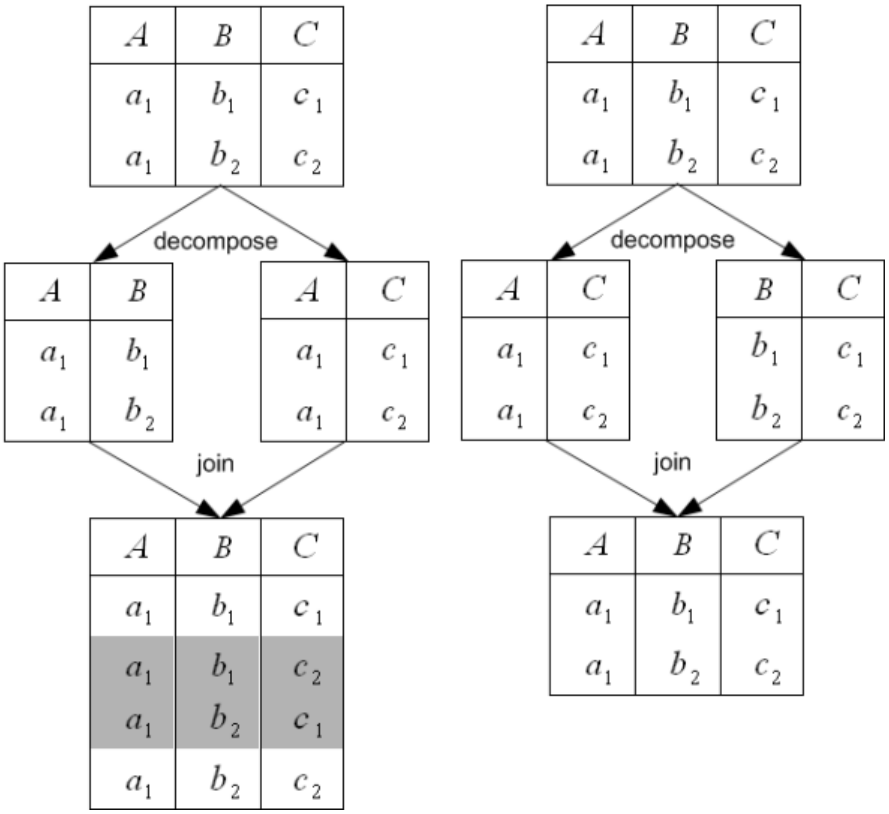
مثال) دیدیم که R (A, B, C, D) و F={D→A, AC→D} با کلید کاندید ABC و DBC در فرم 3NF است. با استفاده از نکته 1 درباره BCNF بودن تصمیم بگیرید.
شرط اول برقرار نیست. بیش از یک کلید داریم
شرط دوم برقرار نیست. هر دو کلید مرکب هستند.
شرط سوم برقرار نیست. BC در هر دو کلید مشترک است. پس نمی‌توان گفت قطعاً BCNF است. دقت کنید طبق این نکته نمی‌توان گفت قطعاً نیست. پایان.

2.7 مشکلات نرمال‌سازی

ابزار اصلی برای نرمال‌تر کردن رابطه (جدول)ها تجزیه (Decomposition) است. به هنگام تجزیه ممکن است با مشکلاتی روبرو شویم.

2.7.1 مشکل گم‌شدگی loss

تعریف) تعریف دقیق گم‌شدگی. اگر R را به دو رابطه‌ی R1 و R2 تجزیه کرده و R <> R1 join R2 گفته می‌شود که اطلاعات گم‌شده است.



شکل 1) مثال مقایسه تجزیه با گم‌شدگی با تجزیه بدون گم‌شدگی

سوال) در شکل 1 سمت چپ گم‌شدگی داریم. همانطور که می‌بینید سطرهای جدول اصلی گم نشده‌است بلکه حتی تعدادی سطر هم اضافه شده‌است. پس چرا می‌گوییم اطلاعات گم شده‌است؟!

2.7.2 مشکل دوم از دست رفتن وابستگی‌ها

مثال) رابطه R(a, b, c, d, e) با مجموعه وابستگی‌های F={a→d, b→c, e→ab, c→e}

تجزیه اول) R1(a, d, e) R2(a, b, c)

F1={b→c} و F2={a→d, e→ab} . وابستگی c→e از دست رفته‌است.

تجزیه دوم) R1(a, d) R2(a, b, c, e)

F1={a→d} و F2={b→c, e→ab, c→e} و هیچ وابستگی از بین نرفته است.

تعریف) تعریف دقیق حفظ وابستگی. اگر R1 و R2 حاصل تجزیه‌ی رابطه R باشند و F مجموعه وابستگی‌های R باشد. با تجزیه R به دو رابطه مجموعه‌ای ازFD ها در R1 و R2 باقی بمانند که آن را G می‌نامیم. اگر $G^+<>F^+$ باشد گفته می‌شود که مشکل از دست رفتن وابستگی داریم.

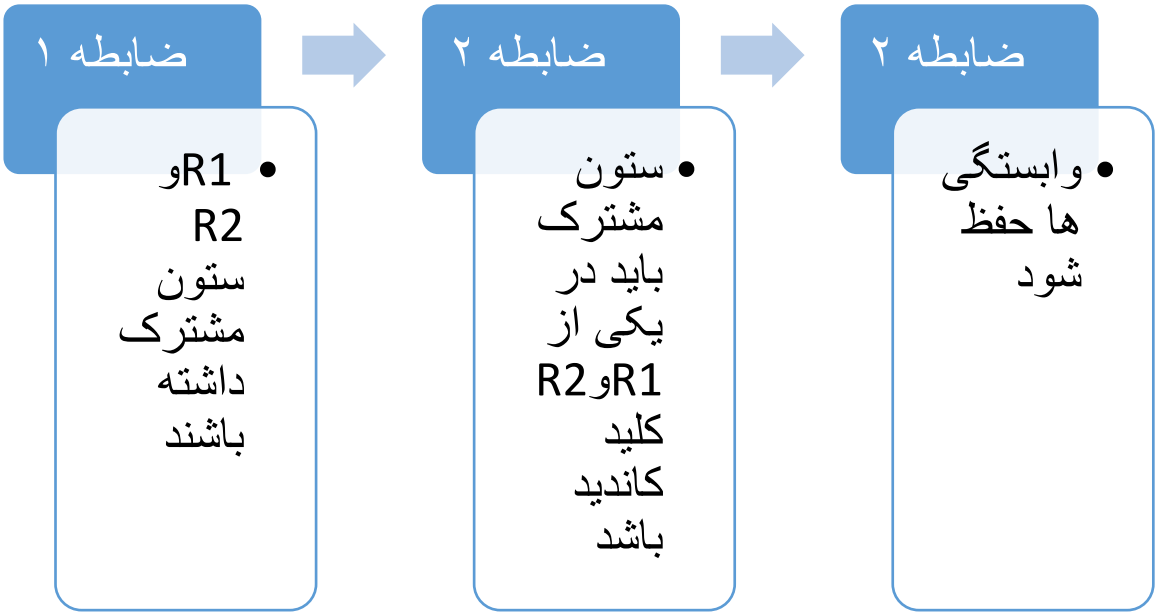
مثال) R(ABCDE) و F={AB→CD, C→D, D→E} را تجزیه می‌کنیم به D= {ABC, CD, DE}

R1(ABC) contains AB→C, R2(CD) contains C→D, R3(DE) contains D→E

ممکن است فکر کنید که AB→CD از دست رفته‌است. ولی با کمی دقت و استفاده از قواعد آرمسترانگ می‌توان به آن هم رسید. AB→C, C→D, Trans => AB→D, AB→C, Union => AB→CD

تعریف)تجزیه خوب. تجزیه خوب تجزیه‌ای است که مشکل نداشته باشد. این مشکلات عبارتند از ۱. گم‌شدگی ۲. از دست رفتن وابستگی‌ها ۳. از دست رفتن خصوصیت‌ها ۴. حذف تاپلها

2.8 ضوابط ریسانن Rissanen برای تجزیه خوب



توصیه) بهتر است این ضوابط به ترتیب بررسی شود.

قضیه هیث HEATH) رابطه R(A, B, C) که در آن A، B و C سه مجموعه از صفات هستند، مفروض است. اگر A→B، آنگاه می‌توان R را به دو رابطه R1(A, B) و R2(A, C) تجزیه کرد و این تجزیه گم‌شدگی ندارد.

نکته) در قضیه هیث تکرار شدن A در R1 و R2 یعنی ستون مشترک. یعنی ضابطه ۱ ریسانن. از طرفی دیگر A →B یعنی که A کلید R1 است. یعنی ضابطه ۲ ریسانن. به بیان دیگر قضیه هیث همان ضابطه ۱ و ۲ ریسانن است.

نکته)ضابطه ۱ و ۲ ریسانن بدون گم‌شدگی بودن تجزیه را تضمین می‌کند.

سوال) رابطه R(A, B, C) که در آن A، B و C سه مجموعه از صفات هستند، مفروض است. اگر A→B و B→ C در این رابطه برقرار باشد. کدام یک از تجزیه‌های زیر درست است؟
تجزیه اول) R1(A, B) و R2(A, C) تجزیه دوم) R3(A, B) و R4(B, C)

توصیه) رابطه R(A, B, C) که در آن A، B و C سه مجموعه از صفات هستند، مفروض است. اگر A→B و B→ C در این رابطه برقرار باشد. به دوشکل تجزیه را می‌توان انجام داد. که تجزیه دوم بهتر است.
تجزیه اول) R1(A, B) و R2(A, C) تجزیه دوم) R3(A, B) و R4(B, C)

2.8.1.1.1.1 تست سراسری مهندسی کامپیوتر 1392
رابطه R(A, B, C, D) و وابستگی‌های تابعی F={A→B, B→C, C→D} را در نظر بگیرید. کدام گزینه بدون گم‌شدگی نیست؟

۱) R1(A, B) , R2(B,C), R3(C, D) ۲) R1(A,B) , R2(A,C) , R3(A,D)

۳) R1(A, D), R2(B, D) , R3(C,D) ۴) هیچ کدام

پاسخ) گزینه ۳.
برای گم شدگی از قضیه هیث استفاده می‌کنیم. بین زیرمجموعه ها باید ستون مشترک داشته باشیم که در حداقل در یکی از روابط کلید باشد.

گزینه ۱) بین R1 و R2 ستون مشترک B است. B به خاطر B→C در R2 کلید است. هیث برقرار است.
بین R2و R3 ستون مشترک C است. C به خاطر C→D در R3 کلید است. شرط هیث برقرار است. بین R1 و R2 ستون مشترک نداشته و شرط هیث برقرار نیست. ولی مهم نیست. اگر بین دو تا از سه تا دوتایی هیث برقرار باشد کافی است.

گزینه ۲)بین R1 و R2 ستون مشترک A که در R1 کلید است. شرط هیث برقرار است.

بین R3 و R1 ستون مشترک A که در R1 کلید است. شرط هیث برقرار است.
بین R2 و R3 ستون مشترک A که در هی کدام کلید نیست. شرط هیث برقرار نیست. ولی مهم نیست اگر بین دو تا از سه تا دو تایی هیث برقرار باشد کافی است که بگوییم

گزینه ۳) بین R1 و R2 ستون مشترک D که هیچ کجا کلید نیست.

بین R2 و R3 ستون مشترک D که هیچ کجا کلید نیست.

بین R1 و R3 ستون مشترک D که هیچ کجا کلید نیست.

چون در تمامی ترکیب‌های R1R2 و R2R3 و R1R3 قضیه هیث نقض شده، می‌توان گفت که گم‌شدگی داریم.

نکته) در این سوالات نداشتن گم‌شدگی در دو تا از دوتایی کافی است که نتیجه گرفته شود اصلاً گم‌شدگی نداریم.

پایگاه داده یک نوع سامانه اطلاعاتی است.

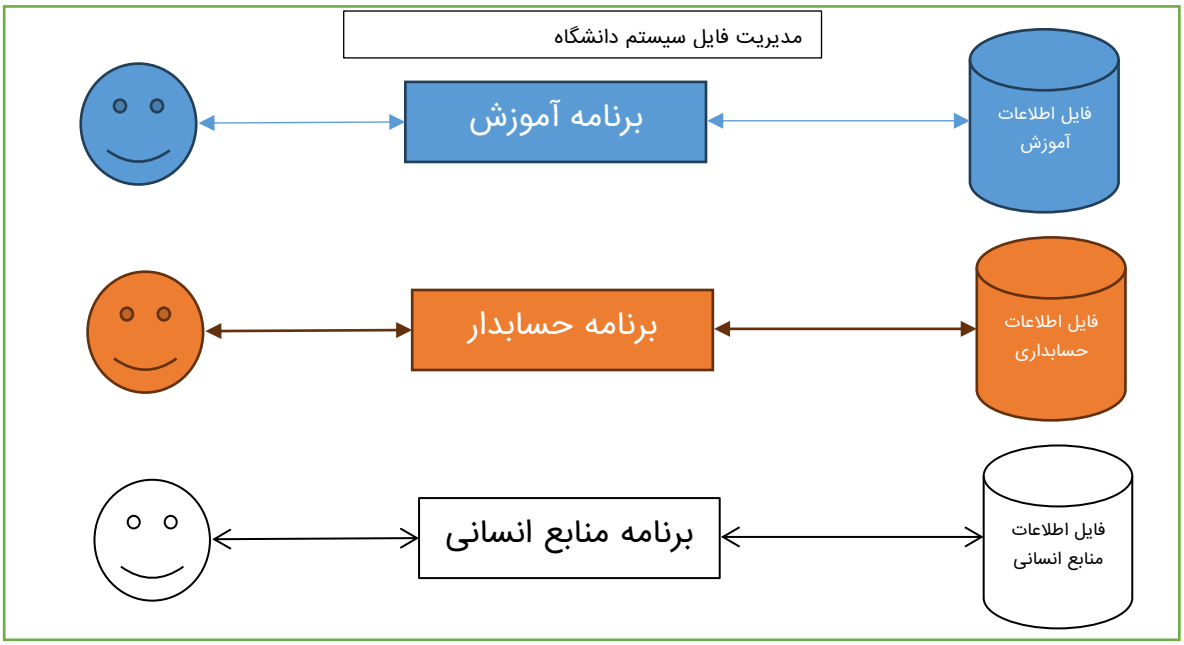
1 سامانه‌های اطلاعاتی

سامانه‌های (سیستم) که اطلاعات یک سازمان (شرکت، دانشگاه، اداره...)را مدیریت می‌کند.

سامانه‌های اطلاعاتی Information System قبل از اختراع کامپیوتر با استفاده از کاغذ، پوشه، پرونده و بایگانی پیاده‌سازی می‌شد. با ظهور کامپیوتر سامانه‌های اطلاعاتی کامپیوتری شد. اولین نسل این سامانه‌های اطلاعاتی کامپیوتری مدیریت فایل سیستمی نامیده می‌شد.

مدیریت فایل سیستمی = File Processing System = FS = فایلینگ

2 مدیریت فایل سیستمی



2.1 ویژگی مدیریت فایل سیستمی

1. داده‌ها در قالب مجموعه‌ای از فایلهای نگهداری می‌شد. FS مدیریت این فایلها را بر عهده داشت.
2. فایل‌ها و محتویات آنها مستقل از هم دیگر بودند.
3. از زبانهای C یا Cobol برای طراحی برنامه کاربردی application آنها استفاده می‌شد.
4. هر فایل حاوی اطلاعات یک بخش از سازمان بود. مثلا یک فایل اطلاعات آموزش، یک فایل برای اطلاعات منابع انسانی و یک فایل برای اطلاعات حسابداری
5. برنامه‌ها مستقل از همدیگر بودند و به شکل کامپیوتری با هم ارتباط نداشتند.

2.2 معایب مدیریت فایل سیستمی

2.2.1 افزونگی داده Data Redundancy

مثلاً هادی ممکن است در دو دانشکده فیزیک و کامپیوتر دانشجو باشد. رکورد اطلاعاتی هادی شامل (نام، نام خانوادگی، آدرس، تلفن، کدملی....) در فایل دو دانشکده تکرار می‌شود. این افزونگی داده حافظه را هدر داده و سرعت دسترسی به اطلاعات را کاهش می‌دهد.

2.2.2 ناسازگاری داده Data Inconsistency

2.2.2.1 به خاطر کپی‌های مختلف

یعنی اطلاعات هادی در دو فایل یکسان نباشد. مثلاً هادی بعد از ثبت‌نام شماره تلفنش تغییر می‌کند. تلفن جدید در دانشکده فیزیک ثبت می‌شود ولی در دانشکده کامپیوتر ثبت نمی‌شود. در نتیجه دو نسخه اطلاعات هادی در سامانه سازگار (یکسان) نیستند.

2.2.2.2 خطا در ورود اطلاعات

یک شکل دیگر ناسازگاری داده ممکن است به هنگام ورود اطلاعات رخ بدهد. مثلاً به هنگام ثبت‌نام هادی در آموزش نام پدر به جای غلامحسن، غلامحسین وارد شود.

2.2.3 دسترسی دشوار به اطلاعات موجود

فرض کنید کارمند دانشکده می‌خواهد دانشجویانی که ساکن شهر تهران هستند را پیدا کند. برنامه آموزش در حال حاضر لیست کامل دانشجویان دانشکده را تولید می‌کند. ولی این امکان را ندارد. کارمند دو انتخاب دارد

1. به شکل دستی دانشجویان تهرانی را لیست کلی استخراج کند
2. از برنامه‌نویس آموزش بخواهد که با تغییر برنامه آموزش این امکان را به برنامه اضافه کند.

حتی اگر این امکان به برنامه اضافه شد. ممکن است کارمند لیست دانشجویان تهرانی مشروط را خواسته باشد. برنامه لیست دانشجویان مشروط و لیست دانشجویان مشروطی را تولید می‌کند ولی متاسفانه لیست دانشجویان مشروطی یک شهر خاص را نمی‌تواند تولید کند.!!!

2.2.3.1 مشکل در ارایه دیدهای (view) ها ترکیبی

به دلیل استقلال سامانه‌ها ارایه دیدهای ترکیبی در عمل وجود نداشته و باید لیستهای ترکیبی به شکل دستی درست شود. مثلا پیدا کردن لیست اساتید و کارمندانی که فرزندان آنها بدهی شهریه دارند. اطلاعات شهریه‌ها در سامانه حسابداری و اطلاعات اساتید و کارمندان در سامانه منابع انسانی موجود است. ولی این دو سامانه کاملاً از هم مستقل و مجزا هستند.

2.2.4 مشکل جامعیتی Integrity

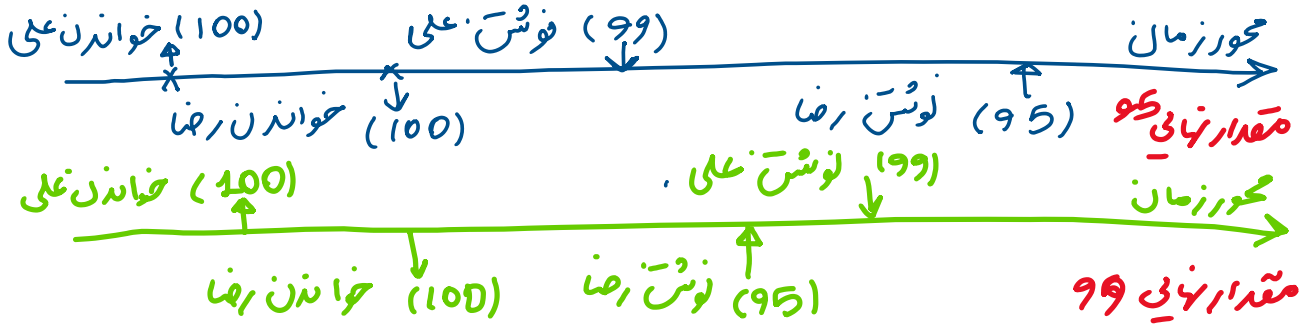
اطلاعات داخل سامانه باید یک سری محدودیتهای consistency constrains داشته باشند. وقتی اطلاعات داخل سامانه این محدودیت‌ها را رعایت می‌کند اصطلاحاً گفته می‌شود جامیعت Integrity خود را حفظ کرده است.

مثلا فرض کنید موجودی حساب دانشکده نباید کمتر از ۱ میلیون تومان باشد. برنامه‌نویس به هنگام نوشتن برنامه این محدودیت را کدنویسی کرده است. بعد از ۱۰ سال به خاطر تورم این رقم باید به ۱۰۰ میلیون افزایش پیدا کند. تغییر این محدودیت مستلزم تغییر تمامی برنامه‌های دانشکده است. نوشتن محدودیتهای که به اطلاعات چند فایل نیاز دارد از این هم مشکل‌تر است. مثلا این محدودیت: موجودی حساب دانشکده‌های بزرگ (با بیش از) ۱۰۰ هیئت علمی نباید کمتر از ۵۰۰ میلیون تومان باشد ولی دانشکده‌های کوچک با کمتر از ۱۰۰ استاد نباید کمتر از ۱۰۰ میلیون باشد.

2.2.5 آنومالی Anomaly در دسترسی همزمان

آنومالی به طور عام یعنی یک اتفاق بد در سامانه اطلاعاتی.

فرض کنید موجودی یک حساب ۱۰۰ میلیون تومان باشد. علی و رضا دو کارمند بانک همزمان از این حساب پول برمی‌دارند، علی ۱ میلیون تومان و رضا ۵ میلیون تومان. علی موجودی جدید حساب را ۹۹ میلیون و رضا ۹۵ میلیون تومان ثبت می‌کند. بر اساس اینکه کدام یک دیرتر بنویسند موجودی حساب ۹۵ میلیون تومان یا ۹۹ میلیون تومان ثبت می‌شود که هر دو اشتباه است، باید ۹۴ میلیون تومان ثبت شود. این اتفاق بد یک آنومالی Anomaly نامیده می‌شود. در این مثال علت آنومالی دسترسی همزمان concurrent access است.

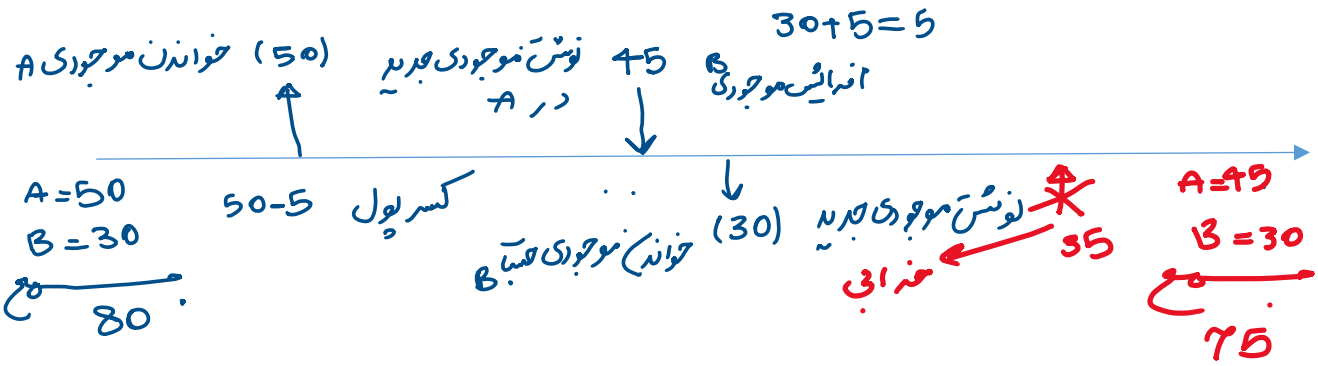


2.2.6 امنیت منطقی پایین

یک کارمند حسابداری که مسئول خرید است نباید از حقوق اساتید خبر داشته باشد. اعمال این محدویت‌های امنیتی در دسترسی‌های کار سختی است.

2.2.7 نقض یکپارچگی عملیات Atomicity

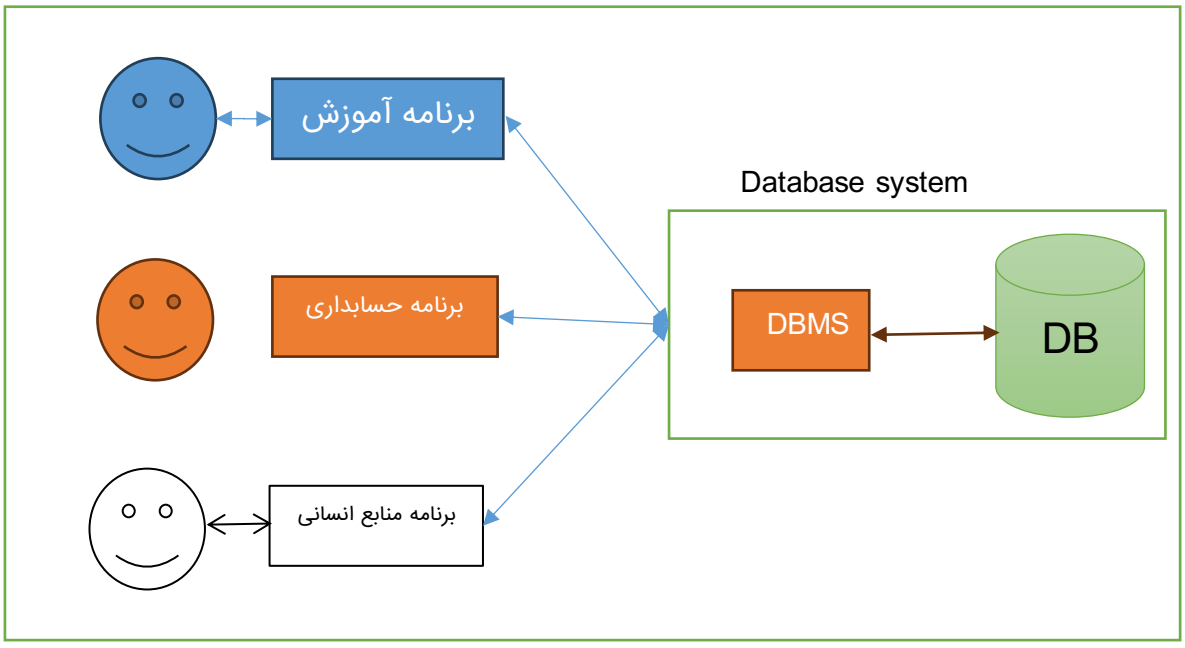
برخی از اعمال باید به شکل Atomic اجرا شوند. یعنی یا همه اجزای آن اجرای شود یا هیچ کدام. یا اصطلاحاً همه یا هیچ (All or None) اجرا شود. مثلا انتقال پول از یک حساب به حساب دیگر در بانکداری.



2.2.8 تعدد برنامه‌نویس، زبان‌های برنامه‌نویسی و فرمت‌های ذخیره داده

3 پایگاه داده

برای رفع مشکلات سیستم فایلینگ نسل دوم سامانه‌های اطلاعاتی کامپیوتری یا همان سامانه پایگاه داده معرفی شد.



3.1 ویژگی‌های پایگاه داده

1 . سیستم پایگاه داده از دو مولفه تشکیل شده است. ۱. (DB) Database ۲. Database management system (DBMS)

تعریف) DBMS . نرم‌افزاری است که از همه جهات پایگاه‌داده را مدیریت می‌کند.

تعریف) DB. فضای ذخیره‌سازی مجتمع با حداقل افزونگی سازماندهی شده بر اساس یک مدل مشخص قابل استفاده توسط یک یا چند کاربر به طور مستقل یا اشتراکی

سوال) در FS هم مانند DB داده‌ها سازماندهی شده بود. پس تفاوت FS و DB در چیست؟

پاسخ) در DB وظایف مدیریت داده‌ها توسط یک نرم‌افزار آماده به نام DBMS انجام می‌شود و برنامه‌های کاربردی و کاربران لازم نیست درگیری این امور مدیریتی بشوند.

سوال) وظایف مدیریتی DBMS چیست؟

1. حفظ جامعیت
2. برقراری امنیت
3. انجام جستجو (کوئری)های سریع
4. کنترل دسترسی‌های همروند
5. جلوگیری از ناسازگاری
6. پشتیبانی از یک Query Language

- Microsoft SQL Server و Oracle دو نمونه از DBMS های رایج است.

3.2 مزایای پایگاه داده

رفع مشکلات فایلینگ در واقع مزایای پایگاه‌داده می‌باشد.

3.2.1 کاهش افزونگی داده Data Redundancy

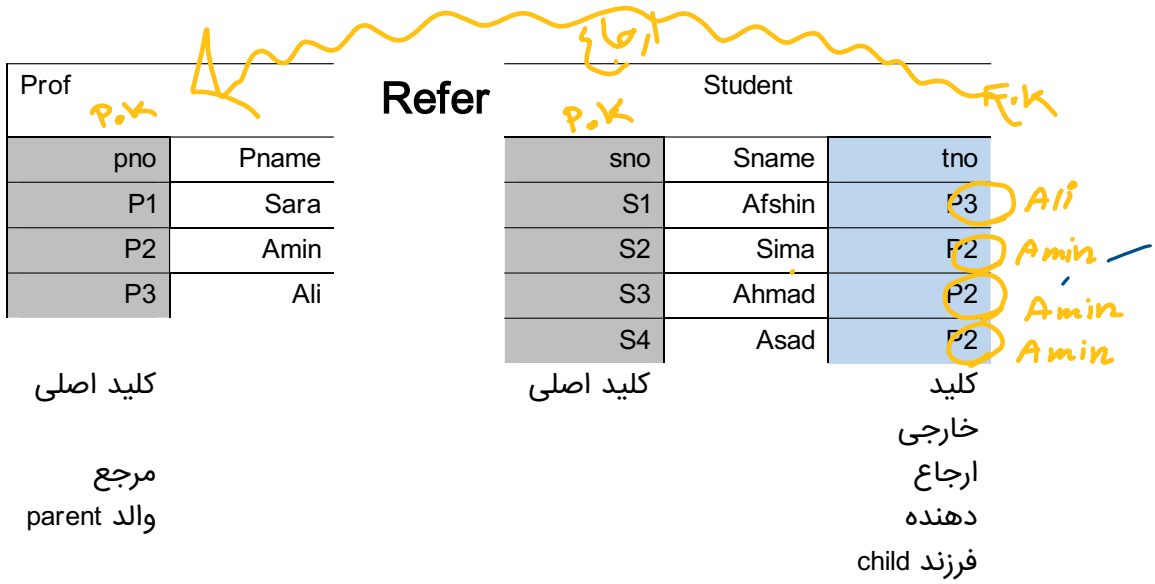
به طور کلی گفته می‌شود که در پایگاه‌داده ما تکرار(افزونگی) را کم می‌کنیم. ولی درباره‌ی این جمله به نکات زیر باید توجه کنید.

3.2.1.1 افزونگی تکنیکال

در پایگاه داده ما افزونگی داده را کم می‌کنیم ولی هیچ وقت نه می‌توانیم و نه باید افزونگی داده را به صفر برسانیم. مثلا یک نوع افزونگی داده در پایگاه داده‌های رابطه‌ای تکرار کلید خارجی Foreign Key است. ما از این تکرار برای ارتباط بین جدول‌ها استفاده می‌کنیم. این افزونگی نه تنها بد نیست بلکه وجود آن ضروری است. ما به این نوع نوع از افزونگی، افزونگی تکنیکال می‌گوییم. مثل بعضی از میکروب‌ها و انگل‌ها که نه تنها برای بدن خطرناک نیست بلکه مفید هم هست.

3.2.1.2 تکثیر Replication

دقت کنید که هر تکراری افزونگی محسوب نمی‌شود. گاهی طراحان پایگاه‌داده با هدف افزایش سرعت دسترسی یا افزایش امنیت فیزیکی داده ممکن است عمداً برخی از اطلاعات را تکرار کنند. به این نوع تکرار تکثیر گفته می‌شود. مثلاً در دیسکهای RAID ما شاهد این تکثیر هستیم.



3.2.2 جلوگیری از ناسازگاری داده Data Inconsistency

3.2.3 دسترسی منعطف و راحت به اطلاعات موجود

3.2.4 حفظ جامعیت Integrity

جامعیت به طور عام یعنی اطلاعات ثبت شده در پایگاه داده از برخی از قوانین پیروی کند. این قوانین یا محدودیت می‌تواند دو دسته دلیل داشته باشد.

3.2.4.1 قوانین کسب و کاری

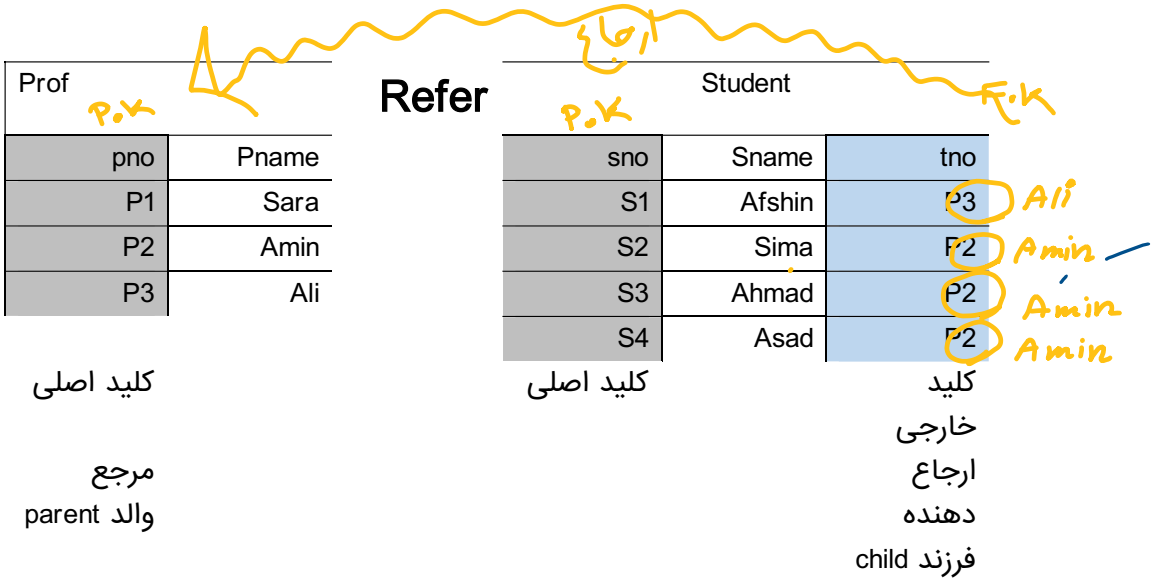
از یک سامانه به سامانه فرق می‌کند.

1. در یک دانشگاه، دانشجویی که مشروط شده نمی‌تواند بیش از ۱۴ واحد بگیرد.
2. در یک بانک شما نمی‌توانید ضامن بیش از ۳ نفر باشید.
3. در یک دانشگاه، یک استاد نمی‌تواند بیش از ۱۶ واحد تدریس داشته باشد.

3.2.4.2 قوانین ساختاری

این قوانین ربطی به قواعد کسب‌وکار ندارد بلکه باید در همه پایگاه‌داده‌های رابطه‌ای برقرار باشد.

1. مقدار کلید اصلی نمی‌تواند تکراری باشد. باید یکتا باشد.
2. مقدار کلید خارجی باید در ستون کلید اصلی جدول مرجع موجود باشد.



3.2.5 دسترسی همزمان Concurrent

3.2.6 افزایش امنیت

در پایگاه داده با اینکه به خاطر یکپارچگی سامانه حفاظت داده در مقابل دسترسی‌های مجاز افزایش می‌یابد. ولی به دلیل تجمیع اطلاعات در یک دیسک میزان آسیب پذیری فیزیکی داده کاهش پیدا می‌کند. اصطلاحاً گفته می‌شود پایگاه‌داده امنیت منطقی را افزایش ولی امنیت فیزیکی را کاهش می‌دهد.

3.2.7 تضمین یکپارچگی عملیات Atomicity

3.3 نقش عوامل انسانی در پایگاه‌داده

افرادی که با پایگاه‌داده در سامانه اطلاعاتی کار می‌کنند یکی از چهار نقش زیر را دارند.

3.3.1 مدیر پایگاه‌داده

مدیر پایگاه‌داده Database Administrator (DBA) مدیر مسئول پایگاه‌داده است. وظایف وی عبارت است از

1. طراحی ساختار کلی پایگاه داده
2. تعیین قواعد امنیتی و سطوح دسترسی
3. تعیین قواعد جامعیتی

دقت کنید. مدیرپایگاه داده متخصص پایگاه‌داده است و در زمینه اطلاعات آن سازمان تخصصی ندارد. به هنگام طراحی نقشی به نام معمار داده Data Architect DA که به سازمان آشنایی کامل دارد، اطلاعات آن، گردش کار و نوع پرس‌وجوها را می‌شناسد به DBA کمک می‌کند. DA متخصص پایگاه داده نیست.

3.3.2 تحلیلگر داده Data Analyst

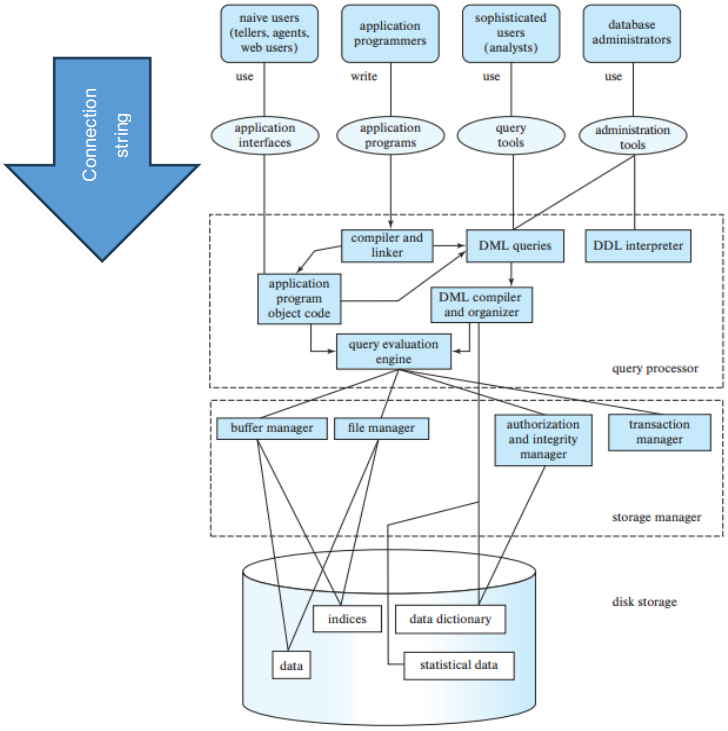
این شخص متخصص پرس‌وجو نویسی است. با شناختی که از ساختار پایگاه دارد می‌تواند اطلاعات مختلف را از پایگاه‌داده استخراج کند.

3.3.3 توسعه دهندگان برنامه کاربردی Application Developer

این فرد متخصص برنامه‌نویسی هستند و به پرس‌وجو نویسی هم تسلط نسبی دارند. برنامه‌های کاربردی مورد نیاز کاربر عادی برای تعامل با پایگاه‌داده توسط وی نوشته می‌شود.

3.3.4 کاربر عادی

کاربر عادی هیچ تخصصی در زمینه کامپیوتر و پایگاه‌داده نداشته و با استفاده از برنامه‌های کاربردی سازمان با پایگاه داده تعامل دارد. در گذشته کاربر عادی عموماً کارمند سازمان بود. مثلا صندوقدار بانک با استفاده از برنامه کاربردی به پایگاه‌داده بانک وصل می‌شد. با ظهور برنامه‌های کاربردی تحت وب و موبایل این وضعیت تغییر کرده است. مثلاً همه شما از طریق موبایل بانک می‌توانید پول جابجا کنید. در حین این عملیات شما با یک موبایل اپلیکیشن به پایگاه‌داده بانک وصل شده‌اید و کاربر عادی پایگاه‌داده محسوب می‌شوید.



3.4 Connection string

اطلاعاتی که برای اتصال و دسترسی به پایگاه‌داده از طرف برنامه باید به پایگاه داده ارسال شود.

3.5 کاتالوگ

تعریف 1. متاداده. متا داده meta data داده درباره داده است.

نام دیگر. متا داده = فراداده

مثال. در پایگاه داده رابطه‌ای، نام جدول‌ها، نام ستون‌ها، نوع داده هر ستون متا داده است.

تعریف 2 کاتالوگ. متا داده‌های پایگاه‌داده در ساختاری به نام کاتالوگ ذخیره می شود.

نام دیگر. کاتالوگ = واژه‌نامه= Data Dictionary = Catalog

در یک پایگاه‌داده رابطه‌ای متاداده‌های زیر در کاتالوگ ذخیره می‌شود.

- نام جدول‌ها (رابطه‌ها)
- نام ستون‌ها (خصوصیت‌ها)
- نوع داده هر ستون و دامنه‌ی آن
- نام دیدهای view ها تعریف شده روی پایگاه‌داده
- قواعد جامعیتی مثلاً یکتایی کلید اصلی
- نام کاربران و رمزعبورها برای احراز هویت کاربران
- دسترسی‌های مجاز هر کاربر

3.6 کوئری

هر گونه درخواستی برای کار با پایگاه داده کوئری نامیده می‌شود.

نام دیگر. پرس‌وجو = کوئری= query

3.7 DDL

Data Definition Language

این دستورات در ساختار پایگاه‌داده اثر گذارند و محتوی پایگاه‌داده را تغییر نمی‌دهند.

مثلا Create Table برای ایجاد جدول، Drop Table برای حذف جدول، دستور Primary key برای تعیین کلید اصلی جدول..

3.8 DML

Data Manipulation Language

این دستورات برای دستکاری و تغییر محتوی پایگاه‌داده کاربرد دارند.

مثلا) insert into برای اضافه‌کردن سطر به جدول، delete from برای حذف سطر از جدول،

3.9 DCL

Data Control Language

این دستورات برای تعریف محدودیت‌های امنیتی و کنترل دسترسی به پایگاه‌داده استفاده می‌شوند.

مثلا) grant برای اعطای مجوز دسترسی، revoke برای بازپس‌گیری مجوز دسترسی

DSL 3.10

به اجتماع دستورات DML و DDL و DCL ، DSL گفته می شود.

نام دیگر. DSL = Data Sub Language = Query Language

معروف ترین زبان DSL ، SQL است. این زبان اینقدر رایج و مهم است که خیلی از افراد آن را معادل پایگاه داده می‌دانند. در حالی‌که SQL یکی از زبان‌ها کوئری نویسی پایگاه‌داده رابطه‌ای می‌باشد.

4 مدل‌های داده

یادآوری تعریف) DB. فضای ذخیره‌سازی مجتمع با حداقل افزونگی سازماندهی شده بر اساس یک مدل مشخص قابل استفاده توسط یک یا چند کاربر به طور مستقل یا اشتراکی

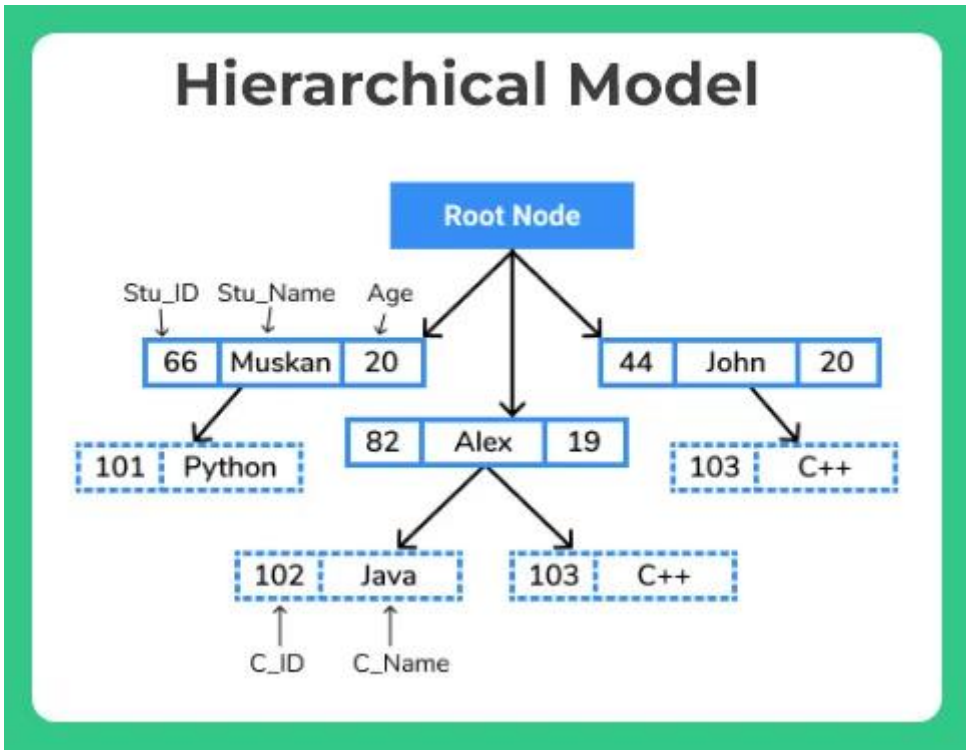
در این طراحی سازماندهی بر اساس مدل‌های مشخص و از پیش تعریف شده‌ای باید انجام شود. مدل‌های متداول در پایگاه‌داده‌ها را در این فصل معرفی می‌کنیم.

4.1 مدل سلسله مراتبی Hierarchical

قدیمی‌ترین مدل است. در این مدل پایگاه‌داده به شکل یک گراف نمایش داده می‌شود. گره‌ها جداولی هستند که رکوردهای اطلاعاتی را در خود نگه می‌دارند. و لبه‌ها ارتباط بین جدول‌ها را نشان می‌دهد.
متاسفانه این لبه‌ها فقط رابطه یک به چند و یک به یک را نمایش می‌دهد. هر گره ساختار مشخص و ثابتی دارد که رکورد record نامیده می‌شود.

وضعیت= منسوخ

تعریف 3) رکورد. ساختار داده‌ای متشکل از تعدادی فیلد که نوع و تعداد فیلدهای به هنگام تعریف رکورد مشخص شده و ثابت می‌ماند.



شکل 1 مدل سلسله مراتبی

4.1.1 نقاط ضعف

- افزونگی داده. مثلاً به ازای هر دانشجوی که درس C++ را گرفته رکورد آن تکرار شده است.
- آنومالی در درج، حذف و برزورسانی
- نداشتن تئوری ریاضی
- پیچیدگی

مثال آنومالی درج) رکورد درس نمی‌تواند بدون دانشجو باشد. برای اضافه کردن یک درس جدید چون هنوز هیچ دانشجویی آن را نگرفته است. مجبوریم یک گره دانشجوی مجازی ایجاد کنیم و زیر آن این گره درس را اضافه کنیم.

مثال آنومالی حذف) اگر بخواهیم درس C++ جان را حذف کنیم. ناخواسته رکورد جان هم حذف می‌شود.

مثال) آنومالی بروزرسانی. اگر بخواهیم نام درس C++ را عوض کنیم. باید در تمامی گره‌ها این کار را انجام دهیم.

4.2 مدل شبکه‌ای Network

براساس مدل سلسله مراتبی تعریف شده‌است. با این تفاوت که از رابطه چند به چند هم دارد.

وضعیت = منسوخ

4.2.1 نقاط قوت

- افزونگی کمتر در مقایسه با سلسله مراتبی

- نداشتن آنومالی

4.2.2 نقاط ضعف

- نداشتن تئوری ریاضی قوی

4.3 مدل رابطه‌ای Relational

مدل رابطه‌ای از مجموعه‌ای از جدول‌ها برای نمایش داده‌ها و رابطه بین آنها استفاده می‌کند. هر جدول شامل چند ستون است و هر جدول یک نام دارد. رابطه بین جدولها نیز با محتویات جدول‌ها نشان داده می‌شود. جدولهای رابطه‌ای نمونه‌ی دیگری از record-based مدل هستند.

student_Id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_Id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_Id	subject_Id	marks
1	1	98
1	2	78
2	1	76
3	2	88

شکل 2 مدل رابطه‌ای

این مدل فراگیرترین و موفق‌ترین مدل پایگاه‌داده در بازار است. به DBMS هایی که بر اساس این مدل طراحی شده RDBMS گفته می‌شود. مانند Oracle یا Microsoft SQL Server. ما در این درس با مدل رابطه‌ای کار داریم. این مدل بر اساس جبر رابطه‌ای Relational Algebra تعریف شده‌است. همین تئوری ریاضی قوی به این مدل در این سادگی قدرت عجیبی داده است.

4.3.1 نقاط قوت

- تئوری ریاضی قوی یعنی جبر رابطه‌ای
- سادگی
- کامل بودن
- درست بودن

تعریف 4)کامل بودن: هر درخواست (کوئری) کلیه پاسخ‌های درست را تولید می‌کند.

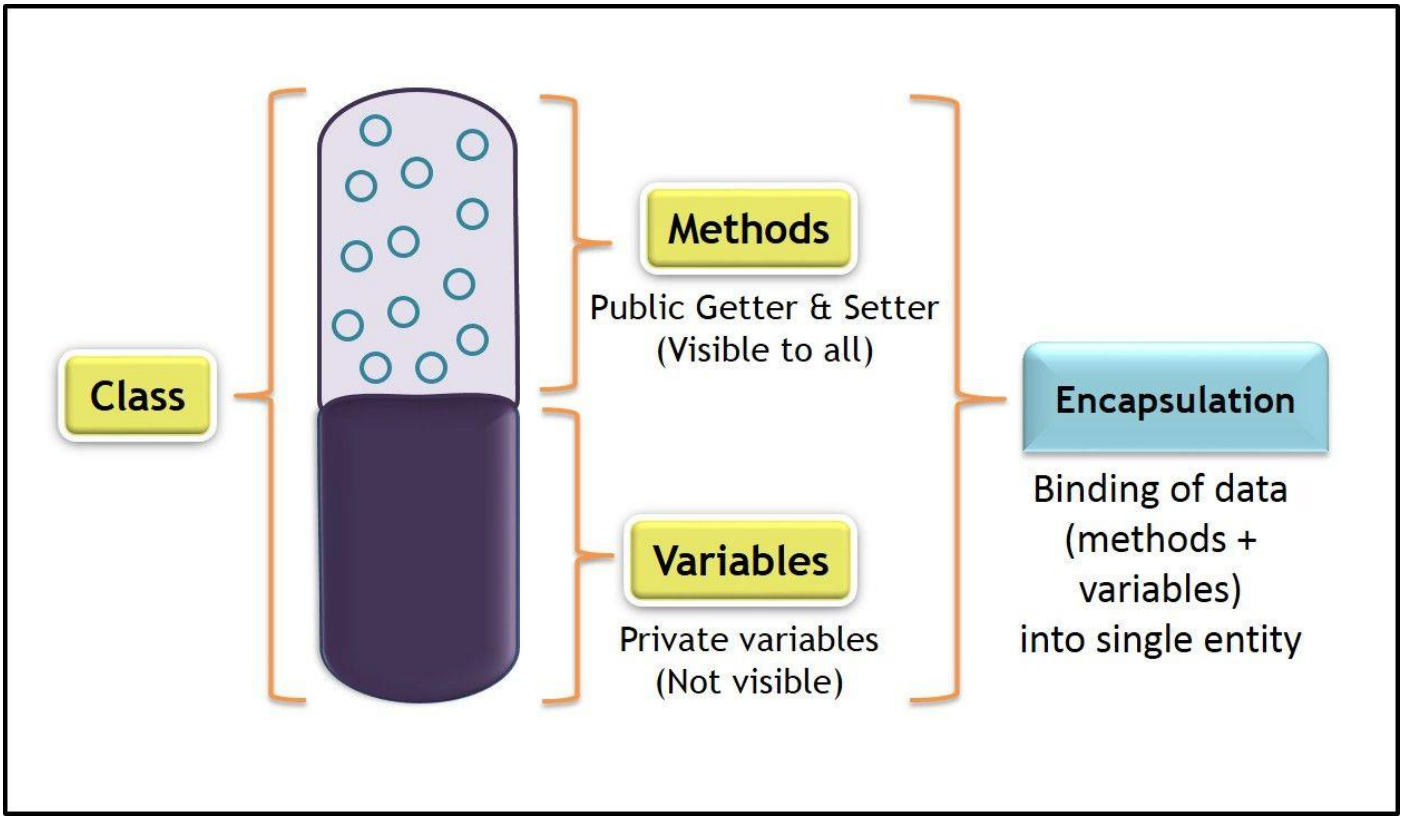
تعریف 5)درست بودن: هر پاسخی که تولید می‌شود قطعاً درست است.

4.3.2 نقاط ضعف

- عدم پشتیبانی از داده‌های پیچیده چندرسانه‌ای مانند تصویر، صدا، ویدیو..

4.4 مدل شی‌گرا Object Oriented

پس از موفقیت برنامه‌نویسی شی‌گرا Object Oriented Programming در صنعت نرم‌افزار، آن ایده‌ها برای پیشنهاد مدل پایگاه داده استفاده شد. مثلا O2 یکی از نمونه DBMS های مبتنی بر شی‌گرایی است. مدل شی‌گرا مانند مدل رابطه‌ای ریاضیات قوی نداشته و به همین خاطر ضعیف‌تر است. هر چند ایده Encapsulation به بسیاری از RDBMS امروزی راه یافته است. در این RDBMS ها شما می‌توانید شی object را در جدول‌های رابطه‌ای ذخیره کنید. از آنجایی که یک شی علاوه بر داده می‌تواند شامل method هم باشد. با این ترفند می‌توان در مدل رابطه‌ای به نوعی عملیات را هم در کنار داده در جدول داشت.



شکل 3 Encapsulation در برنامه‌نویسی شی‌گرا

4.4.1 نقاط قوت

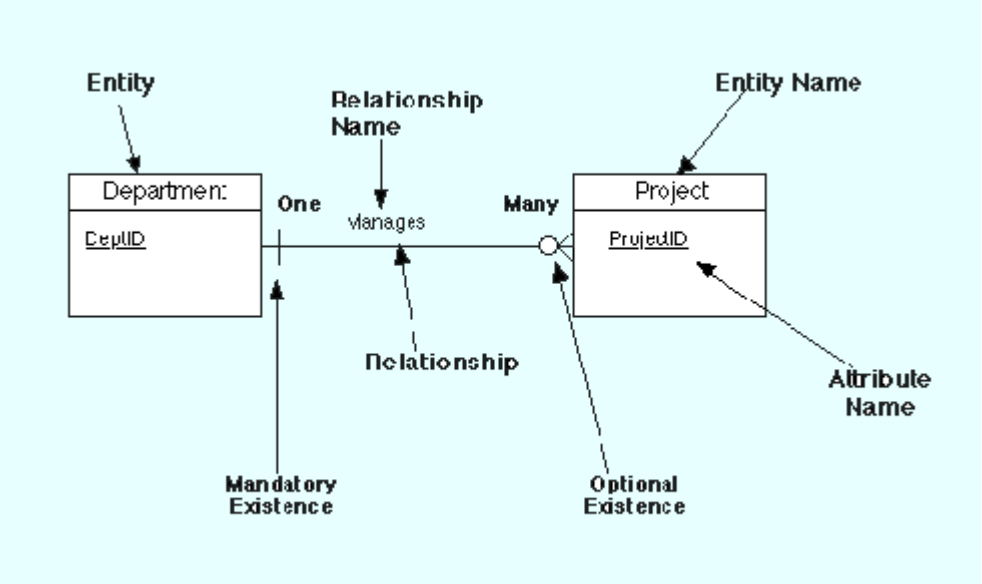
- پشتیبانی از داده‌های چندرسانه‌ای
- ذخیره‌سازی عملیات در کنار داده

4.4.2 نقاط ضعف

- نداشتن تئوری ریاضی قوی

4.5 مدل ER

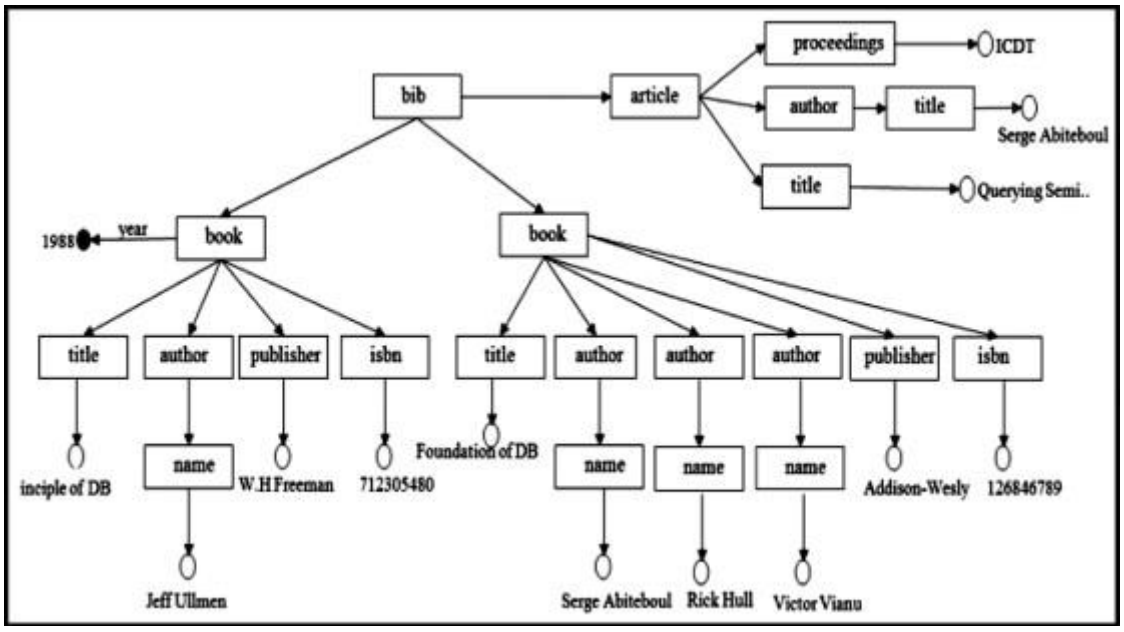
در این مدل پایگاه‌داده با موجودیت Entity و رابطه Relation بین این موجودیت‌ها مدل می‌شود. از این مدل برای طراحی و درک نیازمندی‌های پایگاه داده خیلی استفاده می‌شود.



شکل 4 ER data model

4.6 مدل نیمه ساختار یافته Semi-structured

در مدل نیم ساختار یافته، ساختار داده‌ها منعطف است، یعنی بر خلاف مفهوم رکورد ثابت نیست و در حین کار بر اساس نیاز قابل تغییر است. این ویژگی در سالهای اخیر خیلی مورد توجه قرار گرفته است. در سامانه اطلاعاتی سازمان‌ها سنتی مثل بانک یا دانشگاه نیازمندی‌های اطلاعاتی در طول زمان زیاد تغییر نمی‌کند. ولی در یک سامانه اطلاعاتی شبکه اجتماعی مانند اینستاگرام نیازمندی‌ها به هنگام طراحی اولیه اصلاً مشخص نیست. JSON و XML دو نمونه از این مدل‌های نیمه ساختار یافته است.



شکل 5 پایگاه داده کتابخانه با مدل نیمه ساختار یافته

5 انتزاع داده Abstraction

داده‌ها از پایگاه داده باید با سهولت و سرعت استخراج شود. برای رسیدن به سرعت بالا و صرفه‌جویی در حافظه، برنامه‌نویسان از تکنیکهای خیلی پیچیده‌ای برای سازماندهی و ایندکس‌گذاری داده در پایگاه‌داده استفاده می‌کنند. این حجم از پیچیدگی استفاده از پایگاه‌داده را برای غیرمتخصصین ناممکن می‌کند. راه‌حل مخفی کردن پیچیدگی از دید کاربر معمولی با استفاده از Data Abstraction است.

مفهوم انتزاع قبل از مهندسی کامپیوتر هم وجود داشته است. مثلا اتومبیل ماشین خیلی پیچیده‌ای است. به جز مهندسین و مکانیک‌های کسی از پیچیدگی‌های موتور، فرمان یا گیربکس اتومبیل هیچ اطلاعی ندارد. ولی یک فرد معمولی با استفاده از چند کنترل ساده مثل گاز، کلاچ، ترمز و فرمان می‌تواند ماشین را براند. این مثالی از مخفی کردن پیچیدگی در مهندسی مکانیک است.

انتزاع در پایگاه داده سه سطح داشته و در شکل زیر نمایش داده شده است. در برخی از منابع به آن معماری سه لایه‌ای ANSI گفته می‌شود.

user/developers

DBA سازماندهی ومدل سازی

DBMS/DBA

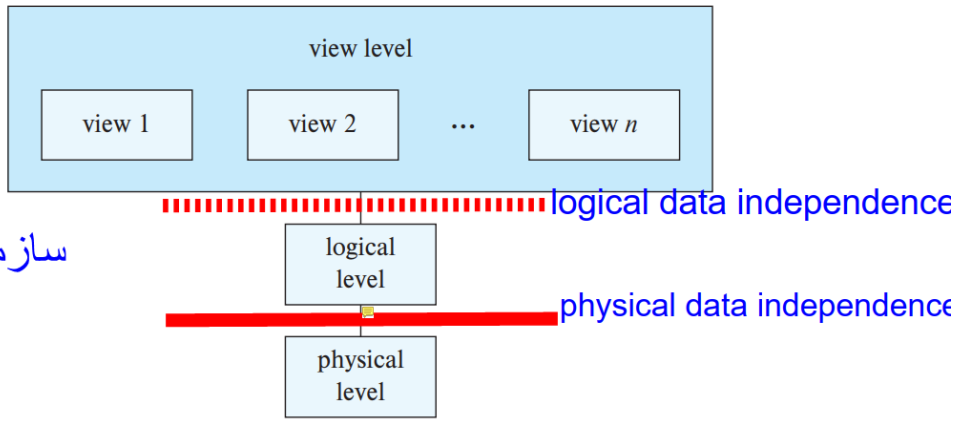


Figure 1.2 The three levels of data abstraction.

5.1 سطح فیزیکی Physical

پایین‌ترین سطح انتزاع است. در این سطح چگونگی (how) ذخیره‌سازی داده روی دیسکها مشخص می‌شود. جزییات این سطح توسط DBMS کنترل می‌شود. البته در برخی موارد DBA می‌تواند با تغییر تنظیمات DBMS را هدایت کند.

نام‌های دیگر. سطح فیزیکی = Physical Level = Internal Level = سطح داخلی

5.2 سطح منطقی Logical

سطح بالاتر، منطقی نامیده می‌شود. در این سطح بر آنچه (what) که باید ذخیره شود تمرکز دارد. مثلا DBA در یک پایگاه‌داده رابطه‌ای در این سطح مشخص می‌کند که پایگاه چه جدولهایی دارد، نام جدول‌ها چیست، در هر جدول چه ستون‌هایی ذخیره می‌شود و این جدول‌ها چگونه با هم ارتباط دارند. با اینکه ذخیره‌سازی این جدول‌ها روی دیسک ممکن است پیچیدگی‌های زیادی داشته باشد ولی DBA اصلاً درگیر این پیچیدگی نمی‌شود. به این استقلال فیزیکی گفته می‌شود.

تعریف 6) استقلال داده فیزیکی. لایه فیزیکی به طور کامل از لایه منطقی مستقل است. یعنی بدون تغییر در لایه منطقی و لایه دید هر تغییری در لایه فیزیکی قابل انجام است.

5.2.1 تقسیم بندی سطح منطقی

سطح منطقی در کنترل DBA و مهمترین سطح مدل سه سطحی است. در برخی از منابع این سطح را لایه ادراکی Conceptual Level نامیده و آن را به دو زیر سطح تقسیم می‌کنند.

1. لایه ادراکی عام. مستقل از مدل پایگاه‌داده که با فرآیند طراحی مفهومی Conceptual Design طراحی می‌شود.
2. لایه ادارکی خاص. بر اساس مدل پایگاه داده در فرآیندی به نام طراحی منطقی Logical Design طراحی می‌شود.

5.3 سطح دید view

ساده‌سازی (مخفی کردن پیچیدگی) لایه منطقی در لایه دید هم ادامه پیدا می‌کند. مثلاً کاربر معمولی دانش پایگاه‌داده رابطه‌ای ندارد. بنابراین با او نمی‌توان از جدول، کلید اصلی، کلید خارجی و ارتباط بین جدول‌ها صحبت کرد. این لایه با اضافه کردن فرم‌های و منوهای ساده روی این جدول‌ها دسترسی را برای وی راحتتر می‌کند. حتی به برنامه‌نویس‌های آشنا به پرس‌وجو نویسی، به جای دسترسی به تمامی جدول‌ها فقط دید محدودی از جدول‌ها در حد نیاز داده می‌شود. این لایه دیدهای view های مختلفی از پایگاه‌داده می‌تواند ایجاد کند. برنامه‌نویس‌ها و تحلیلگران داده در این لایه با نوشتن کوئری اطلاعات از پایگاه‌داده استخراج می‌کنند. چون با تغییر نوع پایگاه‌داده ممکن است که زبان پرس‌وجو نویسی تغییر کند استقلال لایه منطقی از لایه دید مطلق نیست.

تعریف 7) استقلال داده منطقی. سطح دید به طور نسبی از سطح منطقی مستقل است.

نام‌های دیگر. لایه دید = view = خارجی = External

6 پایگاه‌داده و معماری برنامه‌های کاربردی

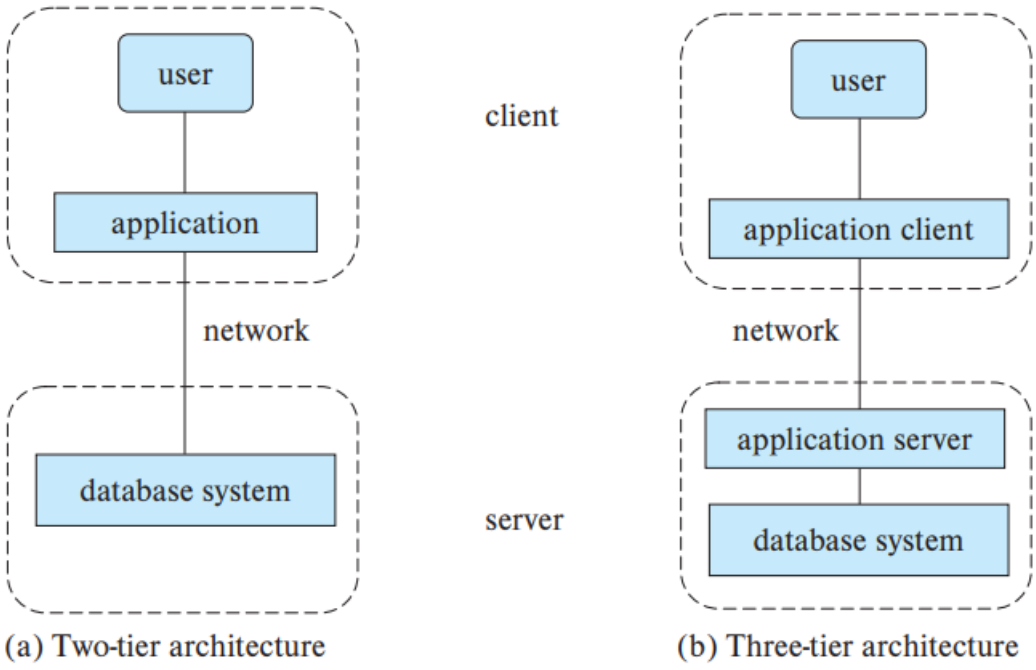


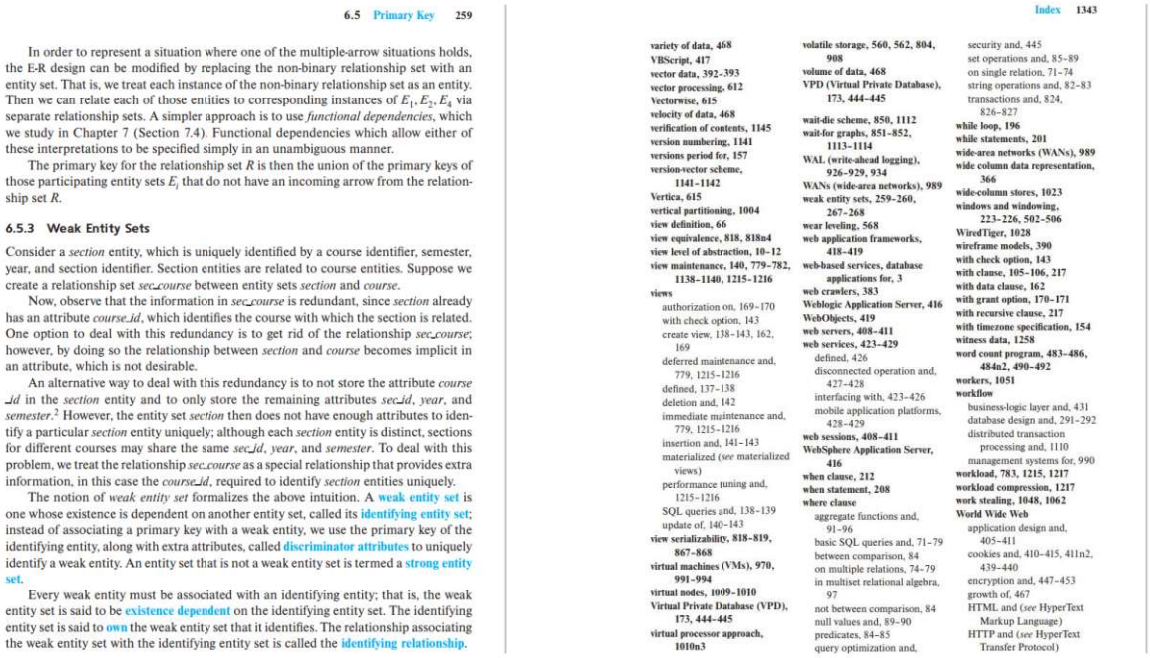
Figure 1.4 Two-tier and three-tier architectures.

مزیای سه سطحی: ۱. امنیت بالاتر ۲. کارایی بالاتر

هدف شاخص‌گذاری indexing= افزایش سرعت دسترسی به اطلاعات در پایگاه داده

1 شاخص‌گذاری چیست؟

ایندکس در پایگاه داده خیلی شبیه ایندکس در کتاب است.



6.5 Primary Key 259

In order to represent a situation where one of the multiple-arrow situations holds, the E-R design can be modified by replacing the non-binary relationship set with an entity set. That is, we treat each instance of the non-binary relationship set as an entity. Then we can relate each of those entities to corresponding instances of E_1, E_2, E_4 via separate relationship sets. A simpler approach is to use *functional dependencies*, which we study in Chapter 7 (Section 7.4). Functional dependencies which allow either of these interpretations to be specified simply in an unambiguous manner.

The primary key for the relationship set R is then the union of the primary keys of those participating entity sets E_i that do not have an incoming arrow from the relationship set R.

6.5.3 Weak Entity Sets

Consider a *section* entity, which is uniquely identified by a course identifier, semester, year, and section identifier. Section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.

Now, observe that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related. One option to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *sec_id*, *year*, and *semester*.[2] However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely; although each *section* entity is distinct, sections for different courses may share the same *sec_id*, *year*, and *semester*. To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case the *course_id*, required to identify *section* entities uniquely.

The notion of *weak entity set* formalizes the above intuition. A **weak entity set** is one whose existence is dependent on another entity set, called its **identifying entity set**; instead of associating a primary key with a weak entity, we use the primary key of the identifying entity, along with extra attributes, called **discriminator attributes** to uniquely identify a weak entity. An entity set that is not a weak entity set is termed a **strong entity set**.

Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

Index 1343

مثلا شما فرض کنید می‌خواهید ببنید که کتاب سیلبرشاتس درباره weak entity چه گفته است. شما مجبور نیستید همه کتاب را بخوانید تا به weak entity برسید! در آخر هر کتاب فنی ایندکس اصطلاحات فنی قرار گرفته. این لیست مرتب است. باید به ایندکس آخر کتاب مراجعه کنید. در لیست مرتب شده اصطلاحات فنی واژه مورد نظر weak entity را پیدا کنید. و به صفحه‌ی که در مقابل آن درج شده است. مراجعه کنید.

شاخص یا ایندکس در پایگاه داده هم همین کار برای شما انجام می‌دهد. برای خواندن یک رکورد لازم نیست همه رکوردها را بخوانید. کافی است آن آدرس آن رکورد را در ایندکس پیدا کنید و تنها آن رکورد را بخوانید. در ایندکس پایگاه داده مثل ایندکس کتاب مرتب است تا کار جستجو در آن سریع انجام شود. ایندکس پایگاه داده مثل ایندکس کتاب کوچک است. مثلاً یک کتاب ۱۰۰۰ صفحه‌ای ایندکسی به اندازه ۲۰ صفحه دارد. به همین شکل یک پایگاه داده ده ترابایتی ممکن است ایندکسی به اندازه ۲ گیگا بایت داشته باشد. این اجازه می‌دهد که ایندکس پایگاه داده به جای دیسک در RAM ذخیره شود.

2 فایل شاخص ترتیبی

تعریف: فایل ترتیبی Sequential File . جدولی (مجموعه رکوردهایی) که بر اساس یکی از ستونها (خصوصیت) مرتب شده باشد.

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Figure 14.1 Sequential file for *instructor* records.

2.1 شاخص‌گذاری غیرخوشه‌ای

تعریف:فایل شاخص Index File. یک جدول که دو ستون دارد. ستون اول یکی از خصیصه‌های جدول اصلی است که می‌خواهیم روی آن شاخص‌گذاری کنیم. ستون دوم اشاره‌گری (pointer) که به یک رکورد در جدول پایگاه داده اشاره می‌کند. به هر سطر (رکورد) این جدول می‌گویند یک مدخل یا Entry. فایل شاخص بر اساس ستون اول مرتب شده است.

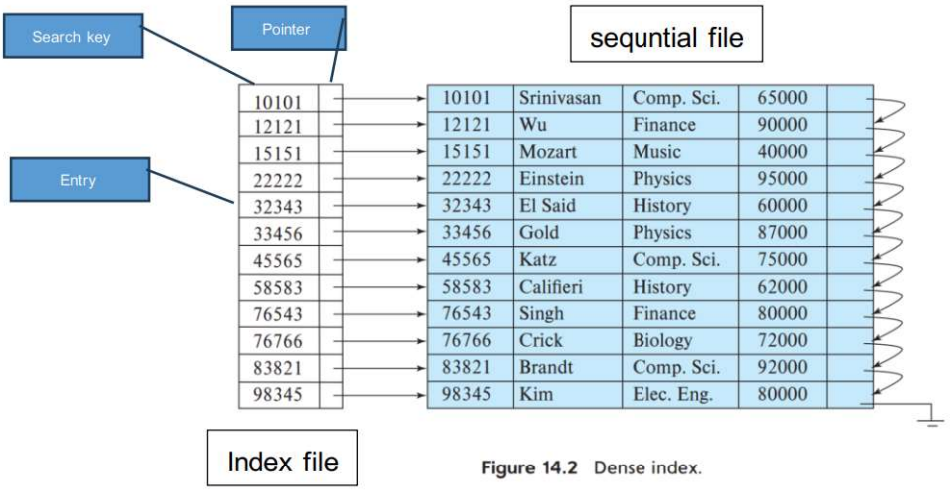


Figure 14.2 Dense index.

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید. استفاده از این جزوه بدون خرید فیلم‌ها غیرقانونی و غیرشرعی است.

تعریف: کلید جستجو Search Key. به خصوصیتی که برای شاخص‌گذاری انتخاب شده است. کلید جستجو گفته می‌شود.

سوال تفاوت کلید جستجو با کلید اصلی یا کلید خارجی چیست؟

کلید جستجو مثل بقیه کلیدها یک خصوصیت یا ستون جدول است ولی تفاوت در کاربرد است. ۱. کلید جستجو در جستجوی رکوردها استفاده می‌شود. ۲. کلید اصلی بین رکوردهای تمایز ایجاد می‌کند. ۳. کلید خارجی بین جدولها ارتباط ایجاد می‌کند. با اینکه در همه آنها از واژه کلید استفاده شده‌است ولی کلاً با هم متفاوت هستند.

سوال) چرا فایل شاخص بر اساس کلید جستجو مرتب شده است؟

پاسخ) برای اینکه امکان جستجوی سریع‌تر فراهم شود.

سوال) آیا کلید جستجو باید همان کلید اصلی باشد؟

نکته) معمولاً همان خصوصیتی که در جدول به عنوان کلید اصلی انتخاب شده برای شاخص‌گذاری هم استفاده می‌شود. مثلا شماره استادی در جدول استادها کلید اصلی باشد و از همین شماره دانشجویی برای شاخص‌گذاری هم استفاده می‌شود، ولی این الزامی نیست.

تعریف شاخص متراکمDense Index . وقتی تمامی مقادیر خصوصیت کلید جستجو در فایل شاخص ذخیره شده باشد.

سوال) آیا باید همه مقادیر خصوصیت کلید جستجو در فایل شاخص ذخیره شده باشد؟ خیر

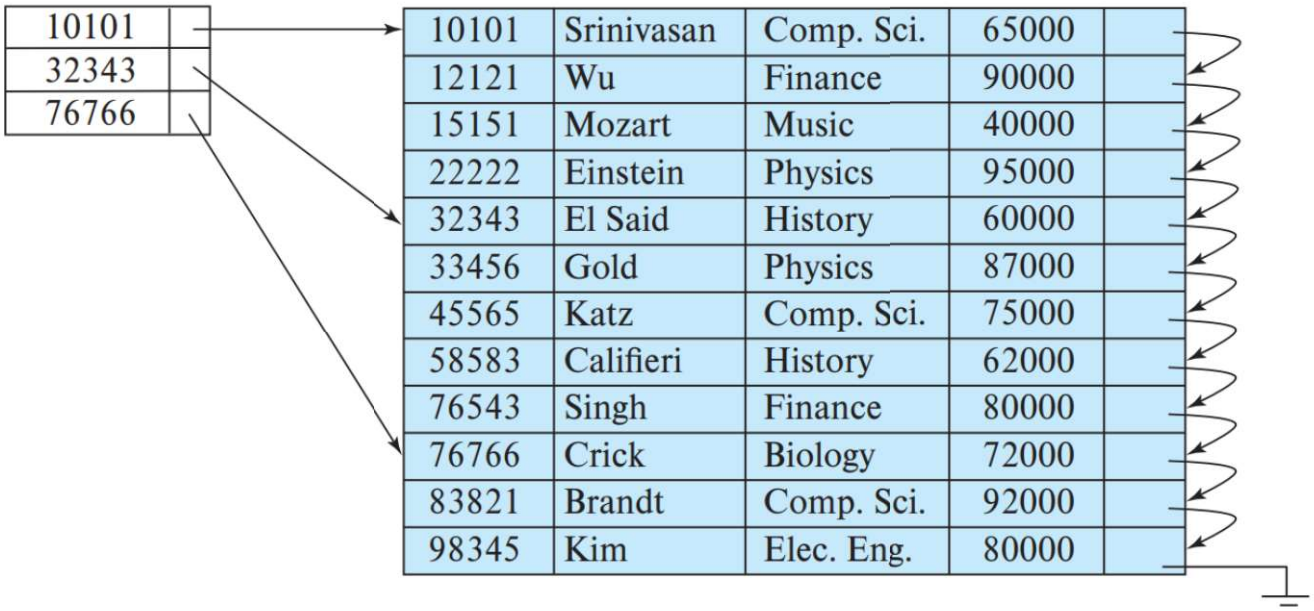


Figure 14.3 Sparse index.

تعریف شاخص خلوت Sparce Index. لازم نیست همه مقادیر خصوصیت کلید جستجو در فایل شاخص ذخیره شده باشد. در این حالت به گفته می‌شود شاخص خلوت Sparce Index داریم.

سوال) سرعت بازیابی در شاخص خلوت بیشتر است یا متراکم؟ شاخص متراکم

سوال) اندازه شاخص خلوت کوچکتر است یا شاخص متراکم؟ شاخص خلوت

سوال) در جدول اساتید می‌خواهیم جستجو را بر اساس نام دپارتمان استاد انجام دهیم. آیا شاخص‌گذاری شماره استاد سرعت جستجو را افزایش می‌دهد؟ خیر. باید شاخص‌گذاری بر اساس نام دپارتمان انجام شود.

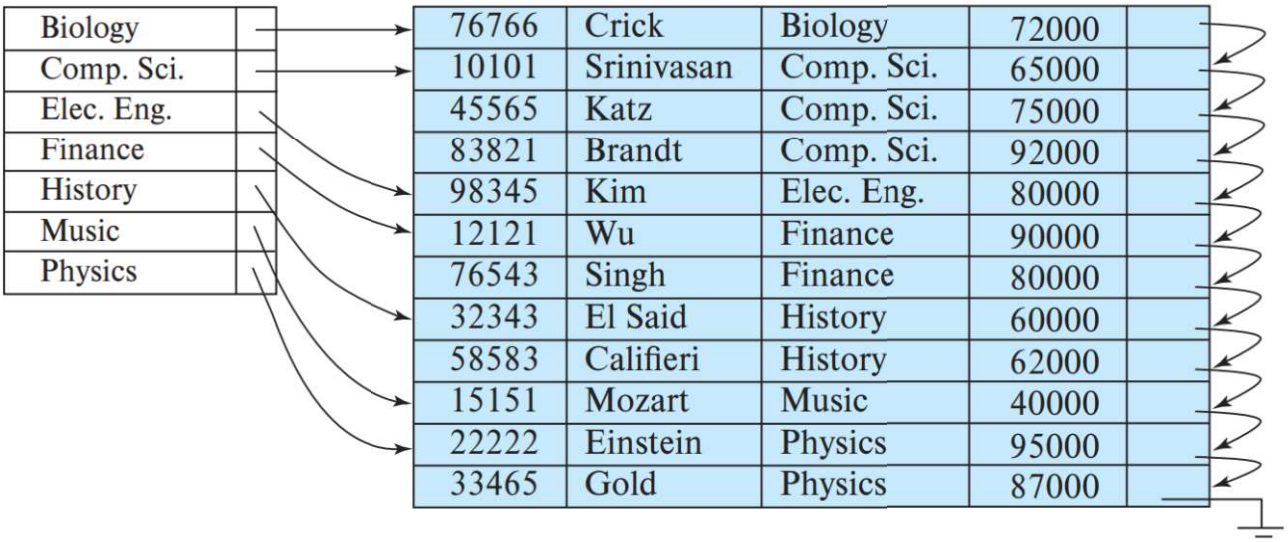


Figure 14.4 Dense index with search key *dept_name*.

نکته: شاخص‌گذاری سرعت دسترسی را برای جستجو بر اساس کلید جستجو افزایش می‌دهد. و در سرعت جستجو بر اساس دیگر خصیصه‌ها تاثیری ندارد.

تعریف فایل ترتیبی شاخص‌گذاری شده index-sequential file: اگر جدولی بر اساس یک خصوصت مرتب باشد و روی همان خصوصیت شاخص‌گذاری کنیم به آن فایل ترتیبی شاخص‌گذاری شده گفته می‌شود.

تعریف شاخص اصلی Primary Index . اگر فایل ترتیبی بر اساس یک خصیصه مرتب شده باشد و کلید جستجو روی همان خصیصه تعریف شده باشد. به این شاخص اصلی می‌گویند.

شاخص اصلی = Primary Index = شاخص خوشه‌ای =Clustering Index

سوال) آیا شاخص اصلی باید روی کلید اصلی تعریف شود؟ خیر. در عمل شاخص اصلی معمولا روی کلید اصلی تعریف می‌شود. ولی از نظر نظری هیچ الزامی نیست.

2.1.1 شاخص‌گذاری چندلایه

سوال) اندازه فایل شاخص بزرگتر است یا فایل ترتیبی متناظر با آن؟ فایل شاخص

سوال) آیا می‌توان فایل شاخص را در RAM ذخیره کرد؟ بلی و خیر

با توجه به اینکه فایل شاخص اندازه کوچکتری دارد معمولا در RAM جا می‌شود. ولی در پایگاه داده‌های خیلی بزرگ ممکن است در RAM جا نشود. اگر فایل شاخص در RAM نباشد سرعت جستجوی رکوردها به شدت کاهش می‌یابد برای رفع این مشکل از شاخص‌گذاری چندلایه استفاده می‌کنند.

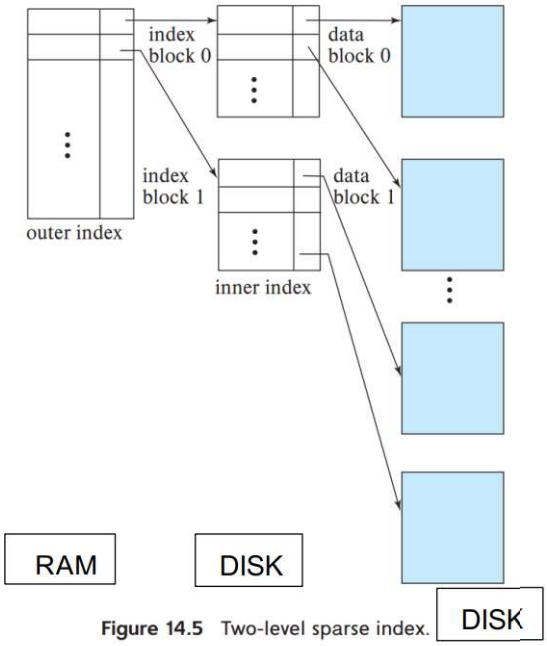


Figure 14.5 Two-level sparse index.

2.2 شاخص‌گذاری غیرخوشه‌ای

سوال) آیا می‌شود کلید جستجوی داشته باشیم که فایل ترتیبی بر اساس آن مرتب نشده باشد؟ بلی

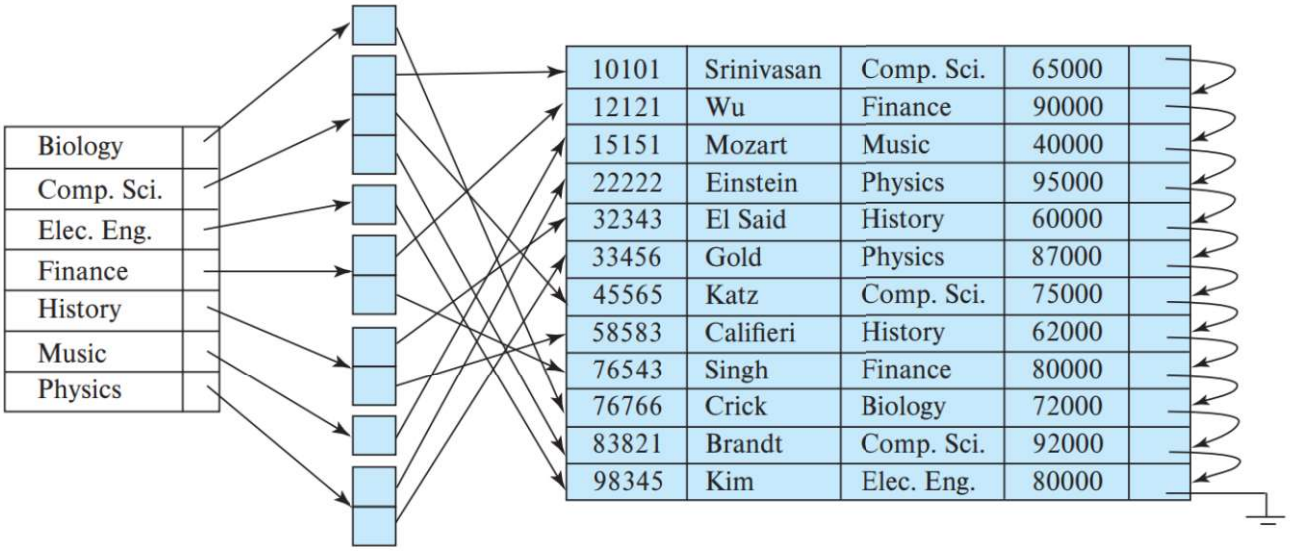


Figure 14.6 Secondary index on *instructor* file, on noncandidate key *dept_name*.

سوال) آیا می‌توان روی یک فایل غیر ترتیبی شاخص‌گذاری داشته باشیم؟ بلی. شاخص‌گذاری ثانویه

تعریف شاخص ثانویه Secondary Index. اگر فایل ترتیبی بر اساس یک خصیصه مرتب شده باشد و کلید جستجو روی خصیصه دیگر تعریف شده باشد. به این شاخص ثانویه می‌گویند.

شاخص ثانویه = Secondary Index = شاخص غیرخوشه‌ای = non Clustering Index

سوال) آیا شاخص ثانویه می‌تواند خلوت باشد؟ خیر

نکته: ما می‌توانیم در پایگاه داده چند کلید جستجو داشته باشیم.

نکته: هر کدام از کلیدهای جستجو ساختار شاخص جداگانه‌ی خودش را داد.

سوال) آیا می‌توان یک شاخص روی چند خصوصیت تعریف بشود؟ بلی. در این صورت کلید جستجو شامل چند خصوصیت می‌شود

تعریف کلید جستجوی مرکب Composite Search Key : کلیدی جستجویی که ترکیبی از چند کلید باشد.

مثال)

For example, for the case of two attribute search keys, $(a_1, a_2) < (b_1, b_2)$ if either $a_1 < b_1$ or $a_1 = b_1$ and $a_2 < b_2$.

سوال) آیا در ساختار شاخص مقادیر خصیصه‌های غیر از کلید جستجو را می‌توان نگهداری کرد؟ بلی.

تعریف شاخص پوششی Covering Index: اگر در ساختار شاخص به غیر از کلید جستجو بقیه خصوصیت‌ها را نیز ذخیره کنیم. شاخص پوششی خواهیم داشت

سوال) فایده شاخص پوششی چیست؟ شاخص پوششی دسترسی به مقادیر خصیصه‌هایی غیر از کلید جستجو را نیز تسریع می‌کند.

3 فایل شاخص B⁺-tree

در پایگاه‌داده‌های خیلی بزرگ، فایل شاخص خیلی بزرگ می‌شود. فایل شاخصی که تا اینجا دیدیم یک آریه مرتب است. با بزرگ شدن این آریه زمان پویش scan خیلی بزرگ خواهد شد. برای رفع این مشکل می‌توان از داده‌ساختار B⁺-tree کمک گرفت.

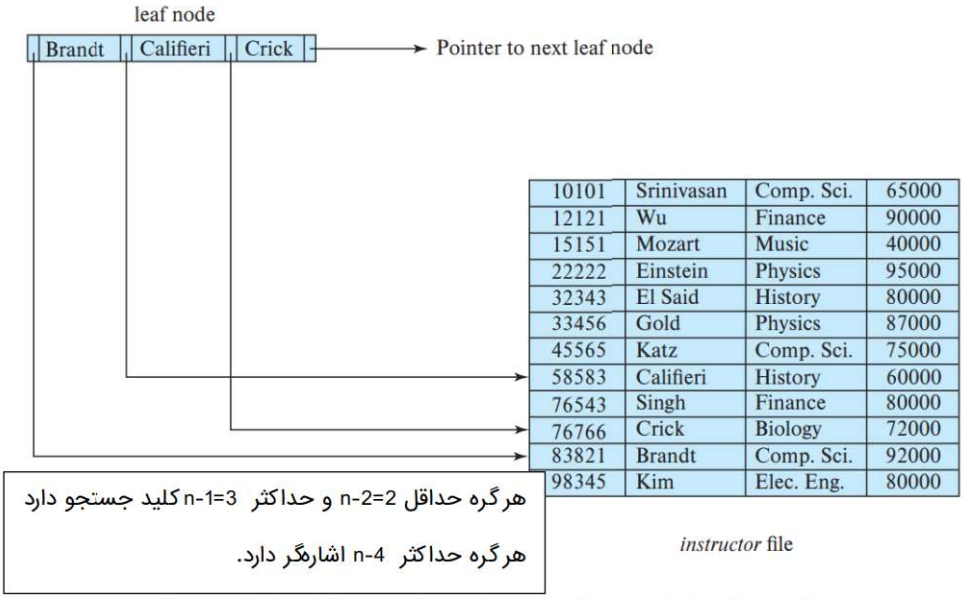


Figure 14.8 A leaf node for *instructor* B⁺-tree index ($n = 4$).

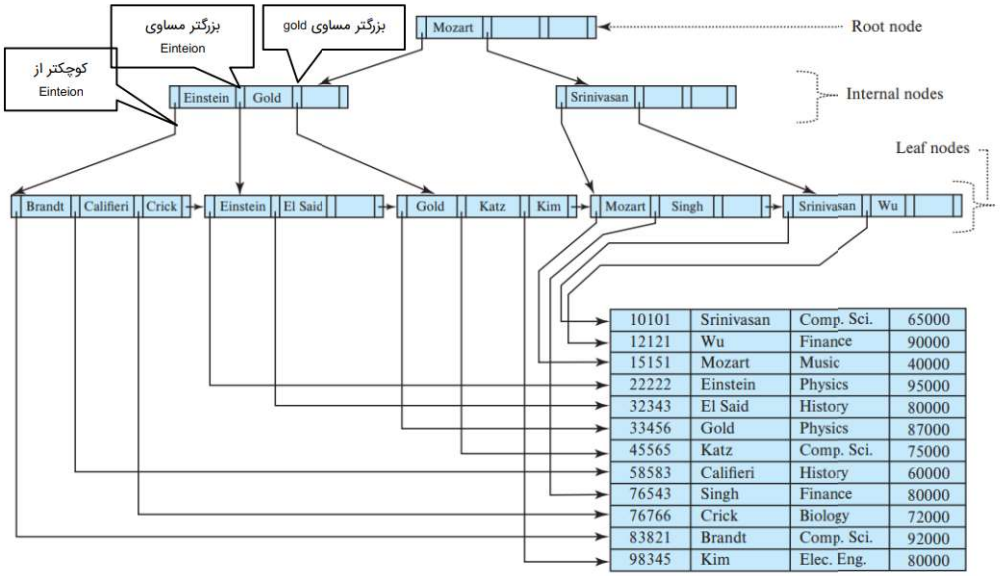


Figure 14.9 B^+-tree for *instructor* file ($n = 4$).

سوال) اگر K رکورد داشته باشیم و درختی با عدد n حداکثر مسیری که برای جستجو طی می‌شود چند گره است؟ $Log_{n/2}(K)$

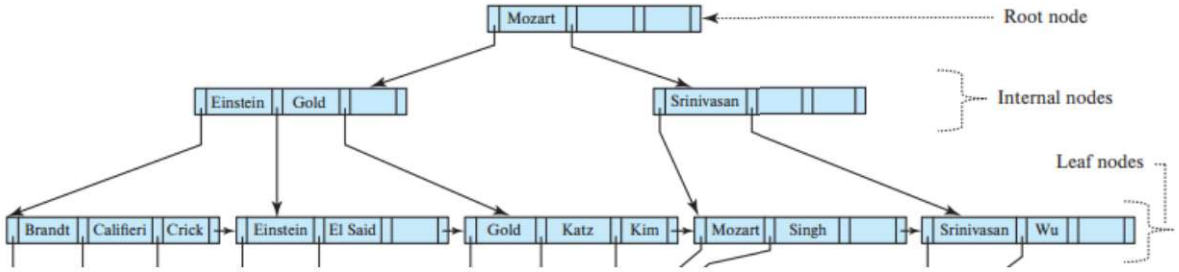
نکته) فایل شاخص B^+-tree نوع خاصی از شاخص‌گذاری چندلایه است.

نکته) مهمترین و جالب‌ترین ویژگی B^+-tree خودتوازنی self-balanced است. این ویژگی باعث پیچیدگی این ساختار در مقایسه با فایل شاخص ترتیبی و یا حتی B-tree شده است.

3.1 درج

حفظ توازی یا همان خودتوازنی باعث پچیده‌شدن عملیات درج یک رکورد در B^+-tree شده است.

می‌خواهیم Adams را در درخت وارد کنیم.



در این حالت Adams باید قبل از Brandt در گره برگ وارد شود. چون گره برگ جا ندارد آن را به دو گره تبدیل می‌کنیم.

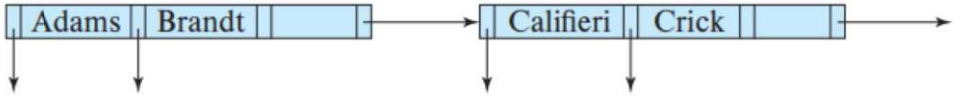


Figure 14.13 Split of leaf node on insertion of "Adams".

و برای اینکه تعداد جستجوها زیاد نشود. مجبوریم که گره داخلی بالایی را بروزرسانی کنیم.

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.
استفاده از این جزوه بدون خرید فیلم‌ها غیرقانونی و غیرشرعی است.

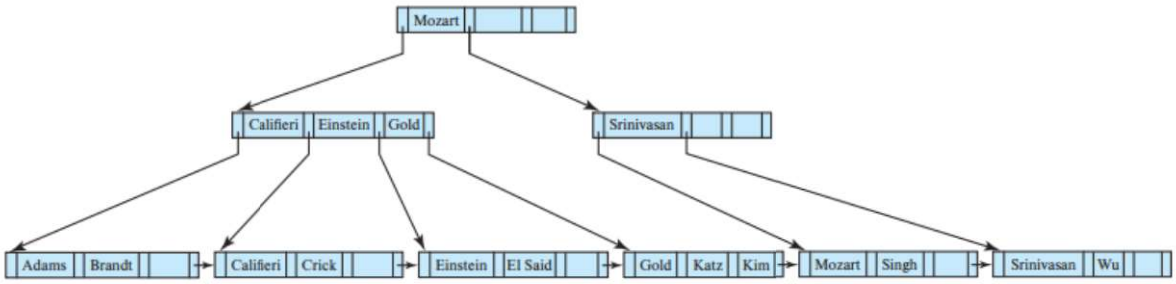


Figure 14.14 Insertion of "Adams" into the B+-tree of Figure 14.9.

اضافه کردن Lamport حتی از اضافه کردن Adams هم سخت‌تر است و مجبوریم که یک گره داخلی در لایه بالایی اضافه کرده و حتی ریشه هم تغییر می‌کند.

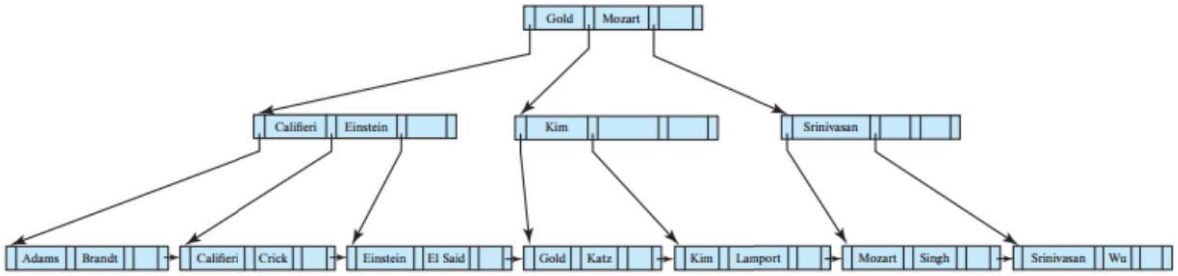
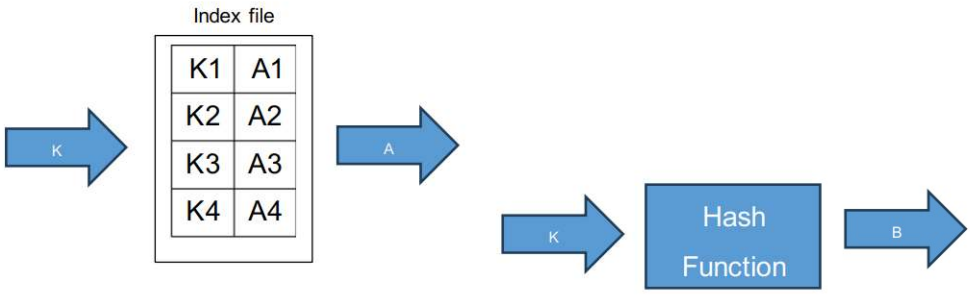


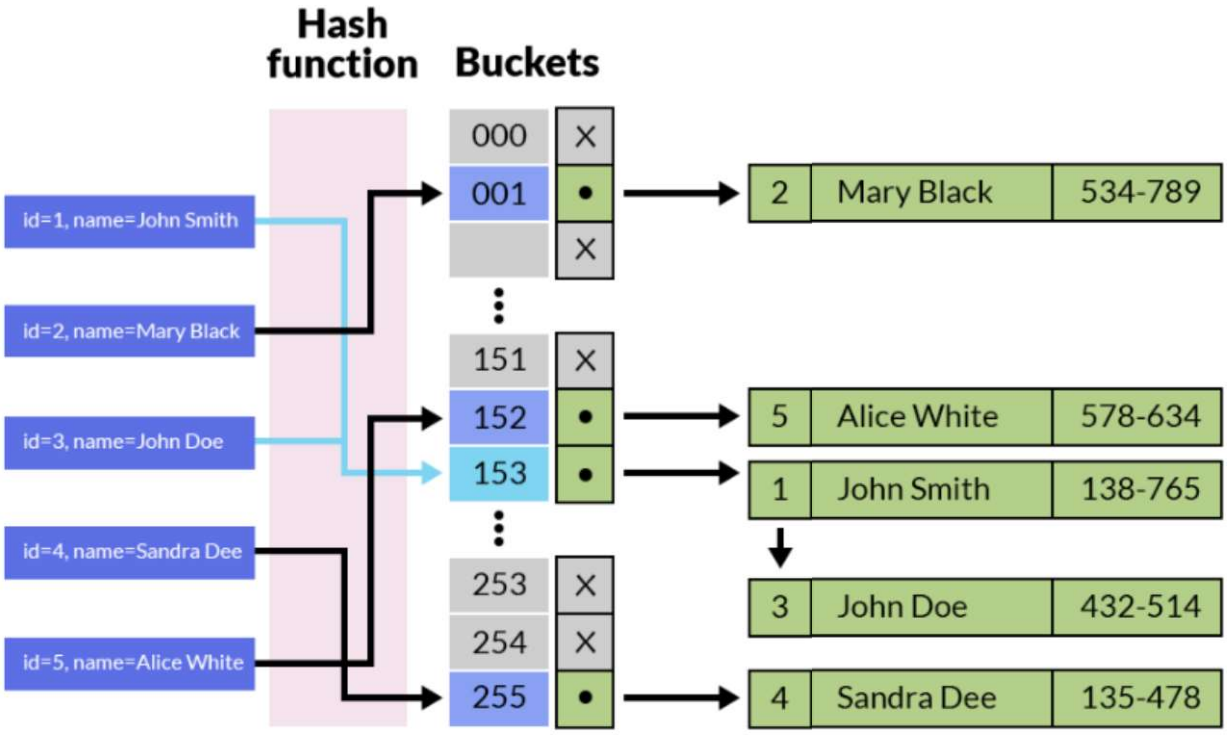
Figure 14.15 Insertion of "Lamport" into the B+-tree of Figure 14.14.

4 شاخص‌گذاری هش

در فایل شاخص ترتیبی ما یک آرایه مرتب داشتیم. وقتی مقدار کلید جستجو K مورد نظر در ستون کلید جستجو پیدا می‌کردیم در روبروی آن اشاره‌گر یا آدرس رکورد A را می‌توانستیم پیدا کنیم. درخت +B روش مشابهی داشت یعنی دوتایی (K,A) ذخیره شده و قابل بازیابی بود. در شاخص‌گذاری هش Hash Indexing این دوتایی ذخیره نشده است. در شاخص گذاری هش، کلید جستجو با تابع هش به آدرس یک باکت bucke تبدیل می‌شود. باکت فضایی است که در آن می‌توان یک یا چند رکورد را ذخیره کرد. با مراجعه به آن باکت می‌توان رکورد مورد نظر را پید کرد.



این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید.
استفاده از این جزوه بدون خرید فیلم‌ها غیرقانونی و غیرشرعی است.



تعریف تابع هش Hash Function: تابع هش یک کلید جستجو K را گرفته و یک آدرس باکت B را برای ما تولید می‌کند. B=h(K)

سوال) فرض کنید ظرفیت باکت یک رکورد باشد، اگر تابع هش برای دو تا کلید یک آدرس تولید کند، چه اتفاقی می‌افتد؟ وقتی ظرفیت باکت تکمیل می‌شود و جا برای رکورد جدید نداشته باشیم، مشکل باکت سرریز Bucket Overflow پیش آمده است. برای حل این مشکل می‌توان از زنجیره سرریز Overflow Chain استفاده کرد. یعنی رکوردهایی که در یک باکت قرار میگرند با یک لیست پیوندی به هم زنجیر می‌شوند. به این روش آدرس دهی بسته یا closed addressing هم گفته می‌شود.

تعریف باکت سرریز Bucket overflow : وقتی تعداد رکوردهای که باید در یک باکت ذخیره شود از ظرفیت آن باکت بیشتر باشد.

دلایل

اگر ظرفیت هر باکت را بزرگتر از یک بگیریم سرریز باکت دیرتر اتفاق می‌افتد ولی باز هم محتمل است. مثلا اگر به جای یک رکورد بتوانیم ۴ رکورد در یک باکت ذخیره کنیم. احتمال این که تابع هش برای پنج کلید یک آدرس را تولید کند وجود دارد.

یکی از سخت‌ترین کارها پیدا کردن تابع هش خوب است. ولی تابع هش خوب چیست؟

سوال) اگر تابع هش را به گونه‌ی زیر تعریف کنیم که حرف اول اسم استاد را به یکی از اعداد ۱ تا ۲۶ نسبت دهیم. H(A)=1 و H(B)=2 و... چه مشکلی به وجود می‌آید؟

پاسخ) چون نام‌های انگلیسی که با بعض از حروف شروع می‌شوند مثلا S و A از برخی از حروف دیگر مثلا X و Y خیلی بیشتر است. باکت بعضی از حرف‌ها خیلی زود پر می‌شود در حالیکه بقیه باکتها خالی می‌ماند. یکی از ویژگیهای تابع هش خوب توزیع یکنواخت Uniform است. ویژگی دوم یک تابع هش خوب تصادفی Random بودن است.

این جزوه برای استفاده به همراه فیلم‌های آموزشی آماده شده است. برای درک بهتر حتما فیلمها را هم مشاهده کنید. استفاده از این جزوه بدون خرید فیلم‌ها غیرقانونی و غیرشرعی است.

نکته) وقتی تابع هش توزیع یکنواخت نداشته باشد گفته می‌شود توزیع نامتقارن skew دارد.

انتخاب تابع هش چالشهای دیگری نیز دارد. اگر تابع هش را به گونه‌ای تعریف کنیم که هر اسم استاد را به عددی بین ۱ تا ۱۰۰۰۰۰۰ تبدیل کند چه مشکلی به وجود می‌آید؟

پاسخ) با انتخاب این تابع هش باید ۱۰۰۰۰۰۰ باکت داشته باشیم. با این تعداد باکت احتمال سرریز باکت خیلی کم خواهد بود. ولی در یک دانشکده معمولی با ۲۰۰ تا استاد خیلی از باکت‌ها خالی و بی‌استفاده خواهند ماند. این به معنای اتلاف فضای ذخیره‌سازی یا حافظه است. انتخاب تابع هش مناسب که نه حافظه زیادی را هدر بدهد و نه مشکل سرریز باکت داشته باشد کار سختی است.

سوال) دلایل مشکل سرریز باکت چیست؟ ۱. تابع هش نامتقارن ۲. تعداد باکتهای کم

تعریف هش ایستا Static Hash: اگر بر اساس برآورد حجم پایگاه‌داده تعداد باکت‌ها و تابع هش قبل از ثبت اطلاعات مشخص کرده باشیم به این هش ایستا می‌گویند.

در هش ایستا با بزرگترشدن پایگاه داده از برآورد اولیه دو رویکرد ممکن است

1. عدم تغییر تابع هش و تعداد باکتها. در این رویکرد زنجیره باکتها طولان شده و باعث افت کارآیی میشود. زیرا مثلا اگر یک زنجیره ۱۰۰ رکورد داشته باشد. وقتی تابع هش از روی کلید آدرس باکت را داد باید در زنجیره آنقدر جلو برویم تا به رکورد مورد نظر برسیم. یعنی ۱۰۰ رکورد را بخوانیم.
2. بروزرسانی دوره‌ای تعداد باکتها و تابع هش. این کار سربار پردازش سنگینی دارد.

برای حل این معضل استفاده از هش پویا Dynamic Hash پیشنهاد شده است. این روش همزمان با رشد پایگاه‌داده ساختار شاخص را بروزرسانی کرده و از افت کارایی و اتلاف حافظه پیش‌گیری می‌نماید.
Extendable hashing یک نمونه از هش پویا است.

4.1 مقایسه روش‌های شاخص گذاری

سه روش شاخص‌گذاری معرفی شد. دقت کنید هیچ کدام از ین روش‌ها بهترین نیستند. بلکه هر کدام مزایا و معایبی دارند و در بعضی از کاربردها مناسب‌تر هستند. به طور کلی معیارهای مقایسه روش‌های شاخص گذاری عبارتند از.

1. نوع دسترسی access type: در دسترسی نقطه‌ای point query ما به دنبال یک رکورد خاص هستیم. در دسترسی بازه‌ای range query ما به دنبال مجموعه‌ای از رکوردها که در یک بازه range قرار می‌گیرند هستیم.

سوال) کدام نوع روش شاخص‌گذاری برای دسترسی نقطه‌ای مناسب‌تر است؟ هش

سوال) کدام روش شاخص‌گذاری برای دسترسی بازه‌ای مناسبت‌تر است؟ شاخص‌گذاری‌های مرتب مانند فایل شاخص ترتیبی و B^+-tree

2. زمان دسترسیaccess time : زمان لازم برای دسترسی به یک رکورد با بازه‌ای از رکوردها

سوال) زمان دسترسی کدام روش کمتر است B^+-tree یا فایل شاخص ترتیبی؟ B^+-tree

سوال) زمان دسترسی کدام روش کمتر است شاخص خوشه‌گذاری خلوت یا متراکم؟ متراکم

3. زمان درجinsertion time: زمان مورد برای درج یک رکورد جدید. این زمان شامل زمان پیدا کردن جای رکورد جدید به علاوه زمان لازم برای بروزرسانی ساختار شاخص می‌شود
4. زمان حذف deletion = زمان پیدا کردن رکورد+ زمان حذف رکورد + زمان بروزرسانی ساختار شاخص

سوال) زمان درج و حذف در کدام روش بیشتر است B^+-tree یا فایل شاخص ترتیبی؟ B^+-tree. این روش به خاطر ویژگی خودتوازنی خیلی پیچیده است.

5. سرباره حافظه Space Overhead: پیاده‌سازی ساختار ایندکس به فضا نیاز دارد. این فضا در مقایسه با خود پایگاه‌داده کوچکتر است.

سوال) سرباره حافظه کدام روش کمتر است B^+-tree یا هش؟ هش

سوال) سرباره حافظه کدام روش کمتر است شاخص خوشه‌ای یا غیرخوشه‌ای؟ خوشه‌ای

نکته: شاخص‌گذاری به طور کلی موجب تحمیل سرباره حافظه، افزایش زمان درج، افزایش زمان حذف می‌شود. ولی ما این نکات منفی را به خاطر افزایش سرعت دسترسی تحمل می‌کنیم.

5 شاخص Bitmap

کاربرد این شاخص پرس‌وجوهای چند خصیصه‌ی است.

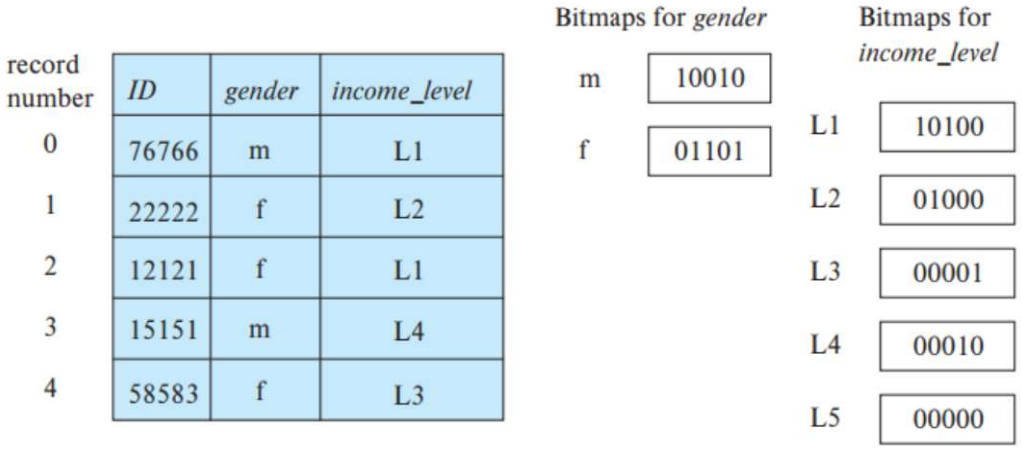


Figure 14.28 Bitmap indices on relation *instructor_info*.

سوال) زنانی که حقوق در سطح L2 دارند؟ f=01101 را با L2=01000 و منطقی کنید. نتیجه می‌شود 01000 یعنی رکورد شماره ۱ یا ID=22222 رکورد مورد نظر است.

سوال) تعداد افرادی که حقوی در سطح L1 دارند؟ تعداد یک های L1=10100 را بشمارید. عدد ۲ می‌شود. پس دو استاد حقوق در سطح L2 دارند.

۱. در صورتی که زمان خواندن کل رکوردهای یک جدول به ترتیب کلید اصلی ملاک ارزیابی باشد، DBMS باید کدام یک از ساختارهای ذیل را در سطح داخلی پایگاه داده مورد استفاده قرار دهد؟ (فناوری اطلاعات ۱۳۸۳)

۱) Indexed- Sequential file ۲) Hash file

۳) Multi Indexed file ۴) Pile file

پاسخ گزینه ۱ این سوال بیش از آنکه به درس پایگاه‌داده باشد. به درس ذخیره و بازیابی اطلاعات مربوط است. و در ویرایش‌های جدید منابع درس پایگاه داه کمتر به آنها پرداخته شده است.

در صورت سوال تاکید کرده است خواندن کل رکوردها به ترتیب کلید اصلی. در فایل ترتیبی شاخص‌گذاری شده همه رکوردها به ترتیب در حافظه ذخیره شده اند و در کمترین زمان همه رکوردها را می‌توان خواند.

1 وابستگی چندمقداری MVD

با رسیدن به BCNF تمامی افزونگی‌های ناشی از FD از بین می‌رود. سوالی که مطرح می‌شود آیا هنوز افزونگی دیگری باقی مانده است؟

مثال) رابطه *inst* (*ID, dept name, name, street, city*) اطلاعات اساتید یک دانشگاه را ذخیره می‌کند.

IDشماره شناسایی استاد، dept_name نام دانشکده،name نام استاد، street خیابان محل سکونت استاد و city شهر محل سکونت استاد را نشان می‌دهد.

سوال)با فرض اینکه هر استاد فقط در یک دانشکده درس می‌دهد. رابطه inst در چه نرمال فرمی است.

پاسخ) BCNF. راه حل هر استاد در یک دانشکده درس می‌دهد. یک محل سکونت دارد. اینکه شماره شناسایی استاد هم تکراری نباشد بدیهی است. پس ID→ dep_name, name, street, city. یعنی ID کلید کاندید است. یک کلید کاندید بیشتر نداریم. طبق تعریف فرم BCNF در تنها FD رابطه سمت چپ سوپر کلید است و در فرم BCNF هستیم.

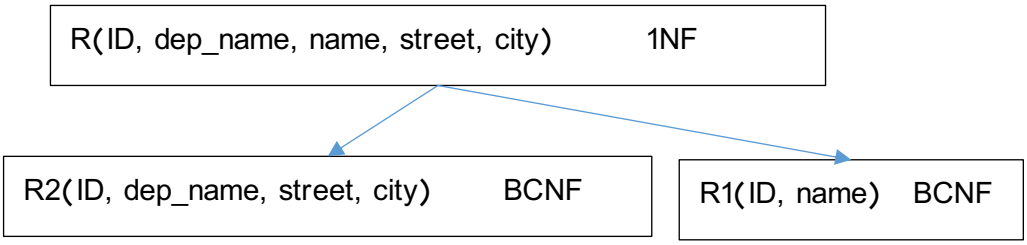
سوال)حالا فرض کنید. هر استاد می‌تواند در چند دانشکده درس بدهد. با این فرض جدید رابطه در کدام فرم قرار می‌گیرد؟

پاسخ)non-BCNF با این تغییر در صورت مساله FD به شکل ID→name, street, city در می‌آید. در این FDسمت چپ سوپر کلید نیست. زیرا از ID به dept_name نمی‌رسیم. پس قطعاً BCNF نیست.

برای تصمیم گیری درباره فرم این رابطه باید کلید کاندید را پیدا کنیم. ID جنسیس است ولی به تنهایی همه خصوصیت‌ها را تولید نمی‌کند. داریم ID, dept_name → dept_name, street, city. پس ID, dept_name کلید کاندید است. به خاطر ID→name وابستگی بخشی داریم. پس 2NF نیست. پایان.

حالا فرض کنید. هر استاد علاوه بر این که در چند دانشکده درس می‌دهد. می‌تواند چند آدرس محل سکونت داشته باشد. با اینکار ID→street, city هم نخواهیم داشت. و تنها ID→name را داریم. برای اینکه دوباره BCNF باشیم. از تجزیه استفاده می‌کنیم.

R1 (*ID, name*) *R2* (*ID, dept name, street, city*)



سوال) آیا این تجزیه مشکل گم‌شدگی و از دست رفتن وابستگی دارد؟

پاسخ) خیر. از ضوابط ریسانن استفاده می‌کنیم. ID ستون مشترک است. پس ضابطه ۱ صحیح است. ID در r1 کلید کاندید است ضابطه ۲ برقرار است. تنها وابستگی R، ID→Name در R1 حفظ شده‌است.

سوال)R1 در چه فرمی است؟

پاسخ)BCNF.

سوال ۲) R2 در چه فرمی است؟

پاسخ)BCNF. رابطه R2 تمام کلید است. پس BCNF است.

سوال) آیا در R2 ما افزونگی داده داریم؟

پاسخ)بلی

ID	Dept_name	street	City
22	ECE	Kaj	Tehran
22	Math	Kaj	Tehran
22	ECE	Bahar	Bam
22	Math	Bahar	Bam
33	ECE	Kaj	Tehran
33	Literature	Kaj	Tehran
33	Physics	Kaj	Tehran
33	ECE	Bahar	Bam
33	Literature	Bahar	Bam
33	Physics	Bahar	Bam

خیابان و شهر محل سکونت استاد چند بار تکرار شده است. این افزونگی داده به خاطر FD نیست به خاطر نوع جدیدی از وابستگی به نام وابستگی چند مقداری multi valued dependency یا MVD است.

تعریف) وابستگی چند مقداری . در رابطه R(X, Y, Z) با صفات ساده یا مرکب X، Yو Z می‌گوییم که Y به X وابستگی چند مقداری (MVD) دارد و با نمایش می‌دهیم X→→Y، اگر به یک مجموعه مقادیر متناظر با یک مقدار از جفت (X,Z) در R، فقط به مقدار X بستگی داشته باشد و وابسته به مقدار Z نباشد. به عبارت دیگر مجموعه مقادیر Y فقط به تغییرات X تغییر کند.

تعریف) تعریف ساده وابستگی چند مقدار. در رابطه R(X, Y, Z) می‌گوییم X→→Y اگر مجموعه‌ای از مقادیر Y متناظر با X باشد.

در مثال قبل، ID→→Dept_name زیرا اگر بدانیم ID=22 می‌توان نتیجه گرفت dept_name = {ECE, Math} یعنی یکی از مقادیر ECE یا Math را دارد. دقت کنید که ID اینجا تعیین کننده است نه street, city. و به همین شکل اگر بدانیم ID=33 میتوان نتیجه گرفت که dept_name یکی از مقادیر ECE، Literature یا Physics خواهد بود.

سوال) MVD چگونه آنومالی ایجاد می‌کند؟

پاسخ) فرض کنید این اطلاعات را می خواهیم درج کنیم (22, Physics, Kaj, Tehran) یعنی استاد با شماره ۲۲ در دانشکده فیزیک می‌خواهد درس بدهد. مجبوریم که یک تاپل هم برای آن آدرس دیگر استاد اضافه کنیم. یعنی (22, Physics, Bahar, Bam) . این آنومالی به هنگام درج است.
نکته) MVD همیشه زوج است. یعنی اگرX→→Y آنگاه X→→(R-(X,Y))

در R2 در کنار ID→→Dept_name داریم ID→→ (street, city)

نکته) در حالات زیر X→→Y بدیهی (نامهم) (Trivial) است.
اول)Y ⊂ X
دوم)X U Y = R
سوم) Yمجموعه تهی باشد.

نکته) X→Y FD حالت خاصی از وابستگی چند مقداری X→→Y که در آن مجموعه مقادیر Y، تک عنصری است.

نکته) X→Y => X→→Y

نکته) مشابه FD برای MVD قواعد آرمسترانگ داریم. در رابطه R(X, Y, Z, ..)

1. اگر Y⊂ X آنگاه X→→Y
2. اگرX→→Y و Y→→Z آنگاه X→→Z-Y
3. اگر X→→Y آنگاه X→→(R(H)-(X,Y))
4. اگر X→Y آنگاه X→→Y
5. اگر X→→Y و X→→Z آنگاه X→→(Y,Z)
6. اگر X→→Yو X→→W آنگاه (Y,Z)→→(W-(Y,Z))
7. اگر X→→Y و W ⊂Z آنگاه (X,W)→→(Y,Z)
8. اگر X→→(Y,Z) آنگاه X→→(Y ∩ Z) و X→→Y-Z و X→→Z-Y
9. اگر X→→Y و Z→W و W ⊂ Y و Z∩Y = ∅ آنگاه X→W

1.1.1.1.1.1 تست سراسری فناوری اطلاعات 1387
اگر وابستگی چند مقداری به شکل x→→y برقرار باشد و همچنین داشته باشیم w ⊃v آنگاه طبق قانون افزایشی augmentation می‌توان درستی کدام گزینه زیر را نتیجه گرفت؟

۱) vx→→wy ۲) vx→→vy ۳) wx→→vy ۴) wx→→wy

پاسخ) گزینه ۳ صحیح است.

راه حل تشریحی) اگر جدول آرمسترانگ MVD را به خاطر داشته باشید به راحتی از روی آن خواهید دید. که پاسخ گزینه ۳ است. اگر X→→Y و Z⊂ W آنگاه (Y,Z)→→(X,W).

راه حل تستی) فرض می‌کنیم شما جدول را از بر نیستید. اگر wx→→wy و vx→→vy را به عنوان درست باشند. یعنی در MVD همان قاعده افزایشی FD را داریم. پس هر دو باید درست باشند. زیرا به دلخواه به دو طرف هر خصیصه‌ای را می‌توان اضافه کرد و از این نظر wx→→wy با vx→→vy تفاوتی ندارد. پس این دو حذف می کنیم.

گزینه ۱ به دلیل زیر غلط است. در حالت خاص فرض کنید v=∅ باشد. از این می‌توان نتیجه گرفت که x→→ wy که کاملاً مشخص است غلط است. پس گزینه ۱ هم حذف می‌شود. تنها گزینه باقیمانده ۳ است.

2 نرمال فرم چهارم 4NF

تعریف) تعریف ساده نرمال فرم چهارم. رابطه R ، 4NF است اگر در BCNF باشد و در آن MVD غیر بدیهی (non trivial) نداشته باشیم.

تعریف) تعریف دقیق 4NF. یک رابطه 4NF است اگر و فقط اگر به ازای همه $\alpha \rightarrow\rightarrow \beta$ یکی از دو شرط زیر برقرار باشد.

اول) $\alpha \rightarrow\rightarrow \beta$ بدیهی باشد. یعنی $\beta \subset \alpha$

دوم) α ابرکلید باشد.

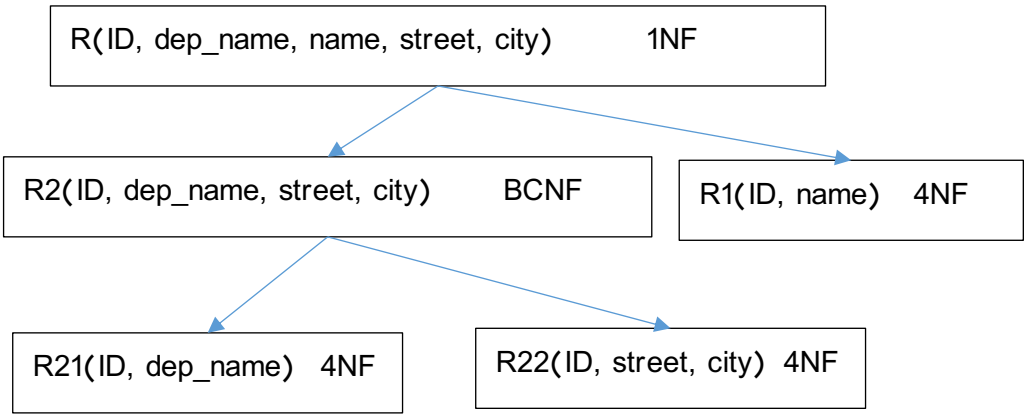
سوال) رابطه R1(ID, name) آیا 4NF است؟

پاسخ) بلی.

قبلا دیدیم که R1 در BCNF است.

در ID→→name ، ID سوپر کلید است. شرط دوم برقرار است. همین برای 4NF بودن کافی است و می‌توان جلوتر نرفت.

علاوه بر آن ID→→name زیر {ID, name} = ID U name تمامی خصوصیتهای R1 را می‌دهد پس. ID→→name بدیهی هم هست. شرط اول هم برقرار است.



سوال)رابطه $R2\ (ID, dept\ name, street, city)$ آیا 4NF است؟

پاسخ)نیست. زیرا همانطور که دیدیم ID→→dep_name بدیهی نیست و ID هم سوپرکلید نیست پایان.

سوال) رابطه $R2\ (ID, dept\ name, street, city)$ را به دو رابطه R21(ID, dept_name) و R22(ID, street, city) تجزیه می‌کنیم. آیا R21 و R22 در فرم 4NF هستند.

پاسخ)بلی. هم R21 و هم R22 تمام کلید هستند. پس BCNF هستند.

ID→→Dept_name در R21 وقتی از دو طرف اجتماع می‌گیریم به تمام خصیصه‌ها می‌رسیم پس بدیهی است. پس R21 در 4NF است.

ID→→street, City در R22 وقتی از دو طرف اجتماع می‌گیریم به تمام خصیصه‌ها می‌رسیم. پس بدیهی است. پس R22 در 4NF است.

R21

ID	Dept_name
22	ECE
22	Math
33	ECE
33	Literature
33	Physics

R22

ID	street	City
22	kaj	Tehran
22	Bahar	Bam
33	Kaj	Tehran
33	Bahar	Bam

R2

ID	Dept_name	street	City
22	ECE	kaj	Tehran
22	Math	Kaj	Tehran
22	ECE	Bahar	Bam
22	Math	Bahar	Bam
33	ECE	Kaj	Tehran
33	Literature	Kaj	Tehran
33	Physics	Kaj	Tehran
33	ECE	Bahar	Bam
33	Literature	Bahar	Bam
33	Physics	Bahar	Bam

اگر اصرار دارید که ID→→street, city در R22 بدیهی نیست می‌توان تجزیه را یک مرحله دیگر تکرار کرد.

برای رسیدن به 4NF هم تجزیه باید خوب و بی مشکل باشد. قضیه فاگین از دست نرفتن اطلاعات را به هنگام تجزیه تضمین می‌کند.

قضیه فاگین (Fagin) رابطه R(X, Y, Z) به دو پرتو R1(X, Y) و R2(X, Z) تجزیه بدون گم‌شدگی می‌شود اگر و فقط اگر X→→Y

سوال) قضیه فاگین شبیه کدام قضیه بود؟

پاسخ)قضیه هیث برای FD

نکته) قضیه هیث حالت خاصی از قضیه فاگین برای FD است.

سوال) آیا در تجزیه R2 به R21 و R22 گم‌شدگی داریم؟

پاسخ)خیر. در R21 و R22 ستون مشترک ID داریم. در R21 داریم ID→→Department

نکته) در پروژه‌های عمل 4NF اصلاً مهم نیست. زیرا اگر رابطه‌ای در BCNF باشد ولی در 4NF نباشد به علت داشتن صفت‌های چند مقداری است. طراح برای پرهیز از این این گونه رابطه‌ها به راحتی می‌تواند از همان ابتدا صفتهای چند مقداری را در دو رابطه مجزا پیاده‌سازی کند. یعنی از همان ابتدا به جای R2، R21 و R22 را طراحی می‌کند. با این ترفند اصلاً درگیر وابستگی چندمقداری و آنومالی‌ها آن نخواهیم شد. و با رسیدن رابطه‌ها به BCNF به طور خودکار 4NF هم خواهند شد. با این همه 4NF از دیدگاه نظری و طراحان تست‌های کنکور می‌تواند جذاب باشد! 😁😁

2.1.1.1.1.1 تست سراسری مهندسی کامپیوتر 86

اگر جدولی در فرم نرمال BCNF باشد، ممکن است در کدام فرم نرمال دیگر نباشد؟

۱) 1NF ۲) 2NF ۳) 3NF ۴) 4NF

پاسخ) گزینه ۴ است. پاسخ به این سوال نیاز به درس MVD و 4NF ندارد. تنها باید به یاد داشته باشید.

$4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$

1 حساب رابطه‌ای تاپلی

برای نوشتن کوئری در پایگاه‌داده رابطه‌های زبان‌های مختلفی وجود دارد.

مثلا SQL، جبررابطه‌ای RA، حساب رابطه‌ای تاپلی Tuple Relational Calculus، حساب رابطه‌ای دامنه‌ای Domain Relational Calculus

در این فصل زبان حساب رابطه‌ای تاپلی TRC را برای نوشتن کوئری معرفی می‌کنیم.

نکته) TRC با نوشتن شرط یک تعداد از سطرها (تاپل‌های) جدول را استخراج می‌کند به خاطر همین حساب رابطه‌ای تاپلی نامیده می‌شود.

نکته) TRC بر خلاف RA روالی نیست. در RA ما عملیات‌هایی که باید انجام دهیم تا به پاسخ برسیم را می‌نویسیم. در TRC به عنوان یک زبان غیرروالی ما پاسخ موردنظر را توصیف می‌کنیم.

تعریف1 . متغیر تاپلی. متغیری است که مقادیر مجازش تاپل‌های رابطه است.

مثلاً عبارت زیر متغیر تاپلی BX را تعریف می‌کند که می‌تواند مقادیر سطرها (تاپل‌های) رابطه Buy را بگیرد.

Rangevar BX Ranges Over Buy

برای کمک به یادگیری سریع‌تر همان کوئری‌های همیشگی را این بار با TRC می‌نویسیم.

Student(s_name, city, uni)

course(c_name, t_name)

Buy(s_name, c_name, tr, amount)

Attendance (s_name, c_name, meeting, hour)

نکته) کلمات کلیدی با رنگ آبی مشخص شده‌اند.

1.1 پرس و جوی 1

کلیه مشخصات خریدهای انجام شده دانشجویان

پاسخ در جبر رابطه ای Buy است.

شکل اول TRC)

```
Rangevar BX Ranges Over Buy
BX
```

شکل دوم TRC) با استفاده از سورهای عمومی

$\{t \mid t \in Buy\}$

1.2 پرس‌وجوی 2

نام کلیه دانشجویانی که دوره ها را خریداری کرده اند. در این مثال ما یک ستون را انتخاب می‌کنیم.

```
Select s_name from Buy
```

$\Pi_{s\_name}(buy)$

شکل اول)

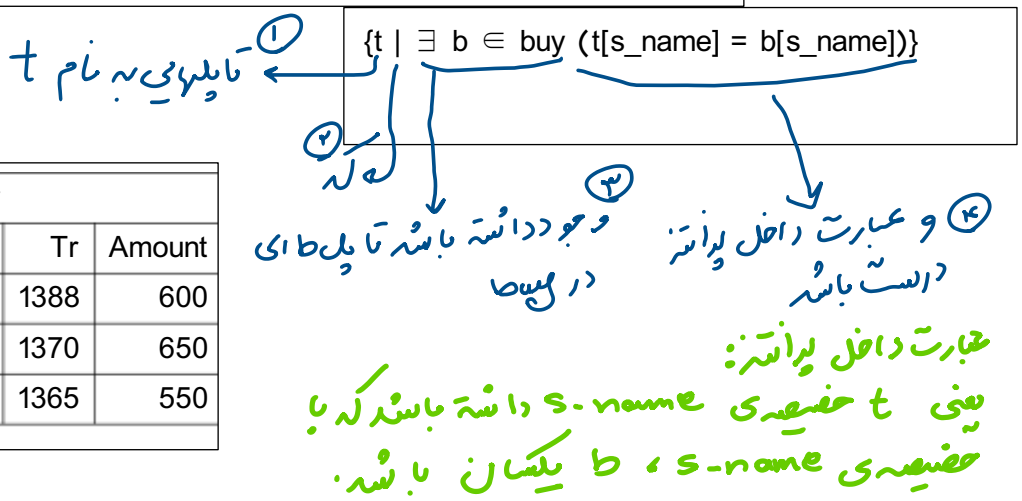
```
Rangevar BX Ranges Over Buy
Bx. s_name
```

شکل دوم)

نکته) وقتی قرار نیست تمام ستونهای را در خروجی داشته باشیم مجبوریم که حتماً یک متغیر رابطه‌ای جدید تعریف کنیم. نمی توانیم بنویسیم $\{s \mid \exists\ s \in instructor\ (s[s\_name])\}$. این از نظر TRC غلط است.

{t | ∃ b ∈ buy (t[s_name] = b[s_name])}

نام دانشجویانی که به ازای آنها تاپلی در جدول buy داریم



Buy			
s_name	c_name	Tr	Amount
Reza	DB	1388	600
Ali	AI	1370	650
Negin	DB	1365	550

1.3 پرس‌وجوی سوم

مثال) نام کلیه افرادی که در دوره DB شرکت کرده اند. در این مثال چند سطر را انتخاب می‌کنیم.

```
Select s_name from  att where c_name = 'DB'
```

$\Pi_{s\_name}(\sigma_{c\_name = 'DB'}(att))$

شکل اول)

```
Rangevar AX Ranges Over att
Ax. s_name
Where Ax.c_name = 'DB
```

شکل دوم)

{t | ∃ a ∈ att (t[s_name] = a[s_name] ∧ a[c_name] = ‘DB’)}

{t | ∃ a ∈ att (t[s_name] = a[s_name] نام تمامی دانشجویانی که تاپلی در ات دارند

∧ a[c_name] = ‘DB’)}و در آن تاپل نام کورس دی بی است

Attendance			
s_name	c_name	Meeting	Hour
Negin	DB	1	4
Negin	DB	2	2
Reza	AI	1	4
Negin	AI	1	3
Sara	DB	1	3

1.4 پرس‌وجوی چهارم

مثال) نام کلیه افرادی که در برای خرید دوره DB کمتر از ۶۰۰ تومان خرج کرده‌اند. در این مثال شرط کمی پیچیده‌تر شده است.

```
select s_name from buy where c_name = DB  and amount < 600
```

$$\Pi_{\text{s\_name}} (\sigma_{\text{c\_name = 'DB' ^ amount<600}} (\text{buy}))$$

شکل اول)

```
Rangevar BX Ranges Over buy
Bx. s_name
Where Bx.c_name = ‘DB’ AND Bx.amount < 600
```

شکل دوم)

{t | ∃ b ∈ buy (t[s_name] = b[s_name] ∧ b[c_name] = ‘DB’ ∧ b[amount] < 600)}

{t | ∃ b ∈ buy (t[s_name] = b[s_name] نام دانشجویانی که تاپلی برای آنها در جدول بای هست

∧ b[c_name] = ‘DB’ و نام درس این تاپل دی بی است

∧ b[amount] < 600)}و هزینه این تاپل کمتر از ۶۰۰ است

Buy			
s_name	c_name	Tr	Amount
Reza	DB	1388	600
Ali	AI	1370	650
Negin	DB	1365	550

یادآوری) برای نوشتن پرس و جو یا کویری سه مرحله باید انجام شود

1- فهم پرس و جو
2- شناسایی جدول یا جدولهای مرتبط با پرس و جو
3- انتخاب عبارت‌های مناسب برای جستجو

1.5 پرس‌وجوی پنجم

مثال) پیداکردن نام و دانشگاه دانشجویانی که درس DB را خریده‌اند. در این مثال دو جدول باید با هم الحاق شوند.

شکل اول)

```
Rangevar SX Ranges Over student
Rangevar BX Ranges Over buy
Sx. s_name, Sx.uni
Where Exists BX (BX.s_name = SX.s_name AND BX.c_name = ‘DB’)
```

توضیح:

Sx. s_name, Sx.uni نام و دانشگاه

Where Exists BX (BX.s_name = SX.s_name دانشجویانی که تاپلی برای آنها در جدول بای وجود دارد

و نام کورس در این تاپل برابر دی بی است(AND BX.c_name = ‘DB’

شکل دوم)

{a | ∃ s ∈ student (a[s_name] = s[s_name] ∧ a[uni] = s[uni] ∧

∃ b ∈ buy (a[s_name] = b[s_name] ∧ b[c_name] = ‘DB’))}

توضیح

{a | ∃ s ∈ student (a[s_name] = s[s_name] نام دانشجویانی که تاپلی برای آنها در جدول استیودنت هست

∧ a[uni] = s[uni] و نام دانشگاه آنها در همان تاپل هست

∧ ∃ b ∈ buy (a[s_name] = b[s_name] و وجود دارد تاپلی برای آنها در جدول بای

∧ b[c_name] = ‘DB’))} که در این تاپل اسم درس دی بی است

1.6 پرس‌وجوی ششم

پیداکردن دانشجویانی که همه درس‌ها را خریده‌اند. عملیات تقسیم.

شکل اول)

Rangevar SX Ranges Over student

Rangevar CX Ranges Over course

Rangevar BX Ranges Over buy

دانشجویانی کهSx

Where forall CX (Exixts BX برای همه کورس‌ها (وجود) داشته باشد تاپل خریدی

BX.s_name = SX.s_name AND BX.c_name = CX.c_name)) که نام دانشجو و نام کورس در آن ذکر شده باشد.

شکل دوم)

{ s | s ∈ student ∧ تاپل دانشجویانی که

∀ c ∈ course (∃ b ∈ buy برای هر کورس (وجود) داشته باشند تاپل خریدی

(s.s_name = b.s_name ∧

c.c_name = b.c_name))} که نام دانشجو و نام کورس در آن ذکر شده باشد.

شکل سوم)

{ s | s ∈ student ∧ تاپل دانشجویانی که

∀ c ((c ∈ course) => برای هر کورس نتیجه بگیریم

∃ b (b ∈ buy ∧ s.s_name = b.s_name ∧

c.c_name = b.c_name))} وجود دارد تاپل خریدی برای آن کورس به نام آن دانشجو

2 حساب رابطه‌ای دامنه‌ای

با حساب رابطه‌ای دامنه‌ای DRC، ما باز هم کوئری می‌نویسیم. این کوئری بر اساس شرط تعدادی از سطرها و ستون‌های جدول را استخراج می‌کند.

مهمترین تفاوت DRC و TRC در نوشتن شرط است. در DRC شرط‌ها با شرط عضویت Membership Condition نوشته می‌شود.

2.1 شرط عضویت

R(A1:x1, A2:x2, …)

که در آن R رابطه، Ai نام صفت و xi مقداری از میدان صفت Ai است. این شرط درست True ارزیابی می‌شود اگر و فقط اگر تاپلی در رابطه R با مقادیر داده شده وجود داشته باشد.

Buy(s_name:'Reza', c_name:'DB', Tr:1388, Amount:600) = True

درست ارزیابی می‌شود چون این سطر را در جدول داریم

Buy(s_name:'Ali', c_name:'DB', Tr:11365, Amount:600) = False

نادرست ارزیابی می‌شود چون چنین سطری در جدول نداریم.

Buy			
s_name	c_name	Tr	Amount
Reza	DB	1388	600
Ali	AI	1370	650
Negin	DB	1365	550

تعریف 2. متغیر دامنه‌ای. به x1و x2و… در شرط عضویت متغیر دامنه‌ای گفته می‌شود.

2.2 پرس و جوی 1

کلیه مشخصات خریدهای انجام شده دانشجویان

پاسخ در جبر رابطه ای Buy است.

شکل اول DRC)

(Buy)

شکل دوم DRC) با استفاده از سورهای $\exists$ و $\forall$

{<s,c,t,a> | $\exists$ s,c,t,a (<s,c,t,a > $\in$ Buy)}

2.3 پرس‌وجوی 2

نام کلیه دانشجویانی که دوره ها را خریداری کرده اند. در این مثال ما یک ستون را انتخاب می‌کنیم.

```
Select s_name from Buy
```

Π_{s_name} (buy)

شکل اول)

SNx Where Exist Buy (s_name:SNx)

SNx یک متغیر دامنه‌ای و Buy(s_name:SNx) یک گزاره از نوع شرط عضویت است. در این عبارت، SNx باید مقادیری را بگیرد که شرط عضویت درست True شود. اگر SNx هر کدام از مقادیر Reza و Ali و Negin را بگیرد. شرط درست می شود. پس خروجی ستون s_name خواهد بود.

Buy			
s_name	c_name	Tr	Amount
Reza	DB	1388	600
Ali	AI	1370	650
Negin	DB	1365	550

نکته) می‌توان کلمه کلیدی Exist را حذف کرد و نوشت

SNx Where Buy (s_name:SNx)

شکل دوم)

{s | ∃ c, t, a (<s, c, t, a> ∈ Buy)}

توضیح ۱) دقت کنید که s از s_name ، c از c_name ، t از Tr و a از amount مقدار می‌گیرد.
می‌خوانیم s هایی که به ازای آنها وجود داشته باشد c, t, a که تاپل <s, c, t, a> در Buy باشد.
دقت کنید که در این عبارت s, c, t, a همگی متغیرهای دامنه‌ای هستند و با متغیر تاپلی تفاوت دارند.

توضیح ۲)s هایی را می‌خواهیم که در مؤلفه اول یکی از تاپل‌های buy باشند.

2.4 پرس‌وجوی سوم

مثال) نام کلیه افرادی که در دوره DB شرکت کرده اند. در این مثال چند سطر را انتخاب می‌کنیم.

```
Select s_name from  att where c_name = 'DB'
```

$\Pi_{s\_name}\ (\ \sigma_{c\_name = 'DB'}(att))$

شکل اول)

SNx Where att (s_name:SNx, c_name:'DB')

توضیح ۱) SNx یک متغیر دامنه‌ای است که از att.s_name می‌تواند مقدار بگیرد. کدام مقادیر از att.s_name مقادیری که به ازای آنها شرط att(s_name: SNx, c_name:'DB') درست باشد. به ازای (Negin, DB, 1, 4) و (Negi, DB, 2, 2) این شرط درست است. پس تنها مقداری که SNx می‌تواند بگیرد Negin خواهد بود که این خروجی کوئری است.

توضیح ۲)

SNx Where att (s_name:SNx)

نام همه دانشجویان که حضور داشته اند

SNx Where att (s_name:SNx, c_name:'DB')

نام همه دانشجویانی که در کلاس DB حضور داشته اند

Attendance			
s_name	c_name	Meeting	Hour
Negin	DB	1	4
Negin	DB	2	2
Reza	AI	1	4
Negin	AI	1	3

شکل دوم)

{s | ∃ m, h (<s, ‘DB’, m, h> ∈ att)}

توضیح ۱) دقت کنید که سه متغیر دامنه‌ای داریم. S مقدار گیرنده از s_name ، m مقدار گیرنده از meeting، h مقدار گیرنده از hour. می‌خوانیم s هایی که به ازای آنها وجود داشته باشد m و h که تاپل <s, ‘DB’, m, h> در buy موجود باشد. تنها مقدار Negin است.

توضیح ۲) s هایی را می‌خواهم که مولفه دوم DB باشد. مقدار مؤلفه سوم و چهارم هم مهم نیست.

توضیح ۳)

{s | ∃ m, h, d (<s, d, m, h> ∈ att)}

نام دانشجویانی که حضور داشته اند.

{s | ∃ m, h (<s, ‘DB’, m, h> ∈ att)}

نام دانشجویانی که در کلاس DB حضور داشته‌اند.

2.5 پرس‌وجوی چهارم

مثال) پیداکردن دانشگاه دانشجویانی که درس DB را خریده‌اند. در این مثال دو جدول باید با هم الحاق شوند.

شکل اول)

Ux Where Exists SNx (Student (s_name:SNx, uni : Ux) AND Buy (s_name:SNx, c_name:’DB’))

- متغیر دامنه‌ای SNx از s_name در جدول Student و جدول Buy وظیفه الحاق دو جدول را برعهد دارد. دقت کنید چون SNx هم در student و هم در buy ظاهر شده است. پس تنها

تاپل‌هایی به خروجی می‌رود که student.s_name = buy.s_name = SNx باشد. و به عبارت دیگر الحاق‌پذیر باشند.

- Ux از uni مقدار می‌گیرد.
- این گونه بخوانید Ux هایی که وجود داشته باشد SNx هایی که عبارت

Student (s_name:SNx, uni : Ux) AND Buy (s_name:SNx, c_name:'DB')= True بشود.

شکل دوم)

{u | ∃ s, c, t, a (<s,c,u> ∈ student ∧ <s, 'DB', t, a> ∈ Buy) }

توضیح) u هایی که برای آن تاپلی در student داشته باشیم. به این تاپل تاپل اول می گوییم. برای تاپل اول تاپلی در buy باشد که در مؤلفه اول با هم مشترک باشند. به این تاپل دوم می‌گوییم. و تاپل دوم حتماً باید مولفه دومش DB باشد.

2.6 پرس‌وجوی پنجم

نام و شهر دانشجویانی که هیچ کدام از دوره‌های دکتر خانی را نخریده‌اند.

شکل اول)

(Sx, Cx)

Where Exists Sx (Student(s_name:Sx, City:Cx)

AND NOT (Buy (s_name:Sx, c_name:CCX) AND course (c_name:CCX, t_name:'Khani)))

توضیح)

(Sx, Cx)

نام و نام شهر

Where Exists Sx (Student(s_name:Sx, City:Cx)

دانشجویانی که

AND NOT (Buy (s_name:Sx, c_name:CCX) AND course (c_name:CCX, t_name:'Khani)))

کورسی از دکتر خانی نخریده باشند.

شکل دوم)

{<s,c>| ∃ u <s,c,u> ∈ student ∧ ¬ ∃ cc,t,a (<s,cc,t,a>∈ buy ∧ <cc,'khani'> ∈ course)}

توضیح)

{<s,c>| ∃ u <s,c,u> ∈ student

نام و شهر همه دانشجویان

{<s,c>| ∃ u <s,c,u> ∈ student ∧ ¬ ∃ cc,t,a (<s,cc,t,a>∈ buy

نام و شهر دانشجویانی که برای آنه خریدی وجود ندارد. یعنی کورسی خرید نکرده اند.

{<s,c>| ∃ u <s,c,u> ∈ student ∧ ¬ ∃ cc,t,a (<s,cc,t,a>∈ buy ∧ <cc,'khani'> ∈ course)}

نام و شهر دانشجویانی که برای آنها وجود ندارد خریدی و آن خرید از دکتر خانی باشد.

یعنی نام و شهر دانشجویانی که از کورسهای دکتر خانی خرید نکرده اند.

3 قدرت توصیف DRC و TRC

تعریف 3. عبارت مطمئن. Safe. یک عبارت حسابی مطمئن است هر گاه نتیجه ارزیابی آن تعداد محدودی از تاپل‌ها باشد.

مثلا عبارت در {s | ¬ (s ∈ student)} ما بی نهایت ترکیب داریم که جزو سه تایی Student(s_name, city, uni) نیست. پس این عبارت مطمئن نمی‌باشد.

ولی عبارت {s | (s ∈ student)} چون تعداد محدودی سطر در جدول student داریم و s یکی از آنهاست در عمل محدود است. و این عبارت مطمئن محسوب می‌شود.

در جبر رابطه‌ای نمی‌توان عبارت نامطمئن نوشت. ولی این امکان در DRC و TRC وجود دارد. اگر عبارات نامطمئن را از DRC و TRC حذف کنیم. قدرت توصیفی جبر رابطه‌ای با DRC و TRC برابر می‌شود. یعنی

- هر آنچه با جبر رابطه‌ای می‌توان توصیف کرد می‌توان با DRC توصیف کرد و بالعکس.
- هر آنچه با جبر رابطه‌ای می‌توان توصیف کرد می‌توان با TRC توصیف کرد و بالعکس.

با در نظر گرفتن عبارات مطمئن قدرت توصیفی DRC و TRC از جبر رابطه‌ای بیشتر می‌شود. یعنی

- TRC کوئری‌های را توصیف می‌کند که نمی‌توان با جبررابطه‌ای نوشت.
- DRC می‌تواند کوئری‌های را توصیف می‌کند که نمی‌توان با جبررابطه‌ای نوشت.

نکته) در عمل از TRC و DRC استفاده نمی‌شود. هر چند از ایده‌های آن در زبان‌های کوئری نویسی استفاده شده است. مثلاً زبان کوئری نویسی QBE Access از DRC الهام گرفته است.

نکته) در کنکور از DRC و TRC خیلی کم سوال می‌آید.