به نام آنکه جان را فکرت آموخت

بخش هشتم:
طراحی پایگاه داده رابطه‌ای

مرتضی امینی

نیمسال دوم ۹۴-۹۵

(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)

- در **طراحی پایگاه داده‌های رابطه‌ای** باید موارد زیر را مشخص نمود:
  - مجموعه‌ای از رابطه‌ها
  - کلید(های) کاندید هر رابطه
  - کلید اصلی هر رابطه
  - کلیدهای خارجی هر رابطه (در صورت وجود)
  - محدودیت‌های جامعیتی ناظر بر هر رابطه

- **روشهای طراحی RDB:**
  - طراحی با روش بالا به پایین (Top-Down)
  - طراحی با روش سنتز [نرمال‌ترسازی رابطه‌ها]

طراحی پایگاه داده رابطه‌ای (ادامه)

- **روش طراحی بالا به پایین**
  - ابتدا مدلسازی داده‌ها را (با روش [E]ER یا UML) انجام می‌دهیم و سپس مدلسازی را به مجموعه‌ای از رابطه‌ها تبدیل می‌کنیم.
- **روش طراحی سنتز رابطه‌ای (نرمال‌ترسازی)**
  - ابتدا مجموعه صفات خرد جهان واقع را مشخص می‌کنیم. سپس با تحلیل قواعد و محدودیت‌های ناظر به صفات و تشخیص وابستگی‌های بین آنها، صفات را متناسباً با هم سنتز می‌کنیم (نوعی گروه‌بندی) تا به مجموعه‌ای از رابطه‌های نرمال دست یابیم.
  - در عمل روش ترکیبی استفاده می‌شود، یعنی ابتدا روش بالا به پایین، سپس نرمال‌ترسازی.

ویژگی‌های طراحی خوب

- نمایش صحیح و واضح از خردجهان واقع باشد.
- تمام داده‌های کاربران قابل نمایش باشد و همه محدودیت‌های (قواعد) جامعیتی منظور شده باشد.
- کمترین افزونگی
- کمترین هیچمقدار
- کمترین مشکل در عملیات ذخیره‌سازی
- بیشترین کارایی در بازیابی

تامین چهار ویژگی آخر به صورت همزمان، در عمل ناممکن است!

طراحی بالا به پایین

- تبدیل نمودار [E]ER به مجموعه‌ای از رابطه‌های نرمال (و نه لزوماً در نرمال‌ترین صورت) در طراحی RDB، نهایتاً طراح تصمیم می‌گیرد چند رابطه داشته باشد و عنوان (Heading) هر رابطه چه باشد.
- در نمودار مدلسازی معنایی داده‌ها، حالات متعدد داریم، که در ادامه به آنها می‌پردازیم.
- **فرض:** تا اطلاع ثانوی، همه صفات ساده‌اند و موجودیت‌ها ضعیف نیستند.

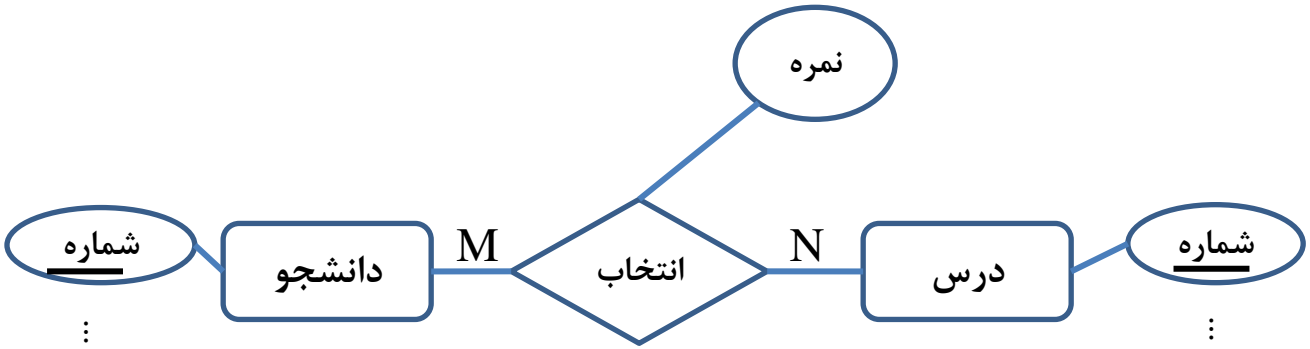
حالت ۱: طراحی ارتباط چند به چند

حالت ۱

- درجه ارتباط: n=2
- چندی ارتباط: M:N

سه رابطه لازم است.

- طراحی در این حالت با کمتر از سه رابطه، افزونگی و هیچ‌مقداری زیادی پدید می‌آورد.



STUD (STID, ….)

COR (COID, ….)

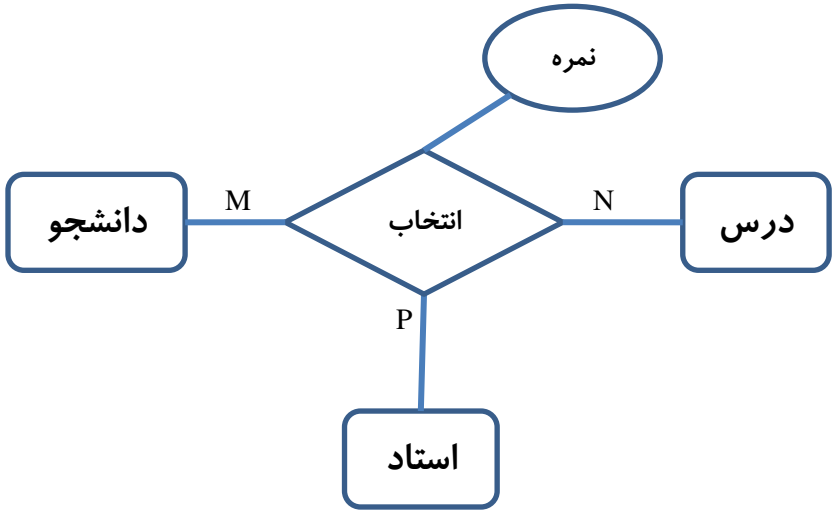
SCR (STID, COID, GR)

حالت ۱: طراحی ارتباط چند به چند (ادامه)

تعمیم حالت ۱

- درجه: $n>2$
- ابتدا فرض می‌کنیم چندی رابطه M:N:P:... است.
- $n+1$ رابطه طراحی می‌کنیم.
- سپس بررسی می‌کنیم که آیا محدودیت خاصی روی چندی ارتباط بین بعض موجودیت‌ها وجود دارد.
- اگر بله، این محدودیت را در مرحله نرمالترسازی دخالت می‌دهیم. ← تعداد رابطه‌ها ممکن است بیش از $n+1$ شود.

حالت ۱: طراحی ارتباط چند به چند (ادامه)



STUD (STID, ….)

COR (COID, ….)

PROF (PRID, ….)

SCP (STID, COID, PRID, GR)

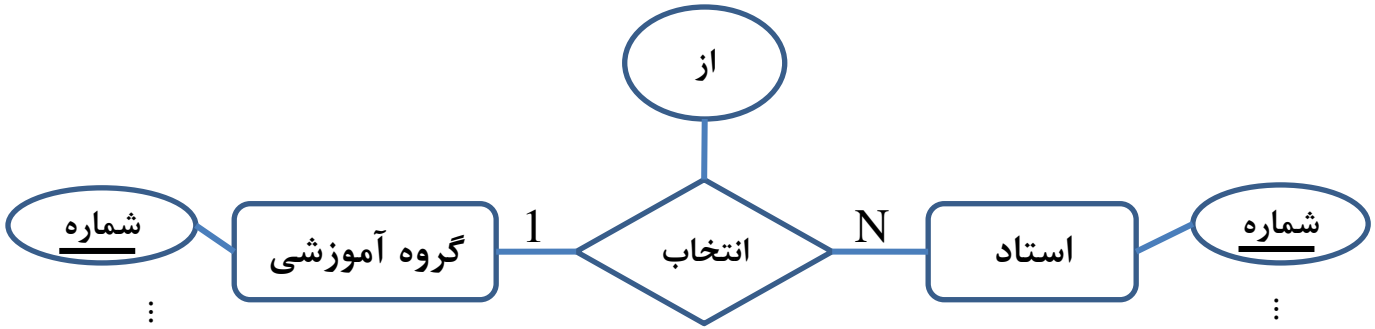
- فرض برای محدودیت: یک استاد فقط یک درس را تدریس می‌کند (البته در این مورد، چندی رابطه دقیق مدل نشده که این محدودیت لحاظ نشده است).
  - در این صورت باید رابطه SCP را به دو رابطه (یا بیشتر) تجزیه عمودی کنیم.
- این محدودیت را در مرحله دوم طراحی (در مباحث آتی) دخالت می‌دهیم.

حالت ۲: طراحی ارتباط یک به چند

حالت ۲

- درجه ارتباط: n=2
- چندی ارتباط: 1:N

دو رابطه لازم است. رابطه سمت 1 به رابطه سمت N، FK می‌دهد (بیرون از کلید اصلی).



DEPT (DEID, DTID, …., DPHONE)

PROF (PRID, PRNAME, …., PRANK, DEID, FROM)

حالت ۲: طراحی ارتباط یک به چند (ادامه)

- **در چه وضعی طراحی این حالت با سه رابطه قابل توجیه است؟**

۱- وقتی که مشارکت سمت N در ارتباط غیرالزامی باشد (درصد مشارکت کمتر از ۳۰ درصد) و تعداد استاد زیاد باشد، برای کاهش مقدار Null، رابطه نمایشگر ارتباط را جدا می‌کنیم.

۲- فرکانس ارجاع به خودِ ارتباط بالا باشد و به صفات دیگر با فرکانس پایین‌تری احتیاج باشد.

۳- تعداد صفات خودِ ارتباط زیاد باشد و باعث زیاد شدن درجه ارتباط PROF شود.

- اگر مشارکت سمت N الزامی باشد، باید این محدودیت معنایی را از طریق هیچمقدارناپذیر بودن صفت کلید خارجی (با استفاده از NOT NULL) در رابطه نمایانگر نوع موجودیت سمت N، اعلام کرد.

حالت ۳: طراحی ارتباط یک به یک

حالت ۳

- درجه ارتباط: n=2
- چندی ارتباط: 1:1

با دو / یا سه/ یا یک رابطه طراحی می‌کنیم.

- در صورت طراحی با **دو** رابطه، رابطه مربوط به نوع موجودیت با مشارکت الزامی، FK می‌گیرد.

COUR (COID, ….., BKID)

BOOK (BKID, ….., BKPRICE)

حالت ۳: طراحی ارتباط یک به یک (ادامه)

- وقتی با **سه** رابطه توجیه دارد که مشارکت طرفین غیرالزامی باشد، تعداد شرکت‌کنندگان (نمونه‌ها) در ارتباط زیاد باشد، درصد مشارکت در رابطه ضعیف (کمتر از ۳۰٪) باشد و نیز ملاحظاتی در مورد فرکانس ارجاع.

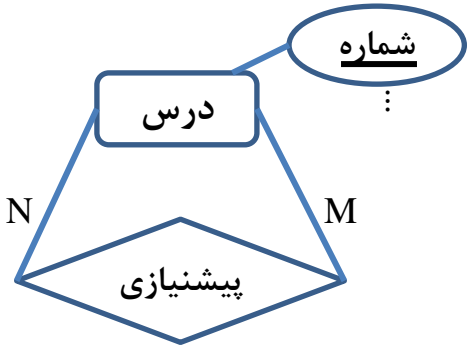
- وقتی با **یک** رابطه توجیه دارد که تعداد صفات موجودیت‌ها کم باشد، مشارکت طرفین الزامی باشد و فرکانس ارجاع به ارتباط کم باشد.

حالت ۴: طراحی ارتباط خود ارجاع چند به چند

حالت ۴

- حالت خاص حالت اول
- درجه ارتباط: n=1
- چندی ارتباط: M:N

دو رابطه لازم است.



COUR (COID, ….)

COPRECO (COID, PRECOID) → بیش از یک صفت از رابطه، از یک دامنه هستند.

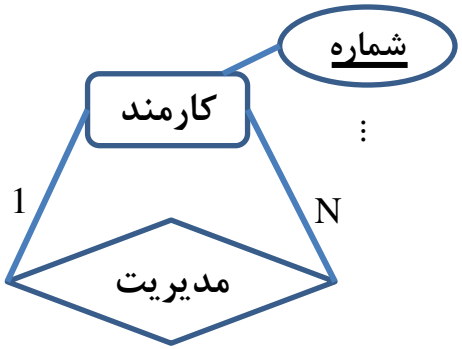
- **گراف ارجاع:** COPRECO ← COUR
- **نتیجه:** صرف وجود ارتباط با خود، چرخه ارجاع ایجاد نمی‌شود. باید به چندی ارتباط توجه کنیم.

حالت ۵: طراحی ارتباط خود ارجاع یک به چند

حالت ۵

- حالت خاص حالت دوم
- درجه ارتباط: n=1
- چندی ارتباط: 1:N

یک رابطه لازم است.



- در این رابطه چه نکاتی وجود دارد؟

EMPL (EMID, ENAME, …., EPHONE, EMGRID)

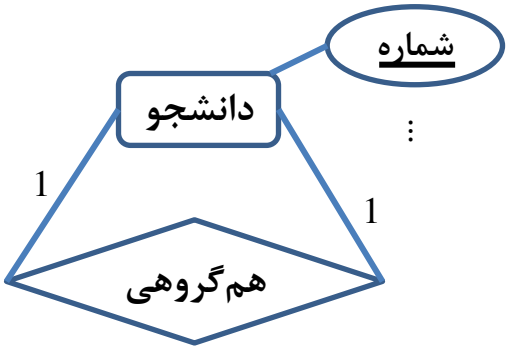
- **گراف ارجاع:** EMPL

- برنامه‌ای در SQL بدهید که سطح (مدیریتی) تمام مدیران در سلسله مدیریت را بدهد (با استفاده از تکنیک Recursion)

حالت ۶: طراحی ارتباط خود ارجاع یک به یک

حالت ۶

- حالت خاص حالت سوم
- درجه ارتباط: n=1
- چندی ارتباط: 1:1



با یک یا دو رابطه طراحی می‌کنیم.

- اگر مشارکت در هم‌پروژگی زیاد نباشد، از مدل II استفاده می‌کنیم.

(I) **STPROJST** (STID, STNAME, …., JSTID)
P.K. C.K.

(II) **STUD** (STID, STNAME, ….)

STJST (STID, JSTID)
C.K. C.K.

- در STJST هر یک از صفات می‌توانند کلید اصلی باشند.
- آیا طرز دیگری هم برای طراحی وجود دارد؟

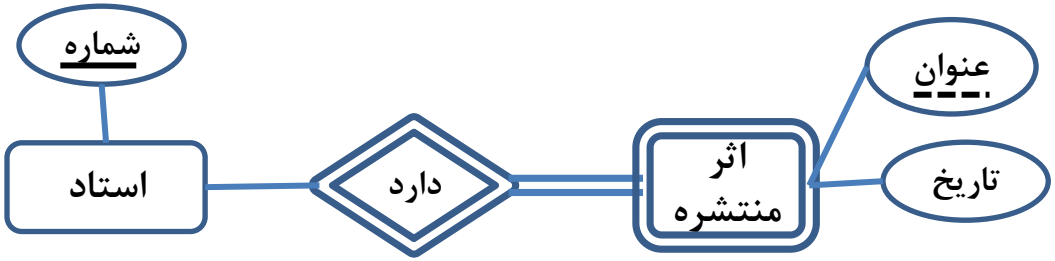
حالت ۷: طراحی موجودیت ضعیف

حالت ۷

- موجودیت ضعیف داریم.

دو رابطه لازم است؛ یکی برای نوع موجودیت قوی، یکی برای نوع موجودیت ضعیف و ارتباط شناسا.

رابطه نمایشگر موجودت ضعیف از موجودیت قوی FK می‌گیرد که در ترکیب با صفت ممیزه می‌شود PK.

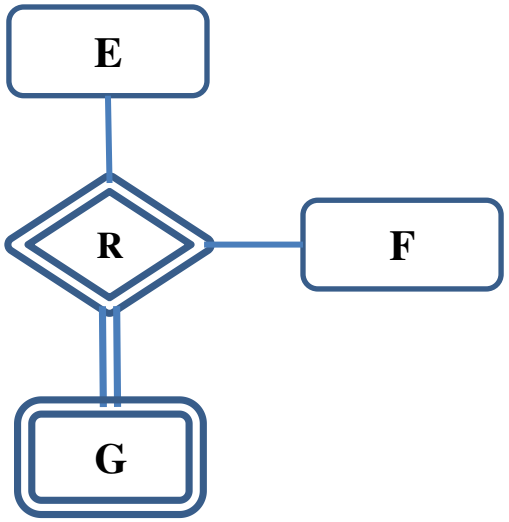


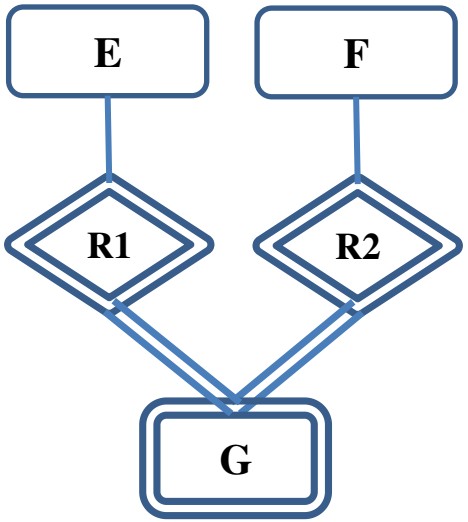
PROF (PRID, PRNAME, ….)

PRPUB (PRID, PTITLE, PTYPE, ….)

حالت ۷: طراحی موجودیت ضعیف (ادامه)

- **تمرین:** رابطه‌های لازم برای مدل‌های داده‌ای زیر طراحی شود.





در این حالت کلید رابطه G از ترکیب کلید رابطه‌های E و F (و در صورت وجود صفت ممیزه G) حاصل می‌گردد.

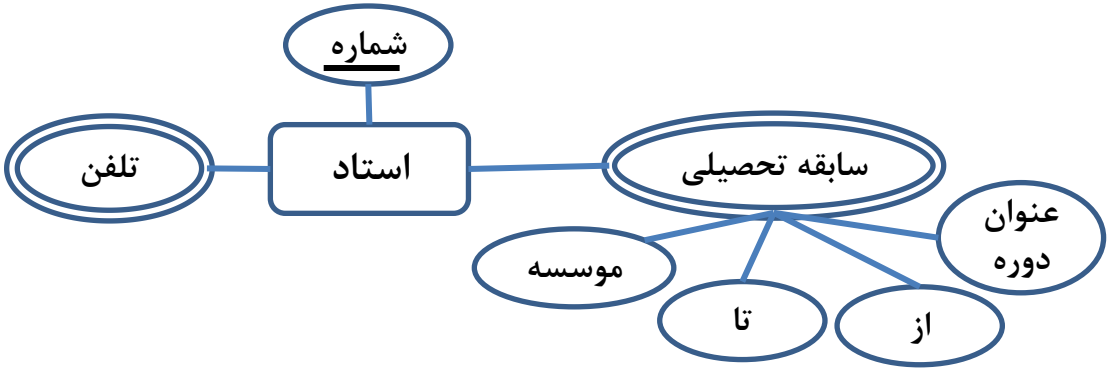
حالت ۸: طراحی صفت چندمقداری

حالت ۸

- وجود یک صفت چندمقداری برای یک نوع موجودیت.
- سه تکنیک دارد:

۱- [تکنیک عمومی] یک رابطه برای خود نوع موجودیت و یک رابطه برای هر صفت چندمقداری.

(بنابراین اگر نوع موجودیت E، m صفت چندمقداری داشته باشد، m+1 رابطه داریم.



✓ رابطه نمایشگر صفت چندمقداری از نوع موجودیت اصلی FK می‌گیرد داخل کلید.

PROF (PRID, PRNAME, ….)

PRTEL (PRID, PHONE)

حالت ۸: طراحی صفت چندمقداری (ادامه)

- در مدلسازی، موجودیت ضعیف به صفت چندمقداری ارجحیت دارد ولی تکنیک عمومی طراحی آنها مثل هم است.

PRHIS (PRID, TTL, FROM, TO, INSTNAME, ….)

- اشکال تکنیک عمومی: اگر برای نوع موجودیت اصلی اطلاعات کامل بخواهیم، باید عمل JOIN انجام دهیم که می‌تواند زمانگیر باشد.

حالت ۸: طراحی صفت چندمقداری (ادامه)

۲- [در شرایط خاص] طراحی با یک رابطه (فرض: یک صفت چندمقداری): یک رابطه برای خود نوع موجودیت و صفت چندمقداری.

- با فرض مشخص بودن حداکثر تعداد مقداری که صفت چندمقداری می‌گیرد، به همان تعداد صفت در رابطه در نظر می‌گیریم.

مثال

فرض: هر استاد حداکثر سه شماره تلفن دارد.

PRTELTEL (PRID, PRNAME, PRRANK, PHONE1, PHONE2, PHONE3)

- مزیت این تکنیک: JOIN لازم ندارد.
- عیب این تکنیک: هیچمقدار (Null) در آن زیاد است، اگر تعداد کمی از استادان، سه شماره تلفن داشته باشند.

حالت ۸: طراحی صفت چندمقداری (ادامه)

۳- [در شرایط خاص] طراحی با یک رابطه (یک رابطه برای خود نوع موجودیت و یک صفت چندمقداری) شامل تمام صفات نوع موجودیت و صفت چندمقداری.

دیگر صفات خود نوع موجودیت

PRTELTEL (PRID, PHONE, PRNAME, PRNAK, …)

- شرط اصلی استفاده: هر استاد حداقل یک تلفن داشته باشد.
- شرایط دیگری که بهتر است برقرار باشد: تعداد کمی از استادها بیش از یک تلفن داشته باشند (به دلیل افزونگی) و حتی‌الامکان تعداد صفات خود نوع موجودیت کم باشد (به دلیل افزونگی).

حالت ۹: طراحی ارتباط IS-A

حالت ۹

- وجود ارتباط IS-A بین دو نوع موجودیت.
- چهار تکنیک دارد:

۱- فرض: نوع موجودیت E، n زیرنوع دارد.

n+1 رابطه طراحی می‌کنیم. یک رابطه برای زبرنوع و یک رابطه برای هر یک از زیرنوع‌ها.

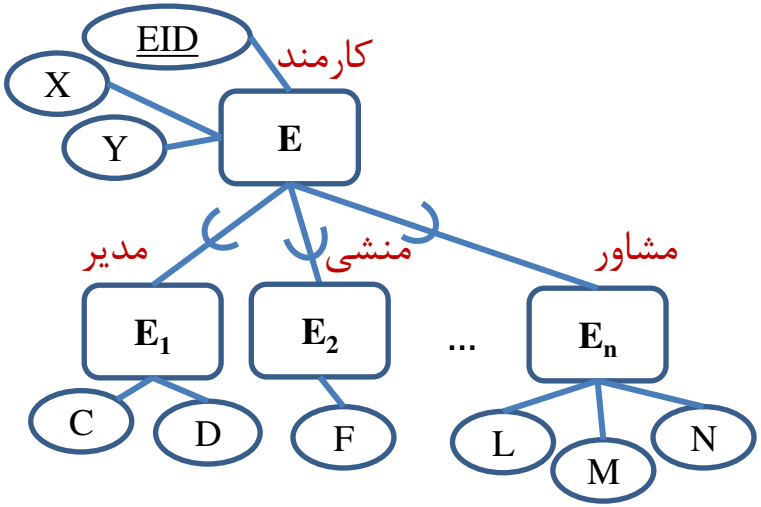
E (EID, X, Y)

E1 (EID, A, B)

E2 (EID, F)

…

En (EID, L, M, N)



حالت ۹: طراحی ارتباط IS-A (ادامه)

- مزیت این تکنیک: شرط خاصی از نظر نوع تخصیص ندارد (تکنیک‌های دیگری که مطرح می‌شود، همگی برای شرایط خاص هستند).
- عیب این تکنیک: اگر بخواهیم در مورد یک زیرنوع، اطلاعات کامل به دست آوریم، باید JOIN کنیم.

حالت ۹: طراحی ارتباط IS-A (ادامه)

۲- طراحی با n رابطه: برای **زیرنوع**، رابطه‌ای طراحی نمی‌کنیم. بنابراین صفات مشترک باید در رابطه نمایشگر هر **زیرنوع** وجود داشته باشد.

- شرط لازم: باید تخصیص کامل باشد. اگر نباشد، بخشی از داده‌های محیط قابل نمایش نیستند.

E1 (EID, X, Y, A, B)

E2 (EID, X, Y, F)

…

En (EID, X, Y, L, M, N)

- مزیت نسبت به تکنیک اول: برای به دست آوردن اطلاعات کامل زیرنوع‌ها نیازی به JOIN نیست.
- نکته: در این تکنیک، لزوماً افزونگی پیش نمی‌آید. اگر تخصیص هم‌پوشا باشد میزانی افزونگی پیش می‌آید.

حالت ۹: طراحی ارتباط IS-A (ادامه)

۳- طراحی فقط با یک رابطه، با استفاده از صفت نمایشگر نوع **زیرنوع‌ها**

- شرط استفاده از این تکنیک: تخصیص مجزا باشد؛ یعنی یک نمونه کارمند، جزء نمونه‌های حداکثر یک **زیرنوع** باشد.

E (EID, X, Y, A, B, F, L, M, N, TYPE)

100	x1	y1	a1	b1	?	?	?	?	مدیر
200	x2	y2	?	?	?	l2	m2	n2	مشاور

- مزیت این تکنیک: برای به دست آوردن اطلاعات کامل زیرنوع‌ها نیازی به JOIN نیست.
- عیب این تکنیک: هیچمقدار (Null) زیاد دارد و درجه رابطه زیاد است.

حالت ۹: طراحی ارتباط IS-A (ادامه)

۴- طراحی فقط با یک رابطه، با استفاده از آرایه بیتی؛ هر بیت نمایشگر نوع یک **زیرنوع**. در واقع برای نمایش هر نمونه موجودیت، بسته به اینکه در مجموعه نمونه‌های کدام زیرنوع باشد، بیت مربوطه‌اش را ۱ می‌کنیم.

- شرط استفاده از این تکنیک: وقتی تخصیص هم‌پوشا باشد (سایر شرایط همانها که در تکنیک ۳ گفته شد).

آرایه بیتی

E (EID, X, Y, A, B, F, L, M, N, TB1, TB2, …,TBn)

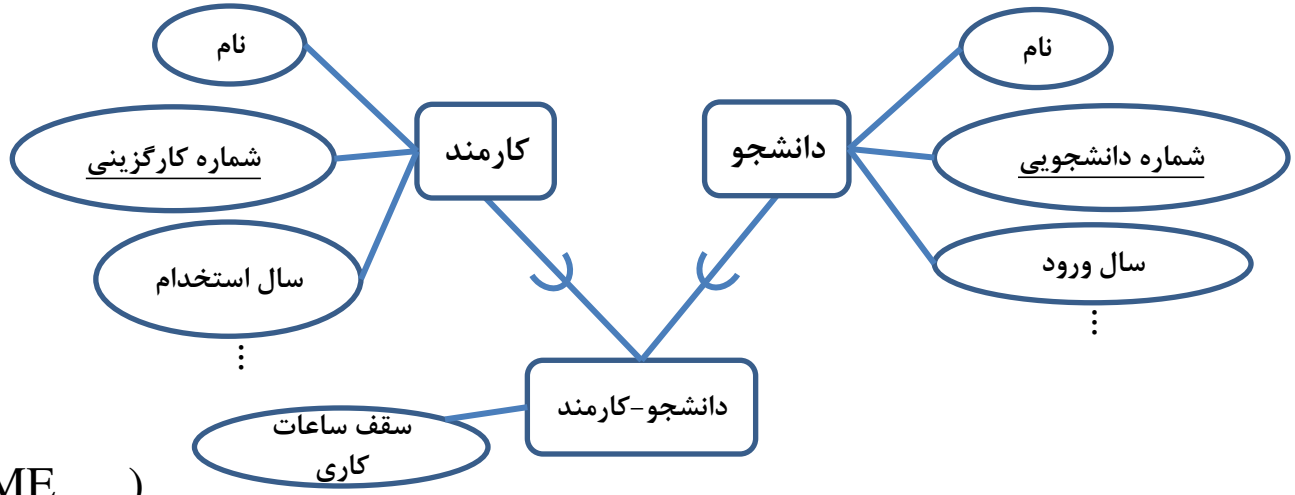
مدیر منشی مشاور

100	x1	y1	1	0	0
200	x2	y2	0	1	0

حالت ۱۰: طراحی ارث‌بری چندگانه

حالت ۱۰

- وجود ارث‌بری چندگانه بین یک زیرنوع با چندزبرنوع
- اگر زیـرنوع، n زبرنوع داشته باشد، رابطه نمایشگر زیـرنوع حداقل n کلید کاندید دارد. کلید کاندید با ارجاع بیشتر کلید اصلی انتخاب می‌شود.



STUD (STID, STNAME, …)

EMPL (EID, ENAME, …)

STEM (STID, EID, MAXW)

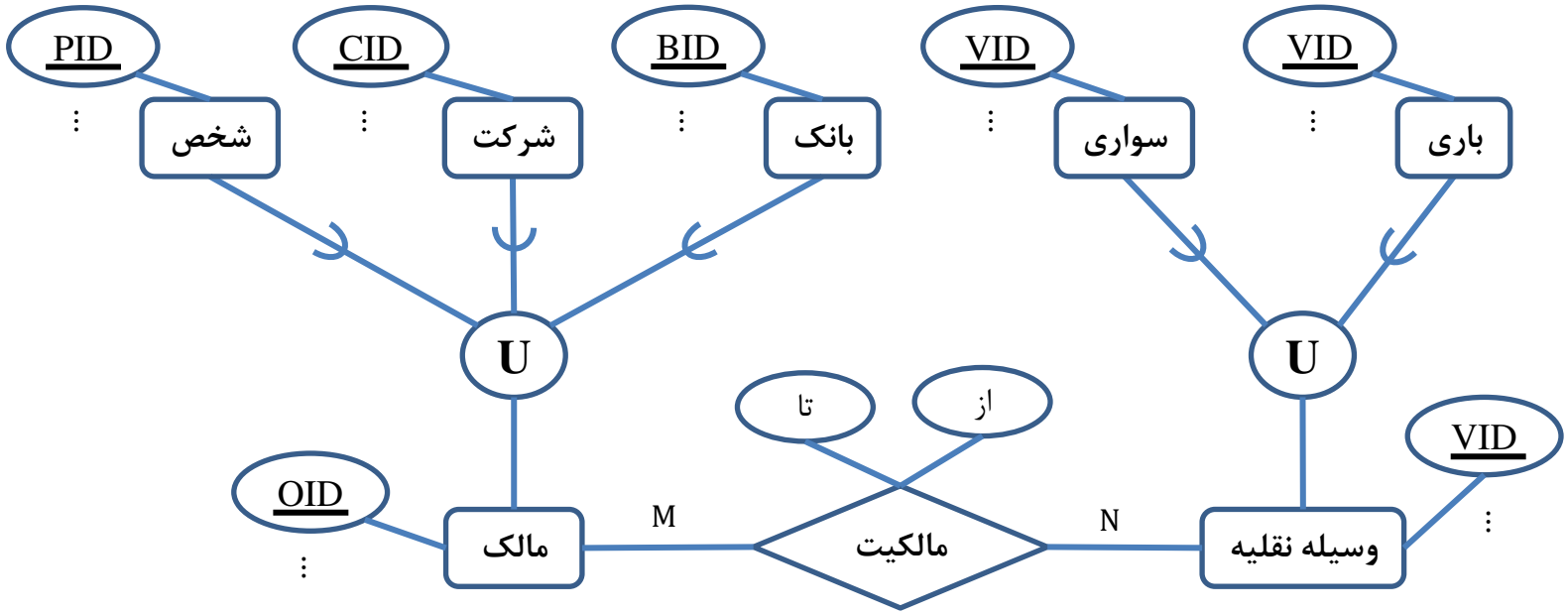
آیا ممکن است برای زیرنوع اصلاً رابطه طراحی نکنیم؟

حالت ۱۱: طراحی زیرنوع اجتماع (U-Type)

حالت ۱۱

- نوع موجودیت E، زیرنوع U-Type (دسته یا Category) n زیرنوع است.

n+1 رابطه طراحی می‌کنیم.



حالت ۱۱: طراحی زیرنوع اجتماع (ادامه)

- n+1 رابطه
- اگر شناسه زبرنوع‌ها از دامنه‌های متفاوت باشد، رابطه نمایشگر زیرنوع، FK می‌دهد به رابطه‌های نمایشگر زبرنوع‌ها، خارج از کلید.
- اگر شناسه زبرنوع‌ها از یک دامنه باشد، کلید رابطه نمایشگر زیرنوع، همان کلید رابطه‌های نمایشگر زبرنوع‌ها است.

PERS (PID, …., OID)

COMP (CID, …., OID)

BANK (BID, …., OID)

OWNER (OID,….) ← چون دامنه کلیدهای زبرنوع‌ها یکسان نیست، خودمان کلید ساختگی می‌گذاریم.

VEHIC (VID, ….)

OWNS (OID, VID, F, T, ….)

SAVARY (VID, N, ….)

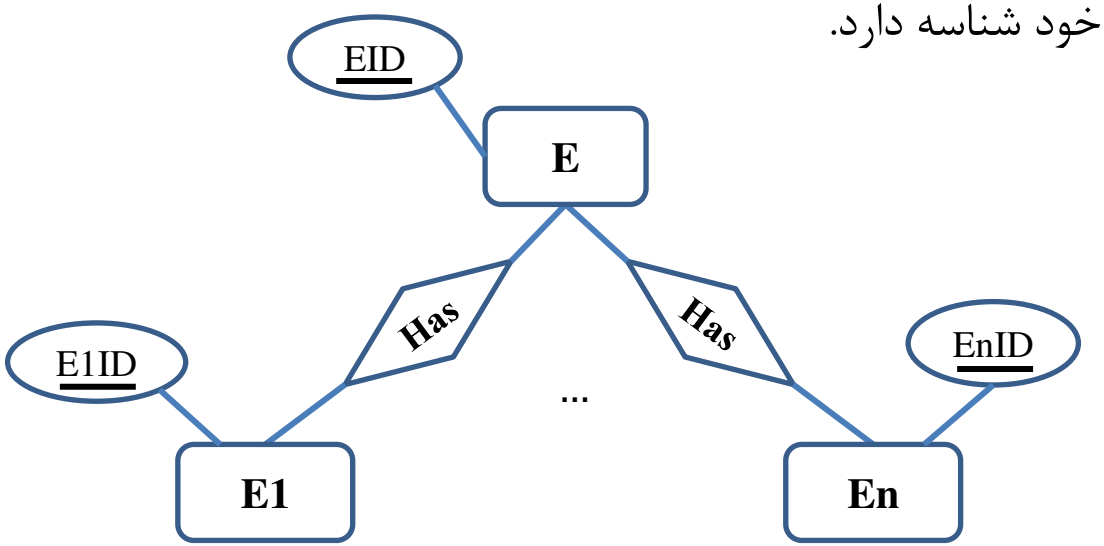
BARY (VID, T, ….)

آیا طرز طراحی دیگری وجود دارد.

حالت ۱۲: طراحی ارتباط IS-A-PART-OF

حالت ۱۲

- وجود ارتباط IS-A-PART-OF
- اگر نوع موجودیت کل، n نوع موجودیت جزء داشته باشد، تعداد n+1 رابطه طراحی می‌کنیم.
- توجه داریم که نوع موجودیت جزء از خود شناسه دارد.



E (EID, ….)

E1 (E1ID, EID, ….)

….

En (EnID, EID, ….)

آیا طرز طراحی دیگری وجود دارد؟ در چه شرایطی؟

حالت ۱۳: طراحی تکنیک Aggregation

حالت ۱۳

- استفاده از تکنیک Aggregation در مدلسازی
- ابتدا نوع موجودیت انتزاعی (بخش درون مستطیل خط‌چین) را طراحی می‌کنیم (با توجه به درجه و چندی ارتباط). سپس بخش بیرون آن را (باز هم با توجه به چندی ارتباط و درجه آن).

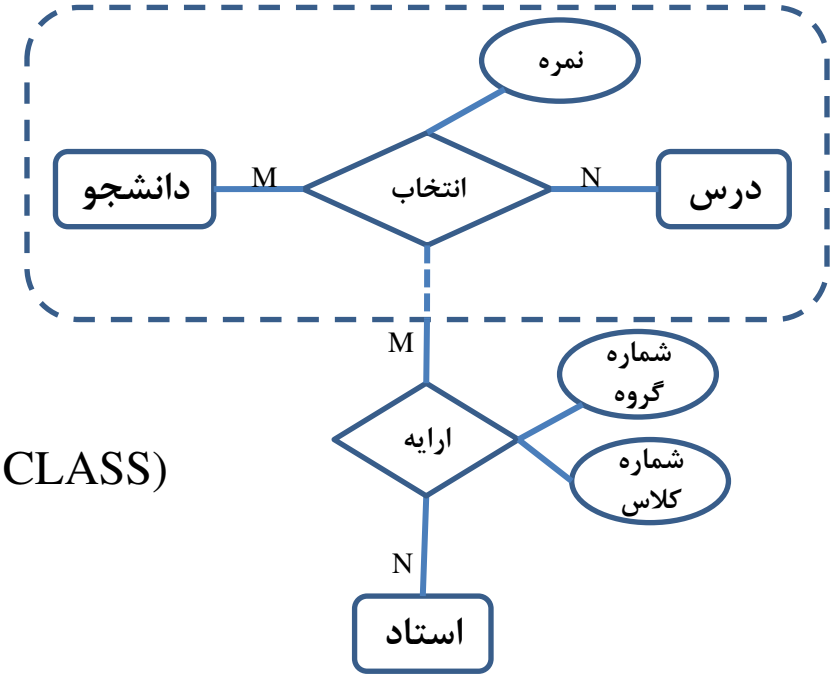
STUD (STID, ….)

COUR (COID, ….)

SCR (STID, COID, GR)

PROF (PRID, ….)

OFFERING (STID, COID, PROFID, GR#, CLASS)



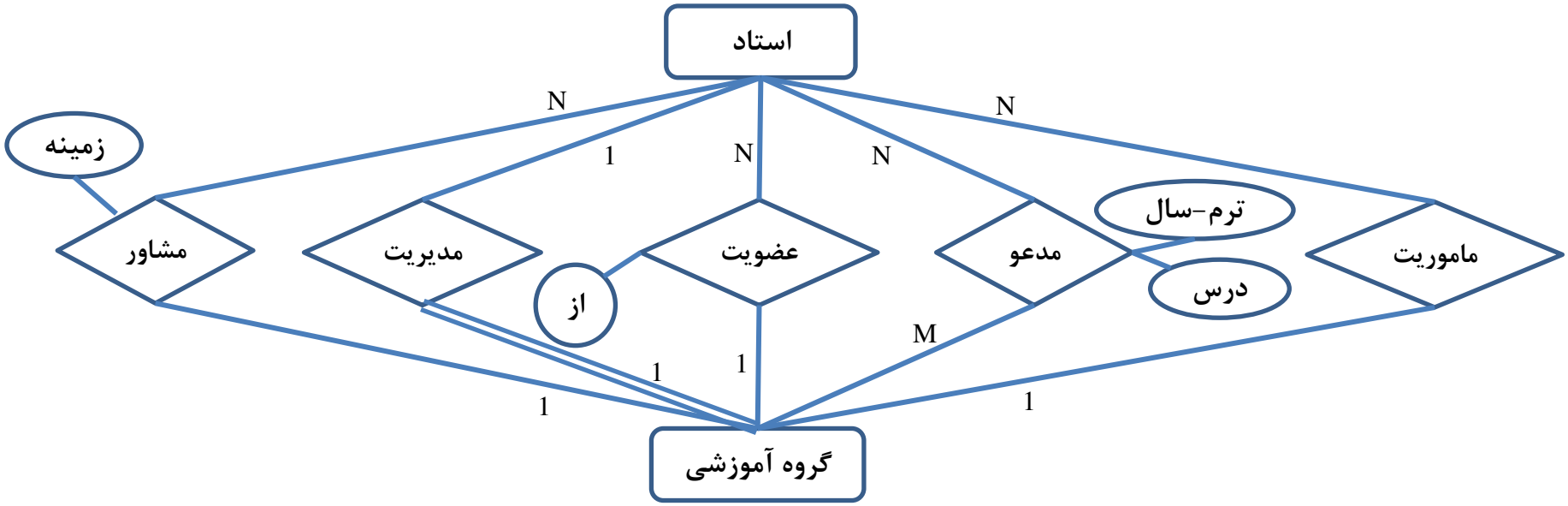
حالت ۱۳: طراحی تکنیک Aggregation (ادامه)

- این تکنیک چگونه کارایی سیستم را افزایش می‌دهد (نسبت به طراحی با یک ارتباط سه-تایی)؟
- اگر مراجعه به ارتباط «انتخاب» بالا باشد و فرکانس ارجاع به ارتباط «ارائه» پایین باشد، سیستم با این طراحی کاراتر عمل می‌کند.

حالت ۱۴: طراحی با وجود چند ارتباط

حالت ۱۴

- در صورتیکه چند ارتباط مثلاً بین دو نوع موجودیت برقرار باشد.
- هر ارتباط را با توجه به وضع آن از نظر درجه و چندی ارتباط طراحی می‌کنیم. اما برای کاهش احتمال اشتباه در طراحی توصیه می‌شود اول ارتباطهای M:N، سپس 1:N و در آخر 1:1 را طراحی نماییم.



حالت ۱۴: طراحی با وجود چند ارتباط (ادامه)

DEPT (DEID, …., DPHONE, PRID)

ماموریت موضوع عضویت از مشاور زمینه

PROF (PRID, …., PRRANK, MDEID, SUB, MEMDEID, FROM, CDEID, INT)

سه کلید خارجی از یک دامنه

INVITED (DEID, PRID, YR, TR)

- همین سیستم حداکثر با هفت رابطه نیز قابل طراحی است.

طراحی RDB– روش سنتز یا نرمال‌تر سازی رابطه‌ها

- **ایده اصلی:** یک رابطه، هر چند نرمال (با تعریفی که قبلاً دیدیم) ممکن است آنومالی (مشکل) داشته باشد در عملیات ذخیره‌سازی (در درج، حذف یا بهنگام‌سازی).
  - **آنومالی در درج:** عدم امکان درج یک فقره اطلاع که منطقاً باید قابل درج باشد.
  - **آنومالی در حذف:** حذف یک اطلاع ناخواسته در پی حذف اطلاع خواسته.
  - **آنومالی در بهنگام‌سازی:** بروز فزون‌کاری.
- پس باید رابطه را نرمال‌تر کرد.

فرم‌های نرمال

- نرمال بودن رابطه (نرمالیتی)، فرم‌ها (صورت‌ها/ سطوح/ درجات) [NF: Normal Forms] مختلفی دارد.
- **فرم‌های نرمال:**
  - 1NF
  - 2NF
  - 3NF
  - BCNF (Boyce-Codd Normal Form)
  - 4NF
  - 5NF یا PJNF (Projection Join Normal Form)
  - 6NF
  - DKNF (Domain Key Normal Form)

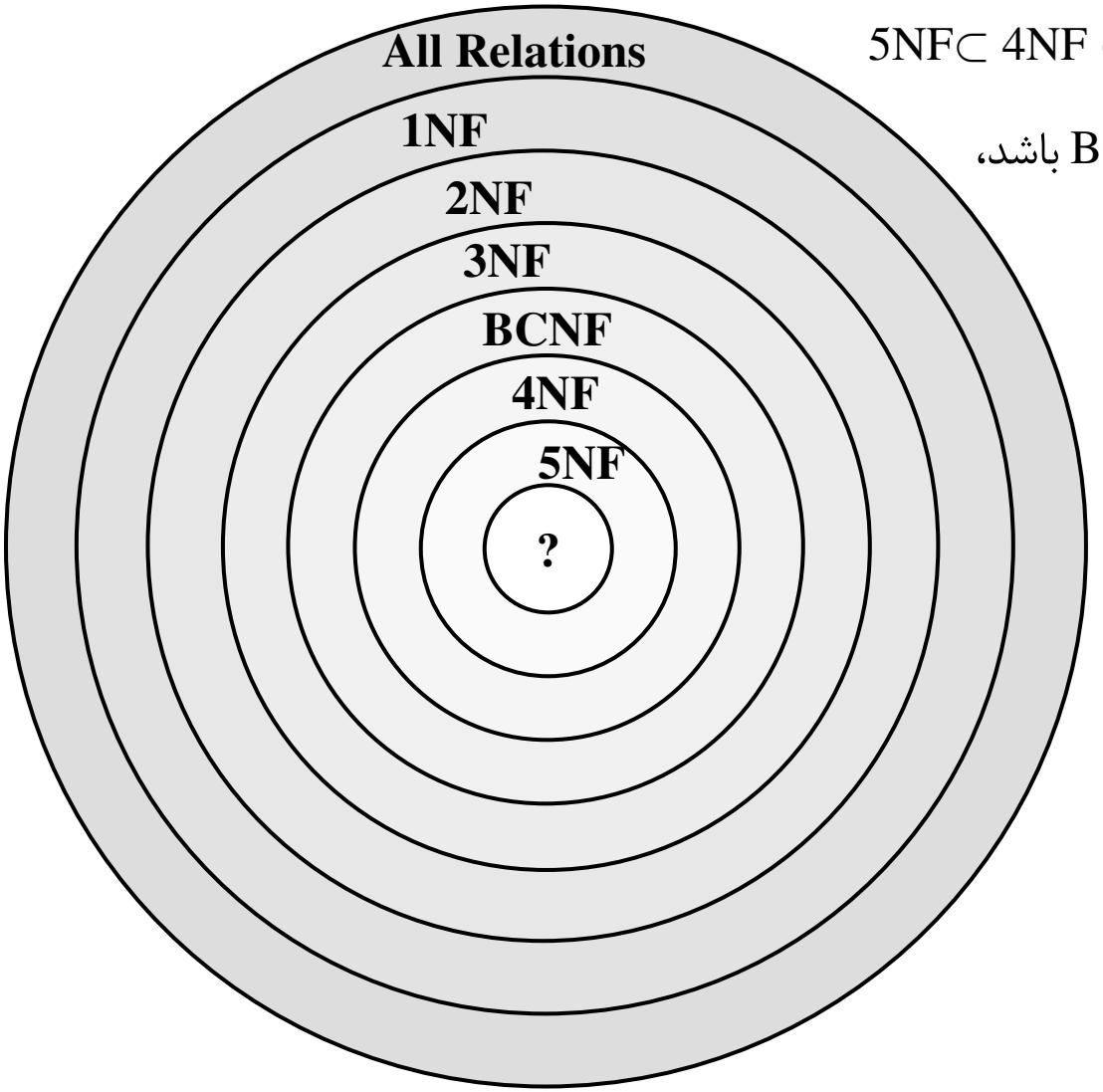
فرم‌های کلاسیک کادی (Codd)

سطوح در تئوری؛ چندان کاربرد عملی ندارند.

رابطه نرمال‌تر / آنومالی کمتر

رابطه بین فرم‌های نرمال

- 5NF⊂ 4NF ⊂ BCNF ⊂ 3NF ⊂2NF ⊂ 1NF
- یعنی به طور مثال، رابطه‌ای که BCNF باشد، 3NF هم هست.



تئوری وابستگی

- برای بررسی فرم‌های نرمال، نیاز به مفاهیمی داریم از تئوری وابستگی (Dependency Theory).
- مفاهیمی از تئوری وابستگی:
  - وابستگی تابعی (Functional Dependency)
  - وابستگی تابعی کامل [تام] (Fully Functional Dependency)
  - وابستگی تابعی با واسطه (Transitive Functional Dependency)

وابستگی تابعی (FD): صفت R.B به صفت R.A وابستگی تابعی دارد اگر و فقط اگر به ازای یک مقدار از A یک مقدار از B متناظر باشد. به عبارت دیگر اگر t_1 و t_2 دو تاپل از R باشند، در این صورت:

$$\text{IF} \quad t_1.A = t_2.A \quad \text{THEN} \quad t_1.B = t_2.B$$

با فرض اینکه کل تاپلهای رابطه به صورت زیر باشد، آیا داریم:

R (A,	B,	C)
a_1,	b_1,	c_1,
a_1	b_1	c_2
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_2	c_3

$a_1 \rightarrow b_1$

$a_1 \rightarrow c_1$, $a_1 \rightarrow c_2$

$A \rightarrow B$؟ بله

$A \rightarrow C$؟ خیر

$B \rightarrow A$؟ خیر

$B \rightarrow C$؟ خیر

وابستگی تابعی (ادامه)

- **نکات:**

(۱) صفات طرفین FD می‌توانند ساده یا مرکب باشند.

(۲) اگر $A \rightarrow B$، لزوماً نداریم: $B \rightarrow A$.

(۳) اگر $B \subseteq A$، به $A \rightarrow B$، FD نامهم یا بدیهی (Trivial) گوییم.

(۴) اگر K در رابطه R، SK یا CK باشد و $G \subseteq H_R$ آنگاه داریم: $K \rightarrow G$.

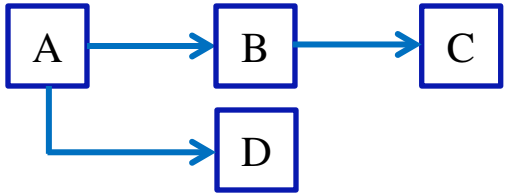
وابستگی تابعی (ادامه)

(۵) نمایش FDهای رابطه R به روشهای مختلف:

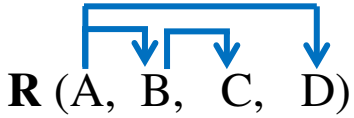
- به صورت یک مجموعه:

$$\mathbf{F}=\{A\rightarrow B,\ B\rightarrow C,\ A\rightarrow D\}$$

- با نمودار FDها:



- روی خود عنوان رابطه با استفاده از فلش‌هایی:



وابستگی تابعی (ادامه)

(۶) تفسیر FD: هر FD نمایشگر یک قاعده معنایی از محیط است: نوعی قاعده جامعیتی (که باید به نحوی به سیستم داده شود. خواهیم دید که در بحث طراحی، از طریق طراحی خوب به سیستم می‌دهیم).

- **تمرین:** در رابطه R(X, Y, Z)، یک اِظهار بنویسید که قاعده معنایی X→Y را پیاده‌سازی نماید.

(به طور مثال می‌توان از EXISTS استفاده کرد)

```
CREATE ASSERTION  XTOYFD
    CHECK ( NOT EXISTS (SELECT X FROM R GROUP BY X HAVING MAX(Y)!=MIN(Y)))
```

حساب رابطه‌ای: **CONSTRAINT** XTOYFD **FORALL** R1 (**FORALL** R2 **IF** R1.X=R2.X **THEN** R1.Y=R2.Y)

مثال

STID→STJ: یک دانشجو فقط می‌تواند در یک رشته تحصیل کند.

STJ→STD: یک رشته فقط در یک دانشکده ارائه می‌شود.

STID→STD: یک دانشجو فقط در یک دانشکده تحصیل می‌کند.

قواعد استنتاج آرمسترانگ

1- if $B \subseteq A$ then $A \rightarrow B$ $\Rightarrow$ $A \rightarrow A$ (قاعده انعکاسی)

2- if $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$ (قاعده تعدی یا تراگذاری)

3- if $A \rightarrow B$ then $(A,C) \rightarrow (B,C)$ (قاعده افزایش)

4- if $A \rightarrow (B,C)$ then $A \rightarrow B$ and $A \rightarrow C$ (قاعده تجزیه)

5- if $A \rightarrow B$ and $C \rightarrow D$ then $(A,C) \rightarrow (B,D)$ (قاعده ترکیب)

6- if $A \rightarrow B$ and $A \rightarrow C$ then $A \rightarrow (B,C)$ (قاعده اجتماع)

7- if $A \rightarrow B$ and $(B,C) \rightarrow D$ then $(A,C) \rightarrow D$ (قاعده شبه‌تعدی)

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

- سه قاعده اول درست و کامل هستند، بدین معنا که با داشتن یک مجموعه از وابستگی‌های تابعی F، تمام وابستگی‌های تابعی منطقاً قابل استنتاج از F، با همین سه قاعده به دست می‌آیند و هیچ وابستگی تابعی دیگر (که از F قابل استنتاج نباشد) نیز به دست نمی‌آید.
- **توجه:** سه قاعده اول به آسانی قابل اثبات هستند و قواعد دیگر از روی همانها اثبات می‌شوند.

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

- **تمرین:** قاعده ۲ را اثبات کنید (با استفاده از برهان خلف).
- اثبات: فرض خلف: گیریم که $A \not\to C$. در این صورت در رابطه R در حداقل دو تاپل، به ازای یک مقدار A، دو مقدار متمایز از C داریم.
- اما به ازای دومقدار متمایز C، مقدار B ممکن است دو مقدار متمایز با یک مقدار باشد.

R (A,	B,	C)
a_1	…	c_1
.	.	.
a_1	…	c_2

R (A,	B,	C)
a_1	b_1	c_1
.	.	.
a_1	b_2	c_2

حالت اول

R (A,	B,	C)
a_1	b_1	c_1
.	.	.
a_1	b_1	c_2

حالت دوم

- در حالت اول، فرض $A \to B$ و در حالت دوم، فرض $B \to C$ نقض می‌شود. پس فرض خلف باطل است و حکم برقرار است.

کاربردهای قواعد آرمسترانگ

۱- محاسبه بستار صفت A: A^+

مجموعه تمام صفاتی که با A، وابستگی تابعی دارند.

نکته: اگر $A^+=H_R \Leftarrow$ A سوپرکلید (الگوریتم تشخیص سوپرکلید و نه کلید کاندید)

۲- محاسبه بستار مجموعه وابستگی‌های تابعی یک رابطه: F^+

مجموعه تمام FDهایی که از F منطقاً استنتاج می‌شوند:

$$F=\{A\rightarrow B,\ B\rightarrow C\} \quad \Rightarrow \quad F^+=\{A\rightarrow B,\ B\rightarrow C,\ A\rightarrow C,\ (A,C)\rightarrow(B,C),\ \ldots\}$$

- **کاربردهای مهم F^+:**

۱- تشخیص معادل بودن دو مجموعه از FDهای رابطه‌ای R: به طور نمونه F و G

- شرط معادل بودن: $F^+=G^+$

هر FD که از F به دست آید، از G هم به دست می‌آید.

۲- تشخیص FD افزونه

- ضابطه تشخیص: وابستگی تابعی $f \in F$ را افزونه گوییم، هرگاه: $(F-f)^+=F^+$
- یعنی بود و نبود f در محاسبه F^+ تاثیری نداشته باشد.

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

۳- محاسبه مجموعه کاهش‌ناپذیر FDهای یک رابطه

سه شرط دارد:

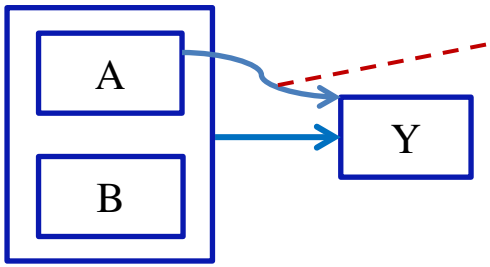
۱- هیچ FD در آن افزونه نباشد.

۲- سمت راست هر FD، صفت ساده باشد.

۳- سمت چپ هر FD، خود کاهش‌ناپذیر باشد: در وابستگی تابعی $X \rightarrow Y$، X را کاهش‌ناپذیر (و وابستگی $X \rightarrow Y$ را **کامل**) گوییم، هرگاه Y با هیچ زیرمجموعه از X (غیر از خود X)، FD نداشته باشد.

در غیر اینصورت X را کاهش‌پذیر گوییم و وابستگی $X \rightarrow Y$ را **ناکامل** گوییم.

اگر وجود داشته باشد، آنگاه X کاهش‌پذیر و $X \rightarrow Y$ یک FD ناکامل است.



$$\begin{cases} \overbrace{(A, B)}^{X} \rightarrow Y \\ A \rightarrow Y \end{cases} \Rightarrow \text{FD ناکامل}$$

- **تمرین:** اگر یک FD کامل به صورت $A \rightarrow Y$ داشته باشیم، آنگاه FD ناکامل $(A,B) \rightarrow Y$ از آن قابل استنتاج است.
- اثبات: با استفاده از قاعده افزایش از $A \rightarrow Y$ نتیجه می‌گیریم $(A,B) \rightarrow (Y,B)$

با استفاده از قاعده تجزیه داریم: $(A,B) \rightarrow B$ که یک FD بدیهی است و $(A,B) \rightarrow Y$ که همان حکم است.

کجای؟ مجموعه کاهش‌ناپذیر چه کاربردی دارد؟

تعریف **وابستگی تابعی با واسطه (TFD):** اگر $A \rightarrow B$، $B \rightarrow C$ و $B \not\rightarrow A$، می‌گوییم C با A، FD با واسطه از طریق B دارد.

اگر $B \rightarrow A$ هم برقرار باشد، آنگاه آن FD با واسطه، بدیهی (نامهم) است.

فرم‌های نرمال کلاسیک کادی

- **توجه:** در سه فرم کلاسیک کادی، فقط با مفهوم کلید اصلی (PK) کار می‌کنیم و نه هر CK.

تعریف **1NF**: رابطه R در 1NF است اگر و فقط اگر تمام صفات آن تک‌مقداری باشد.

- این تعریف می‌گوید هر رابطه نرمال در 1NF است.

تعریف **2NF**: رابطه R در 2NF است اگر و فقط اگر در 1NF باشد و هر صفت ناکلید (که خود PK یا CK نباشد و جزء PK یا CK هم نباشد) در آن، با کلید اصلی رابطه، FD کامل داشته باشد.

- به بیان دیگر در این رابطه FD ناکامل با کلید اصلی نداشته باشیم.

- الگوریتم تبدیل 1NF به 2NF: حذف FDهای ناکامل از طریق تجزیه عمودی رابطه به طور مناسب.

تعریف **3NF**: رابطه R در 3NF است اگر و فقط اگر در 2NF باشد و هر صفت ناکلید با کلید اصلی رابطه، فقط FD بی‌واسطه داشته باشد (FD باواسطه نداشته باشد).

- الگوریتم تبدیل 2NF به 3NF: حذف FDهای با واسطه.

فرم‌های نرمال کلاسیک کادی (ادامه)

مثالی قید می‌کنیم و در آن تا 3NF پیش می‌رویم.

- در حالت کلی، تمام صفات دانشجو، درس و انتخاب در یک رابطه می‌توانند باشند.
- قواعد محیط:

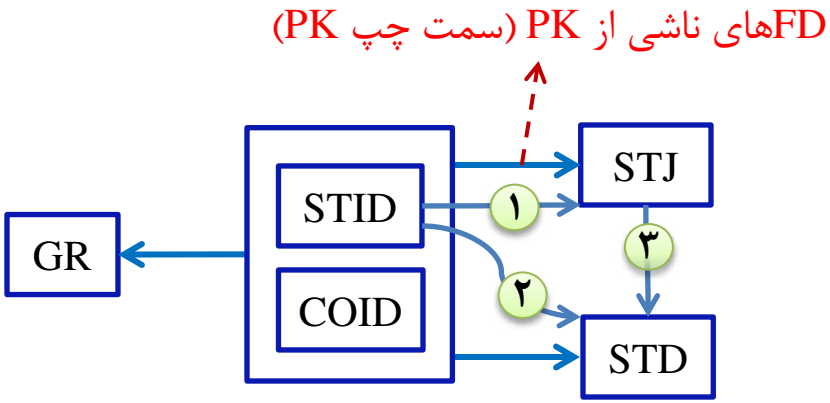
۱- یک دانشجو در یک رشته تحصیل می‌کند.

۲- یک دانشجو در یک دانشکده تحصیل می‌کند.

۳- یک رشته در یک دانشکده ارائه می‌شود.

R (STID, COID, STJ, STD, GR)

STID	COID	STJ	STD	GR
777	CO1	Phys	D11	19
777	CO2	Phys	D11	16
777	CO3	Phys	D11	11
888	CO1	Math	D12	16
888	CO2	Math	D12	18
444	CO1	Math	D12	13
555	CO1	Phys	D11	14
555	CO2	Phys	D11	12



فرم‌های نرمال کلاسیک کادی (ادامه)

- رابطه **R** در 1NF است (چون همه صفات تک مقداری هستند) ولی آنومالی دارد و باید نرمال‌تر شود.
- **آنومالی‌های رابطه R:**

۱- **در درج:**

درج کن این فقره اطلاع درمورد یک دانشجو را: ‘666’, ‘chem’, ‘D16’〉

درج ناممکن: تا ندانیم حداقل یک درسی که گرفته شده چیست.

۲- **در حذف:**

فرض می‌کنیم ‘444’ در این لحظه فقط همین تک درس را داشته باشد.

حذف کن فقط این اطلاع را: ‘444’, ‘CO1’, 13〉

حذف انجام می‌شود اما اطلاع ناخواسته هم حذف می‌شود.

۳- **در بهنگام‌سازی:**

تغییر رشته تحصیلی دانشجو با شماره 777 به Chem.

برای انجام آن فزونکاری داریم؛ بهنگام‌سازی منتشرشونده (Propagating Update).

فرم‌های نرمال کلاسیک کادی (ادامه)

- **دلیل آنومالی‌های رابطه R:**
  - از دیدگاه عملی: پدیده اختلاط اطلاعات، یعنی اطلاعات در مورد خود موجودیت دانشجو با اطلاعات در مورد انتخاب درس مخلوط شده است.
  - از دیدگاه تئوری: وجود FDهای ناکامل

$$\begin{cases} (STID, COID) \rightarrow STD \\ STID \rightarrow STD \end{cases} \qquad \begin{cases} (STID, COID) \rightarrow STJ \\ STID \rightarrow STJ \end{cases}$$

  - این FDهای ناکامل باید از بین بروند. برای این منظور رابطه **R** را باید چنان تجزیه عمودی کنیم که در رابطه‌های حاصل، FD ناکامل نباشد.
  - برای این کار از عملگر پرتو استفاده می‌کنیم. پرتوی که منجر به یک تجزیه خوب شود.

فرم‌های نرمال کلاسیک کادی (ادامه)

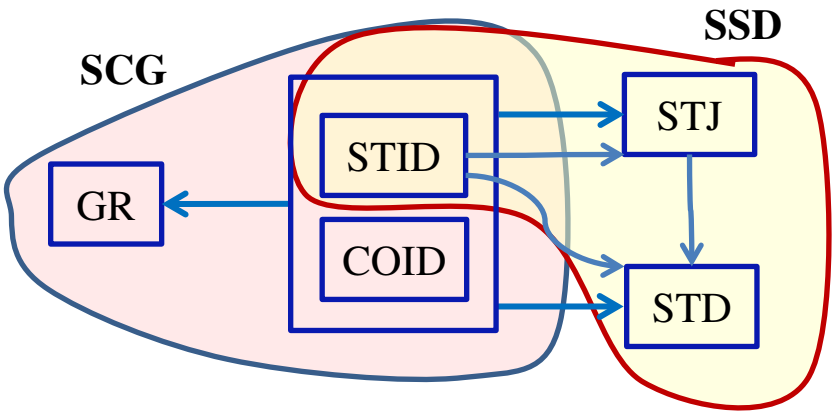
$\Pi_{\langle STID,COID,GR\rangle}(R)$ و $\Pi_{\langle STID,STJ,STD\rangle}(R)$

SCG (STID, COID, GR)

STID	COID	GR
777	CO1	19
777	CO2	16
777	CO3	11
888	CO1	16
888	CO2	18
444	CO1	13
555	CO1	14
555	CO2	12

SSD (STID, STJ, STD)

STID	STJ	STD
777	Phys	D11
888	Math	D12
444	Math	D12
555	Phys	D11



فرم‌های نرمال کلاسیک کادی (ادامه)

- رابطه‌های جدید آنومالی‌های R را ندارند:

۱- درج کن: ⟨'666', 'chem', 'D16'⟩

بدون مشکل در SSD درج می‌شود.

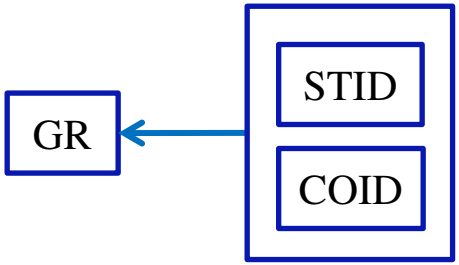
۲- حذف کن: ⟨'444', 'CO1', 13⟩

بدون مشکل از SCG حذف می‌شود.

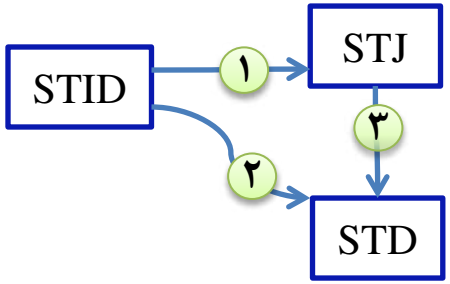
۳- بهنگام‌سازی کن: تغییر رشته دانشجوی 777 را به Chem

بدون مشکل در SSD بروز می‌شود.

SCG



SSD



فرم‌های نرمال کلاسیک کادی (ادامه)

- در طراحی جدید، FDهای ناکامل از بین رفتند. بنابراین SSD و SCG، 2NF هستند.
- **تاکید:** رابطه R، 2NF است هرگاه اولاً در 1NF باشد و ثانیاً هر صفت ناکلید با کلید اصلی، FD کامل داشته باشد (رابطه، FD ناکامل نداشته باشد).
- **تمرین:** بررسی شود که آیا در این تجزیه همه FDها محفوظ می‌مانند؟
- **نکته:** باید توجه کنیم که در تجزیه، FDای از دست نرود، چون هر FD یک قاعده جامعیت در محیط است.
- توجه داشته باشید که در این تجزیه هیچ اطلاعی از دست نمی‌رود. یعنی اگر کاربر رابطه اصلی را به هر دلیلی بخواهد با پیوند دو رابطه جدید به دست می‌آید.

$$R = SCG \bowtie SSD$$

فرم‌های نرمال کلاسیک کادی (ادامه)

- در حالت کلی اگر R_1، R_2، .... و R_n پرتوهای دلخواه از R باشند، به شرط عدم وجود هیچمقدار داریم (ممکن است تاپل‌های افزونه بروز کند):

$$R \subseteq R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$$

- **تجزیه بی‌حذف:** شرطش این است که در صفات پیوند هیچمقدار (Null Value) نداشته باشیم.
  - اگر در صفات پیوند هیچمقدار داشته باشیم، چه پیش می‌آید؟

$$T(\underline{A},\ B,\ C,\ D,\ E) \Rightarrow \quad T_1(A, B) \quad T_2(B,\ C,\ D,\ E)$$

تاپل‌هایی در پیوند از دست می‌روند. به این تاپل‌ها، تاپل‌های آونگان [معلق] (Dangling) گوییم.

- در مباحث نرمالترسازی معمولا فرض بر این است که **صفت (صفات) پیوند هیچمقدار ندارند**.

فرم‌های نرمال کلاسیک کادی (ادامه)

- آیا رابطه‌های جدید (SSD و SCG) آنومالی ندارند؟
- **آنومالی‌های SSD:**

۱- در درج:

اطلاع: «رشته IT در دانشکده D20 ارائه می‌شود.» به دلیل FD شماره ۳، این اطلاع منطقاً باید قابل درج باشد، اما درج ناممکن است. چون کلید ندارد، باید حداقل یک دانشجوی این رشته را بشناسیم.

۲- در حذف:

حذف کن ⟨'666', 'Chem'⟩ و با فرض اینکه تنها یک دانشجو در رشته Chem ثبت شده است.

حذف انجام می‌شود ولی اطلاع «رشته شیمی در D16 ارائه می‌شود»، ناخواسته حذف می‌شود.

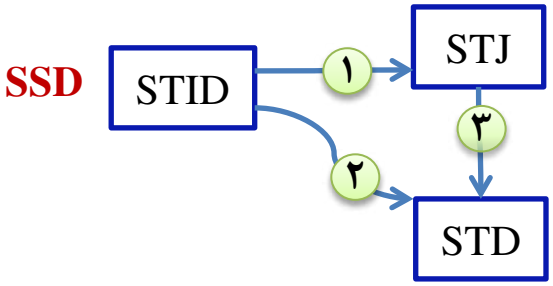
۳- در بهنگام‌سازی:

«شماره دانشکده رشته فیزیک را عوض کنید». به تعداد تمام دانشجویان این رشته باید بهنگام‌سازی شود.

SSD باید نرمال‌تر شود.

دلیل آنومالی‌های SSD:

- دلیل آنومالی‌های SSD، وجود FD با واسطه بین صفت ناکلید با کلید اصلی است (به دلیل FD شماره ۳).
- این FD باید از بین برود.



- فرض کنید SSD را به صورت زیر تجزیه کنیم:

SD (STJ, STD) و **SJ (STID, STJ)**

STJ	STD
Phys	D11
Math	D12

STID	STJ
777	Phys
888	Math
444	Math
555	Phys

- افزونگی کم شد!
- تمرین: بررسی شود که رابطه‌های جدید آنومالی‌های SSD را ندارند.

فرم‌های نرمال کلاسیک کادی (ادامه)

- این رابطه‌ها در 3NF هستند.
  - اولاً در 2NF هستند.
  - ثانیاً FD با واسطه نداریم.

SJ: STID → STJ

SD: STJ → STD

- **تمرین:** بررسی شود که در این تجزیه هیچ اطلاعی از دست نمی‌رود و FDها هم حفظ می‌شوند.
- **تاکید:** رابطه R در 3NF است اگر و فقط اگر اولاً در 2NF باشد و ثانیاً هر صفت ناکلید با کلید اصلی FD بی‌واسطه داشته باشد (تمام FDها مستقیماً ناشی از PK باشد).
- **نتیجه:** FDهای ناکامل و باواسطه مزاحم هستند و باید از بین بروند.
- در عمل رابطه‌ها باید حداقل تا 3NF نرمال شوند و خواهیم دید حتی‌الامکان در BCNF یا بیشتر باشند.
- در رابطه 3NF داریم که «یک بوده (واقعیت) : یک رابطه» و یا «یک شیئ : یک رابطه».

- **تجزیه خوب (Nonloss/Lossness Decomposition)**

۱- بی‌حشو: در پیوند پرتوها، تاپل حشو [افزونه] بروز نکند.

۲- حافظ FDها: هیچ FDای در اثر تجزیه از دست نرود و همه FDهای رابطه اصلی حفظ شوند.

۳- بی‌حذف: در پیوند پرتوها هیچ تاپلی حذف نشود (صفت یا صفات پیوند هیچمقدار نباشند).

۴- حافظ صفات: $\cup_{i\in\{1,\ldots,n\}} H_{R_i} = H_R$

پیش‌فرض یا بدیهی

- در بیشتر متون کلاسیک، بحث تجزیه خوب، تحت عنوان **تجزیه بی‌کاست یا بی‌گمشدگی** (Nonloss/Lossless Decomposition) مطرح شده است، که منظور همان بی‌حشو و حافظ وابستگی‌های تابعی بودن است (و دو ویژگی دیگر تجزیه خوب را پیش‌فرض تجزیه خوب بدانیم).

- در واقع تاپلهای افزونه باعث از دست رفتن بخشی از اطلاعات می‌شوند.

- **قضیه ریسانِن (Rissanen):**
  - رابطه R به دو پرتوش (R_1 و R_2) **تجزیه خوب** می‌شود، اگر R_1 و R_2 از یکدیگر مستقل باشند.
  - R_1 و R_2 **مستقل** از یکدیگرند اگر و فقط اگر:
    - صفت مشترک، حداقل در یکی از آنها CK باشد ⇐ بی‌حشو بودن
    - تمام FDهای رابطه اصلی یا در مجموعه FDهای R_1 و R_2 وجود داشته باشند یا از آنها منطقاً استنتاج شوند ⇐ حافظ FDها
- **نکته:** بر اساس ضوابط ریسانن، اگر در رابطه R(A, B, C)، وابستگی‌های A→B، B→C و A→C برقرار باشد، در اینصورت تجزیه خوب چنین است: $R_1(\underline{A}, B)$ و $R_2(\underline{B}, C)$.
  - در اینجا B در رابطه دوم کلید کاندید است، چون همه صفات به آن وابستگی تابعی دارند و کاهش‌پذیر هم نیست.

- مثال: رابطه SSD را در نظر می‌گیریم. این رابطه به سه شکل به پرتوهای دوگانی قابل تجزیه است.

I SS (STID, STJ) SD (STJ, STD)

II SS (STID, STJ) SD (STID, STD)

III SS(STID, STD) SJ (STJ, STD)

- تجزیه I خوب است، چون هر دو شرط ریسانِن را دارد.

$$\left.\begin{array}{l} STID \rightarrow STJ \\ STJ \rightarrow STD \end{array}\right\} \Rightarrow STID \rightarrow STD$$

- تجزیه II خوب نیست، چون FD از دست می‌دهد.
- تجزیه III خوب نیست، چون FD از دست می‌دهد.

فرم نرمال BCNF

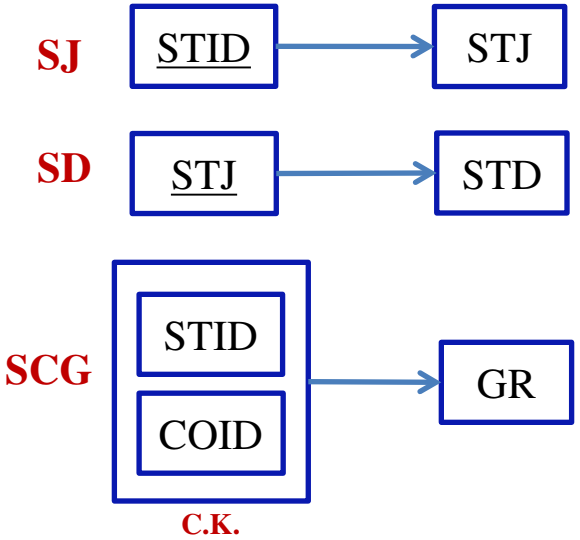
- **اصطلاح:** در وابستگی تابعی A→B (A Determines B) به A دترمینان گویند.

BCNF: رابطه R در BCNF است اگر و فقط اگر در آن دترمینان هر FD مهم و کاهش‌ناپذیر، CK باشد.

- در 3NF، تنها باید دترمینان رابطه PK باشد.
- چون رابطه می‌تواند بیش از یک CK داشته باشد، BCNF از 3NF قوی‌تر است.

رابطه‌های زیر در BCNF هستند.

SCGJD
- **SCG(SID, COID, GR)**
- **SJ (STID, STJ)**
- **SD (STJ, STD)**



فرم نرمال BCNF (ادامه)

- BCNF از 3NF قوی‌تر است. ⇐ رابطه می‌تواند در 3NF باشد، اما در BCNF نباشد.
- **حالت I:** رابطه R فقط یک CK داشته باشد. ⇐ اگر R در 3NF باشد، در BCNF هم هست (مثال دیده شده).
- **حالت II:** رابطه R بیش از یک CK داشته باشد.
  - **II-1)** CKها مجزا باشند (صفت مشترک نداشته باشند). ⇐ اگر R در 3NF باشد، در BCNF هم هست.
  - **II-2)** CKها هم‌پوشا باشند. ⇐ اگر R در 3NF باشد، لزوماً در BCNF نیست.

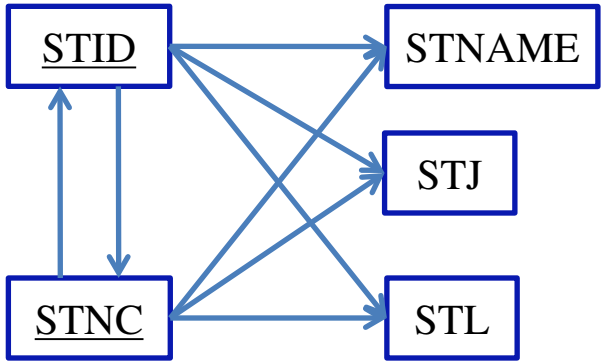
مثال

برای حالت II-1

دو دترمینان، هر دو هم CK هستند.

ST (STID, STNAME, STNC, STJ, STL, …)

C.K. C.K.



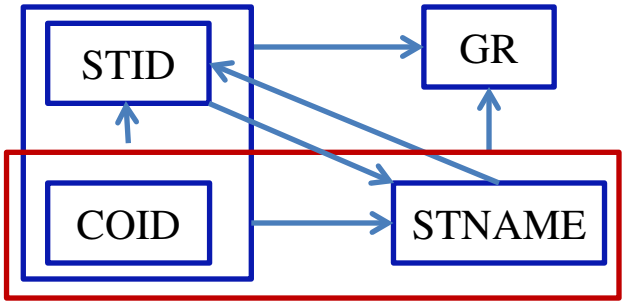
مثال

برای حالت II-2

(فرض: هیچ دو دانشجویی نام یکسان ندارند.)

SCNG (STID, COID, STNAME, GR)

C.K. C.K.



فرم نرمال BCNF (ادامه)

- کافی است یک دترمینان در رابطه پیدا کنیم که CK نباشد. ⇐ رابطه BCNF نیست.
- پس در کدام فرم نرمال است؟
  - 1NF هست. چون صفت‌ها تک‌مقداری هستند.
  - 2NF هست. چون FD ناکامل نداریم. ⇐ هر صفت ناکلید با کلید اصلی FD ناکامل نداشته باشد.

    ⇐در اینجا STNAME صفت غیرکلید نیست، پس FD ناکامل نیست.
  - 3NF هست. چون FD باواسطه با کلید اصلی نداریم.
- آیا این رابطه تجزیه می‌شود؟

SCG(STID, COID, GR)
C.K.

SSN (STID, STNAME)
C.K. C.K.

⇒ هر دو BCNF هستند.

- آیا طرز دیگر هم می‌شود تجزیه کرد؟ بله، به جای STID در SCG، STNAME بگذاریم.

فرم نرمال BCNF (ادامه)

- نشان دهید که این تجزیه خوب است؛ یعنی با پیوند پرتوها، رابطه اصلی به دست می‌آیدو هیچ FD از دست نمی‌رود.
- چه پدیده‌ای در اینجا دیده می‌شود؟ این رابطه اختلاط اطلاعات دارد! با این همه 3NF است.

SCNG (STID, COID, STNAME, GR)

C.K. C.K.

- **نکته:** صرف وجود اختلاط اطلاعات ایجاب می‌کند که رابطه در فرم نرمال ضعیفی باشد.

- **تمرین:** محیط دانشکده، قواعد معنایی:

۱- یک دانشجو یک درس را با یک استاد انتخاب می‌کند.

۲- یک استاد فقط یک درس تدریس می‌کند.

۳- یک درس توسط بیش از یک استاد ارائه می‌شود.

فرم نرمال BCNF (ادامه)

- فرض می‌کنیم طراح رابطه زیر را طراحی کرده است.

C.K.

SCNG (ST#, CO#, PR#)

C.K.

- این رابطه در کدام فرم نرمال است؟
  - ابتدا باید با استفاده از قواعد، CKها را مشخص کنیم. سپس نمودار FD را رسم کنیم.
- آیا این رابطه، تجزیه خوب دارد؟
  - **نکته:** اگر رابطه مثلاً 3NF باشد و تجزیه خوب نداشته باشد، نباید تجزیه کنیم تا رابطه‌های حاصل BCNF باشد.
- رابطه فوق در 3NF است و از نکته فوق این نتیجه مهم به دست می‌آید که این رابطه تجزیه خوب ندارد.

قضیه هیث (Heath): در رابطه $R(A, B, C)$، که در آن A، B و C سه مجموعه از صفات هستند، اگر $A \rightarrow B$ (در F^+ باشد)، آنگاه تجزیه R به دو پرتو $R_1(A, B)$ و $R_2(A, C)$، **تجزیه بی‌کاست** (Nonloss) است.

دقت شود که برقراری شرایط قضیه هیث، یک تجزیه بی‌کاست (و نه لزوما خوب که حافظ FD باشد) را تضمین می‌نماید اما برقراری شرایط قضیه ریسانن، یک تجزیه خوب را تضمین می‌نماید. واضح است که در قضیه ریسانن شرایط قضیه هیث نیز برقرار است. بیانی دیگر از قضیه هیث تحت عنوان تست NJB به صورت زیر است.

تست پیوند بی‌حشو برای تجزیه دودویی (NJB- Nonadditive Join Test for Binary Decompositions):

تجزیه دودویی $D=\{R_1, R_2\}$ از رابطه R خاصیت پیوند بی‌حشو دارد اگر و تنها اگر یکی از موارد زیر با توجه به مجموعه FDهای F برقرار باشد:

- وابستگی تابعی $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ در F^+ باشد یا
- وابستگی تابعی $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ در F^+ باشد.

فرم نرمال 4NF

4NF: رابطه R در4NF است اگر و فقط اگر در BCNF باشد و وابستگی چندمقداری (MVD) مهم در آن وجود نداشته باشد.

وابستگی چندمقداری (MVD): در رابطه R(A, B, C) (رابطه با سه صفت یا سه مجموعه صفت)، صفت B با صفت A، MVD دارد (A→→B) اگر و فقط اگر به ازای یک مقدار A، مجموعه‌ای از مقادیر B متناظر باشد.

[یعنی به ازای هر جفت مشخص از (A,C)، مجموعه مقادیر B فقط با تغییرات A تغییر کند.]

R (A, B, C)

$$\begin{array}{l} a_1 \left\{ \begin{array}{l} b_1 \\ b_2 \\ b_3 \end{array} \right\} c_1 \\ a_1 \left\{ \begin{array}{l} b_1 \\ b_2 \\ b_3 \end{array} \right\} c_2 \\ a_2 \left\{ \begin{array}{l} b_1 \\ b_7 \end{array} \right\} c_i \end{array} \quad \Rightarrow \quad \mathbf{A \rightarrow\rightarrow B}$$

- **نکات:**

۱- اگر $B \subseteq A$ باشد، به $A \rightarrow\rightarrow B$ می‌گوییم MVD بدیهی [نامهم]

اگر $A \cup B = H_R$ باشد، به $A \rightarrow\rightarrow B$ می‌گوییم MVD بدیهی [نامهم]

۲- MVD در رابطه‌های با سه صفت [ساده یا مرکب] همیشه جفت است.

If $A \rightarrow\rightarrow B$ **then** $A \rightarrow\rightarrow (H-\{A,B\})$ **یا** $A \rightarrow\rightarrow C$

برای اثبات این نکته کافی است به جای یک جفت مقدار از (A, C)، یک جفت (A, B) را بگیریم، آن مجموعه برای C تشکیل می‌شود.

۳- برای MVD هم قواعد آرمسترانگ وجود دارد که با قواعد مربوط به FDها متفاوت است.

فرم نرمال 4NF (ادامه)

مثال

استاد از دانشجو گزارش آزمایشگاه می‌گیرد.

رابطه غیرنرمال با صفت چندمقداری

NNPSR (PR#, ST#, RE#)

- در این محیط یک قاعده معنایی خاص وجود دارد: یک استاد از هر یک از دانشجویان یک گروه، هر یک از گزارش‌های یک مجموعه گزارش را می‌گیرد.
- اگر این قاعده معنایی نباشد، این مجموعه‌ها شکل نمی‌گیرد.

NNPSR (PR#, ST#, RE#)

$$\begin{array}{lll} PR_1 & \begin{bmatrix} 777 \\ 888 \\ 444 \end{bmatrix} & \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} \\ PR_2 & \begin{bmatrix} 777 \\ 666 \end{bmatrix} & \begin{bmatrix} R_3 \end{bmatrix} \\ \dots & \dots & \dots \end{array}$$

فرم نرمال 4NF (ادامه)

مثال

درس C توسط استاد T از روی کتاب B ارائه می‌شود.

- پدیده MVD بیان فرمال صفت چندمقداری است.

مثال

فرم نرمال شده این مثال، افزونگی زیادی دارد.

- رابطه تمام‌کلید است؛ یعنی هیچ یک به تنهایی و هیچ ترکیب دوتایی آن CK نیست.
- رابطه تمام‌کلید حداقل BCNF است.

زیرا یک دترمینان دارد که آن هم CK است.

رابطه غیرنرمال با صفت چندمقداری

NNCTX (C#, T#, B#)

$$c_1 \left\{ \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \right\} \left\{ \begin{matrix} b_1 \\ b_2 \end{matrix} \right\}$$

$$c_2 \left\{ \begin{matrix} t_4 \\ \\ t_2 \end{matrix} \right\} \left\{ \begin{matrix} b_3 \\ b_5 \\ b_7 \end{matrix} \right\}$$

CTX (C#, T#, B#)

c_1	t_1	b_1
c_1	t_2	b_1
c_1	t_3	b_1
c_1	t_1	b_2
c_1	t_2	b_2
c_1	t_3	b_2
c_2	t_4	b_3
c_2	t_2	b_3
c_2	t_4	b_5
c_2	t_2	b_5
c_2	t_4	b_7
c_2	t_2	b_7

فرم نرمال 4NF (ادامه)

- با این همه رابطه اخیر آنومالی دارد.
  - **در درج:** در درس c_1، کتاب b_8 نیز به عنوان مرجع درس ثبت شود.

نمی‌توانیم بگوییم چون کلید نداریم نمی‌توانیم درج کنیم. باید قواعد معنایی رعایت شود.

باید درج کنیم: $\langle c_1, t_1, b_8 \rangle$

$\langle c_1, t_2, b_8 \rangle$

$\langle c_1, t_3, b_8 \rangle$

یعنی عمل منطقاً تاپلی تبدیل شده به عمل مجموعه‌ای

- در حذف و بهنگام‌سازی هم به دلیل وجود افزونگی، آنومالی داریم.
- رابطه CTB باید تجزیه شود تا رابطه‌های حاصل 4NF شود.

فرم نرمال 4NF (ادامه)

- دلیل آنومالی این رابطه، وجود پدیده MVD است.

$$\begin{cases} C\# \rightarrow\rightarrow B\# \\ C\# \rightarrow\rightarrow T\# \end{cases}$$

- پس CTB را باید چنان تجزیه کنیم که در رابطه‌های حاصل، MVD وجود نداشته باشد.
- برای این کار CTB را پرتوگیری می‌کنیم به نحوی که در عنوان هر پرتو، مبدأ MVD وجود داشته باشد.

CT (C#, T#)

c_1	t_1
c_1	t_2
c_1	t_3
c_2	t_4
c_2	t_2

CB (C#, B#)

c_1	b_1
c_1	b_2
c_2	b_3
c_2	b_5
c_2	b_7
c_1	b_8

درج به صورت عملاً تاپلی و نه مجموعه‌ای

- رابطه‌های جدید آنومالی CTB را ندارند.
- این دو رابطه جدید BCNF هستند، چون تمام کلید هستند. MVD مهم ندارند، پس 4NF هستند.
- **تمرین:** نشان دهید با پیوند این دو رابطه، رابطه اصلی به دست می‌آید.

فرم نرمال 4NF (ادامه)

- **قضیه فاگین (Fagin):** رابطه $R(A, B, C)$ به دو پرتوش $R_1(A, B)$ و $R_2(A, C)$ **تجزیه بی‌کاست** (Nonloss) می‌شود اگر و فقط اگر $A \rightarrow\rightarrow B$.
- قضیه فاگین (برای MVD) تعمیم قضیه هیث (برای FD) است.
- آیا می‌توان گفت مفهوم MVD تعمیم مفهوم FD است؟ آیا می‌توان گفت FD حالت خاصی از MVD است؟
  - FD حالت خاصی از MVD است که در آن مجموعه مقادیر صفت وابسته، تک عنصری هستند.
- همچنین این استنتاج منطقی را هم داریم:

If $A \rightarrow B$ **then** $A \rightarrow\rightarrow B$

فرم نرمال 4NF (ادامه)

- **نکته:** بحث 4NF از یک دیدگاه می‌تواند اصلاً موضوعیت نداشته باشد. زیرا رابطه‌ای که BCNF باشد و MVD داشته باشد قطعاً صفت چندمقداری دارد و می‌دانیم در طراحی برای صفات چندمقداری، از همان ابتدا می‌توان رابطه‌های جداگانه طراحی کرد.
- با این همه مفهوم MVD به عنوان بیان فرمال صفت چندمقداری قابل توجه است.

کجای؟ تعریف زاینولو از 3NF، BCNF، 4NF و ... مطالعه شود.

وابستگی پیوندی (JD): رابطه R وابستگی پیوندی به n پرتو R_1، R_2، ... و R_n دارد اگر و فقط اگر R حاصل پیوند این n پرتو باشد.

$R=[JD]*(R_1, R_2, \ldots, R_n)$

مثال

$CTB=[JD]*(CT, CB)$

- JD را نامهم گوییم هرگاه عنوان (Heading) یکی از R_iها همان عنوان (Heading) رابطه R باشد.

5NF [PJNF]– فرم نرمال پرتو پیوندی: رابطه R در 5NF است اگر و فقط اگر تمام JDهای آن ناشی از CK باشد. ⇐ ناشی از CK بودن یعنی عنوان همه پرتوها، در همه JDها، سوپرکلید باشد.

- رابطه CTB در 5NF نیست، چون (C#, T#) و (C#, B#) سوپرکلید رابطه CTB نیستند.

فرم نرمال 5NF (ادامه)

مثال

STUD (STID, STNAME, STJ, STL)

- فرض می‌کنیم که 3NF هست و FD مزاحم نداریم.

STN (STID, STNAME)
SJL (STID, STJ, STL) ⇒ STUD= [JD]*(STN, SJL) JD به دو پرتو

STN (STID, STNAME)
SJ (STID, STJ) ⇒ STUD= [JD]*(STN, SJ, SL) JD به سه پرتو
SL (STID, STL)

- رابطه STUD در 5NF است. چون عنوان همه پرتوها در همه JDهای آن، سوپرکلید هستند (ناشی از کلید کاندید هستند).

فرم نرمال 5NF (ادامه)

- **نکته:** اگر رابطه‌ای در 3NF باشد و تمام CKهای آن ساده باشند، آن رابطه در 5NF است.

مثال

PCD رابطه‌ای است که در 4NF است ولی در 5NF نیست. JD به دو پرتو ندارد، بلکه به سه پرتوش JD دارد، که هیچکدام سوپرکلید نیستند.

- استاد PR# درس CO# را در دانشکده D# ارائه می‌دهد.

PCD (PR#, CO#, D#)

PR_1	co_1	d_2
PR_1	co_2	d_1
PR_2	co_1	d_1
PR_1	co_1	d_1

مثال

رابطه SPJ تمام کلید است. ⇐ حداقل BCNF

MVD ندارد. ⇐ 4NF است.

SPJ (S#, P#, J#)

S_1	P_1	J_1
S_1	P_1	J_2
S_1	P_2	J_1
S_2	P_1	J_1

فرم نرمال 5NF (ادامه)

- فرض می‌کنیم بخواهیم این رابطه را تجزیه کنیم:

SP (S#, P#)

S_1	P_1
S_1	P_2
S_2	P_1

PJ (P#, J#)

P_1	J_1
P_1	J_2
P_2	J_1

$\bowtie$

SPJ' (S#, P#, J#)

S_1	P_1	J_1
S_1	P_1	J_2
S_1	P_2	J_1
S_2	P_1	J_2
S_2	P_1	J_1

تاپل حشو

- این رابطه JD به دو پرتوش ندارد.
- یک پرتو دیگر هم می‌گیریم:

SJ (S#, J#)

S_1	J_2
S_1	J_1
S_2	J_1

$\bowtie$

SPJ (S#, P#, J#)

۱	S_1	P_1	J_1
۲	S_1	P_1	J_2
۳	S_1	P_2	J_1
۴	S_2	P_1	J_1

فرم نرمال 5NF (ادامه)

- پس SPJ، JD دارد به سه پرتوش و نه کمتر: SPJ= [JD]*(SP, PJ, SJ)

و 5NF نیست چون عنوان (Heading) پرتوهایش سوپرکلید نیست.

- در این مثال از سه فقره اطلاع دو موجودیتی، باید یک اطلاع سه موجودیتی را استنتاج کنیم، چرا که این یک محدودیت جامعیتی حاکم بر محیط است (وجود وابستگی پیوندی).

- توجه داشته باشید که در حالت کلی چنین استنتاجی درست نیست و پدیده دام پیوندی حلقه‌ای بروز می‌کند، ولی در اینجا به دلیل وجود وابستگی پیوندی، چنین مشکلی بروز نمی‌کند.

فرم نرمال 5NF (ادامه)

نکته: در این رابطه یک محدودیت بسیار نادر، موسوم به محدودیت با **ماهیت چرخشی (CC)** وجود دارد.

- با وجود تاپل‌های دوم تا چهارم در رابطه SPJ باید تاپل (S_1, P_1, J_1) نیز وجود داشته باشد.
- این محدودیت ناشی از وجود (S_1,P_1) در تاپل دوم، (S_1,J_1) در تاپل سوم و (P_1,J_1) در تاپل چهارم است.
- در واقع مقدار هر یک از سه صفت در سه تاپل از چهار تاپل رابطه SPJ یکسان است و در هر یک از سه پرتو دوتایی، یک صفت مشترک با دو پرتو دیگر وجود دارد.
- به دلیل وجود این محدودیت باید گروههای ۴ تاپلی در بدنه رابطه وجود داشته باشند.

- اگر یک رابطه CC داشته باشد در فرم نرمال 5NF نیست.

- برای تشخیص این محدودیت در رابطه درجه n دوتست انجام می‌دهیم:

۱- تعداد تاپل‌ها: n+1

۲- مقدار هر صفت، در n تاپل یکسان باشد.

فرم نرمال 5NF (ادامه)

مثال: در رابطه R هر ترکیب دوتایی CK است. لذا در فرم نرمال 5NF است زیرا:

- سه دترمینان دارد که هر سه CK هستند. ⇐ BCNF است.
- MVD ندارد. ⇐ 4NF است.
- CC ندارد و همه JDهای آن ناشی از کلید کاندید هستند. ⇐ 5NF است.

R (A, B, C)

a_1	b_1	c_2
a_1	b_1	c_1
a_2	b_1	c_1

فرم نرمال 6NF

تعریف: رابطه R در 6NF است هر گاه اصلاً JD نداشته باشد.

- **نکته:** در رابطه درجه n، اگر غیر از کلید فقط یک صفت دیگر داشته باشد، در 6NF است.

مثال: به طور مثال رابطه SPJ که 5NF نبود را به سه رابطه SP، SJ و PJ تجزیه می‌کنیم.

این سه رابطه در فرم نرمال 5NF و 6NF هستند.

کجای؟: فرم نرمال DKNF چیست؟

- تئوری نرمال‌ترسازی به عنوان ابزار طراحی RDB، مزایا و معایبی دارد.
- **مزایای تئوری نرمال‌ترسازی:**

۱- ارائه یک طراحی واضح از خُردجهان واقع (Clean Design)؛ یعنی با کمترین اختلاط اطلاعات. یعنی در واقع رعایت یک اصل در عمل (one fact : one table).

۲- کاهش بعض افزونگی‌ها؛ آن افزونگی‌هایی که با پرتوگیری از بین می‌روند (کاهش می‌یابد).

۳- کاهش بعض آنومالی‌ها [ناشی از اختلاط اطلاعات].

۴- بعض قواعد جامعیت را اعمال می‌کنیم (ناشی از وابستگی بین صفات).

- این تئوری به طراح کمک می‌کند تا تصمیم بگیرد چند رابطه داشته باشد و هر رابطه عنوانش چه باشد و کلیدش چه باشد.

جمع‌بندی (ادامه)

معایب تئوری نرمال‌ترسازی:

۱- فزون‌کاری در بازیابی (اگر کاربر به هر دلیلی رابطه اصلی را بخواهد، عمل پیوند (Join) باید انجام شود که در حجم بالای داده، سربار زیادی دارد).

به دلیل همین عیب، گاه در عمل لازم است غیرنرمال‌سازی (Denormalization) انجام دهیم.

یعنی تبدیل حداقل دو رابطه (i+1)NF به یک رابطه (i)NF.

۲- فرآیند نرمال‌ترسازی زمان‌گیر است به ویژه اگر مجموعه صفات محیط بزرگ باشد و نمودار FDها گسترده باشد.

۳- مبتنی است بر یک فرض نه چندان واقع‌بینانه [فرض: در آغاز مجموعه‌ای از صفات داریم در یک مجموعه Universal، آنگاه با روش سنتز صفات (دسته‌بندی صفات) به تعدادی رابطه می‌رسیم.] در حالیکه در عمل ابتدا روش بالا به پایین و رسیدن به تعدادی رابطه با درجه متعارف، آنگاه استفاده از ایده‌های این تئوری برای تست نرمالیتی (اول تست 3NF، بعد BCNF و 5NF).

۴- همه وابستگی‌های بین صفات دیده نشده‌اند؛ مثلاً وابستگی شمول دیده نشده است.

۵- ایجاد میزانی افزونگی؛ چون اگر بخواهیم تجزیه خوبی داشته باشیم، یا CK باید در همه پرتوها تکرار شود یا پیوندهای CK-FK وجود داشته باشد!

۶- استفاده محدود از عملگرهای جبر رابطه‌ای. تجزیه ← پرتو بازسازی ← پیوند

حال آنکه در عمل گاه لازم است رابطه را تجزیه افقی کنیم:

$$ST_1 = \sigma_{STJ='Phys'}(STUD)$$

$$ST_2 = \sigma_{STJ='IT'}(STUD)$$

$$\dots$$

$$ST_n = \sigma_{STJ='Comp'}(STUD)$$

$$STUD = \cup_{i=1}^{n}(ST_i)$$

جمع‌بندی (ادامه)

- به رابطه‌های ناشی از تجزیه افقی می‌گوییم:

فرم نرمال گزینش اجتماع (تحدید اجتماع) RUNF (Restriction Union Normal Form)

- RUNF لزوماً در امتداد فرم‌های نرمال نیست. به موازات آنها مطرح است. یعنی ممکن است رابطه 3NF باشد، تجزیه افقی کنیم و باز هم 3NF باشد.

کجای؟ در چه شرایطی رابطه حاصل از تجزیه افقی از خود رابطه نرمال‌تر است؟

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

اصول طراحی و پیاده سازی پایگاه داده ها

یادداشت "تکمیلی"
(سری I)

این مجموعه شامل مطالب زیر است :

۱. مشی ناپایگاهی و مشی پایگاهی: در یک نگاه

۲. پایگاه داده هادر سازمان ها

۳. نماد های نمایش نمودار مدلسازی معنایی داده‌ها(ERD)

۴. جدول مقایسهٔ DSها

۵. جنبه های فایلینگ پایگاه داده ها

۶. در باره ی DBMS و تکنولوژی پایگاه داده ها

۷. در بارهٔ DBA (و ریز فعالیت های طراحی و پیاده سازی پایگاه داده ها)

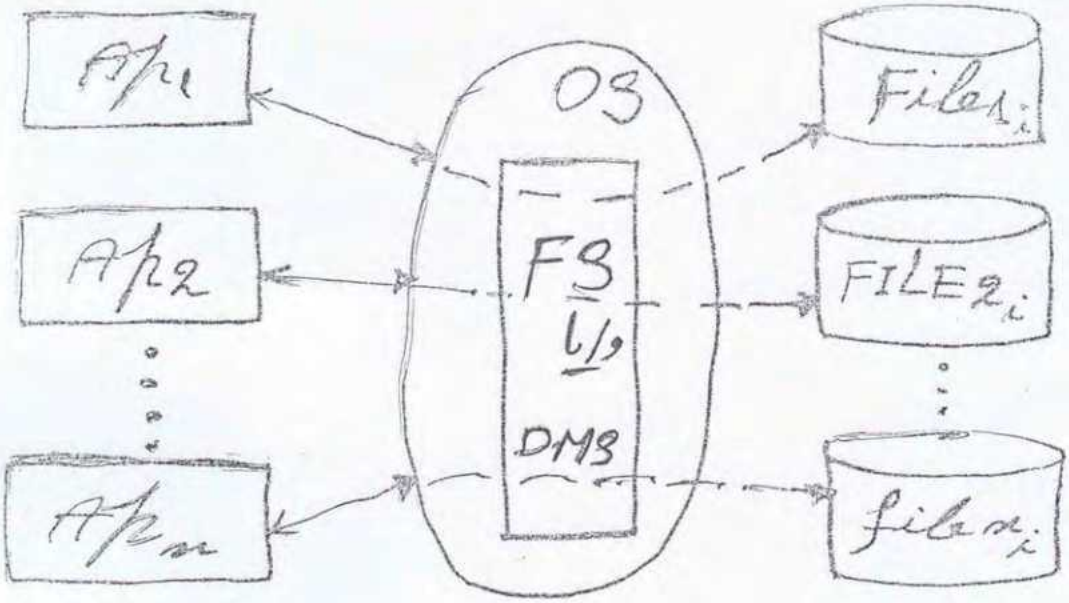
۸. درباره کاربر

B

۱- هسته ناپایگاهی و هسته پایگاهی: در یک نگاه

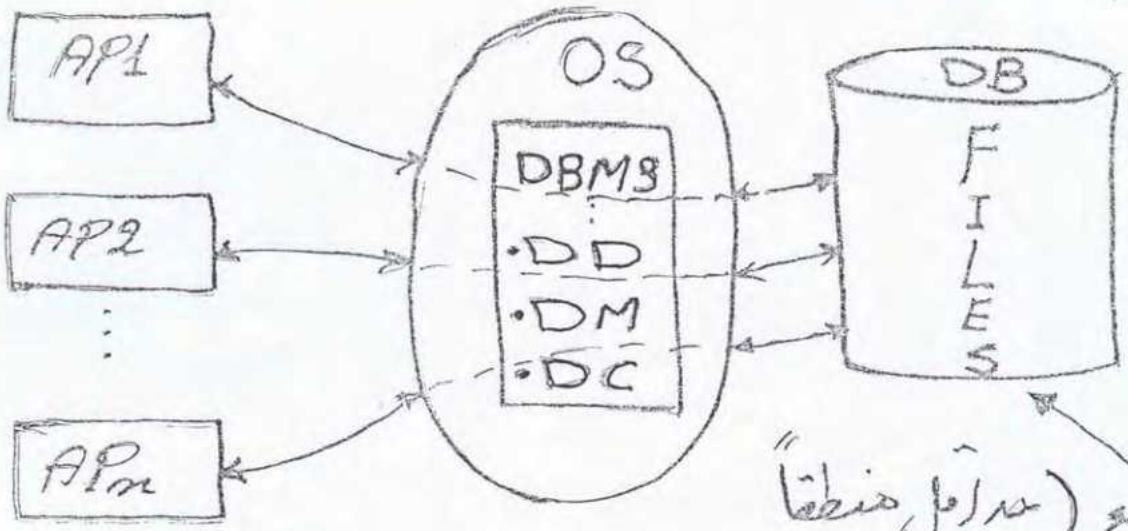
- رده بندی کلی سیستمهای داده داری - داده پردازی { - ناپایگاهی - پایگاهی

▷ هسته ناپایگاهی:



$i = 1, \ldots$

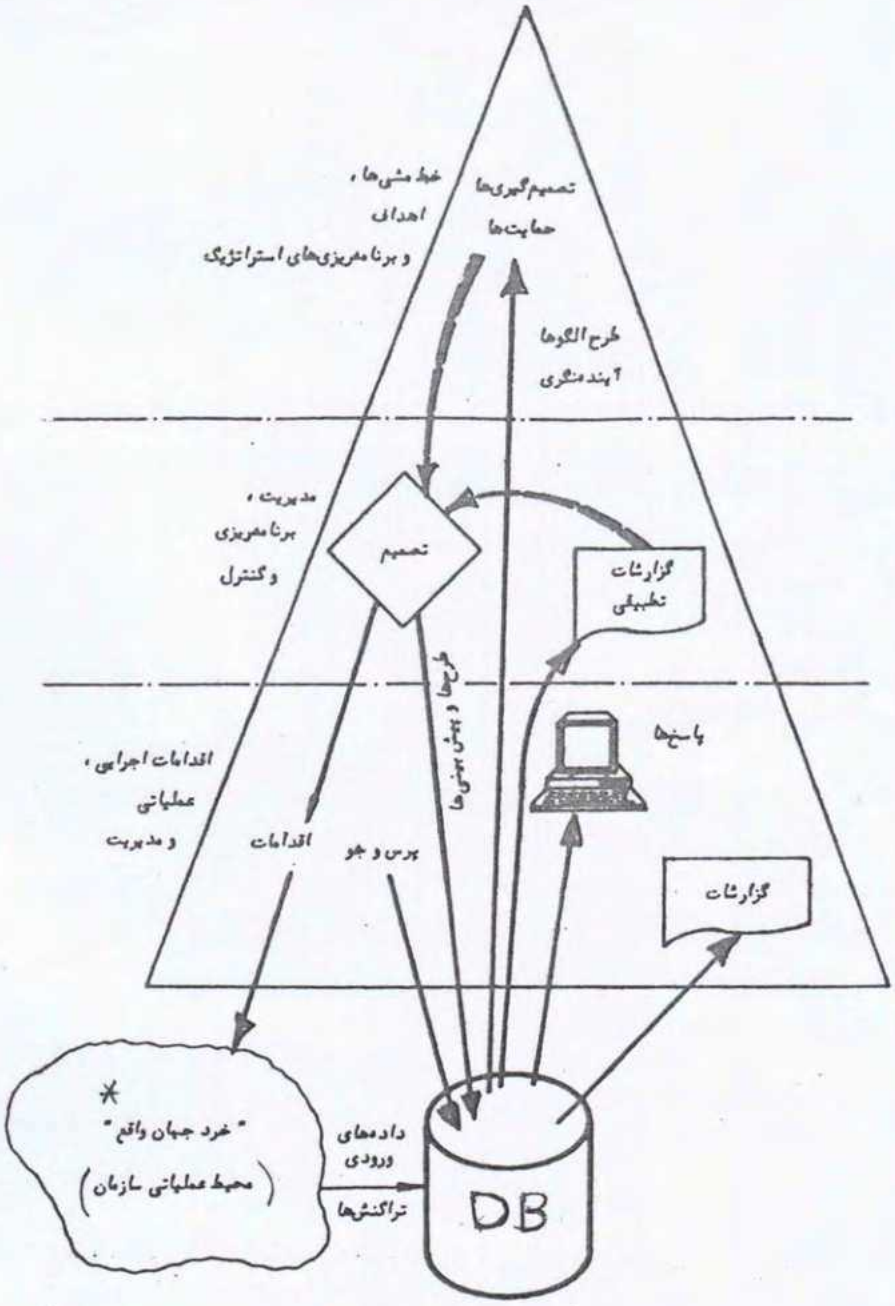
▷ هسته پایگاهی:



مجتمع (حداقل منطقاً و گاه بطور فیزیکی)

DD: Data Definition
DM: Data Manipulation
DC: Data Control

۲- پایگاه داده‌ها در سازمانها

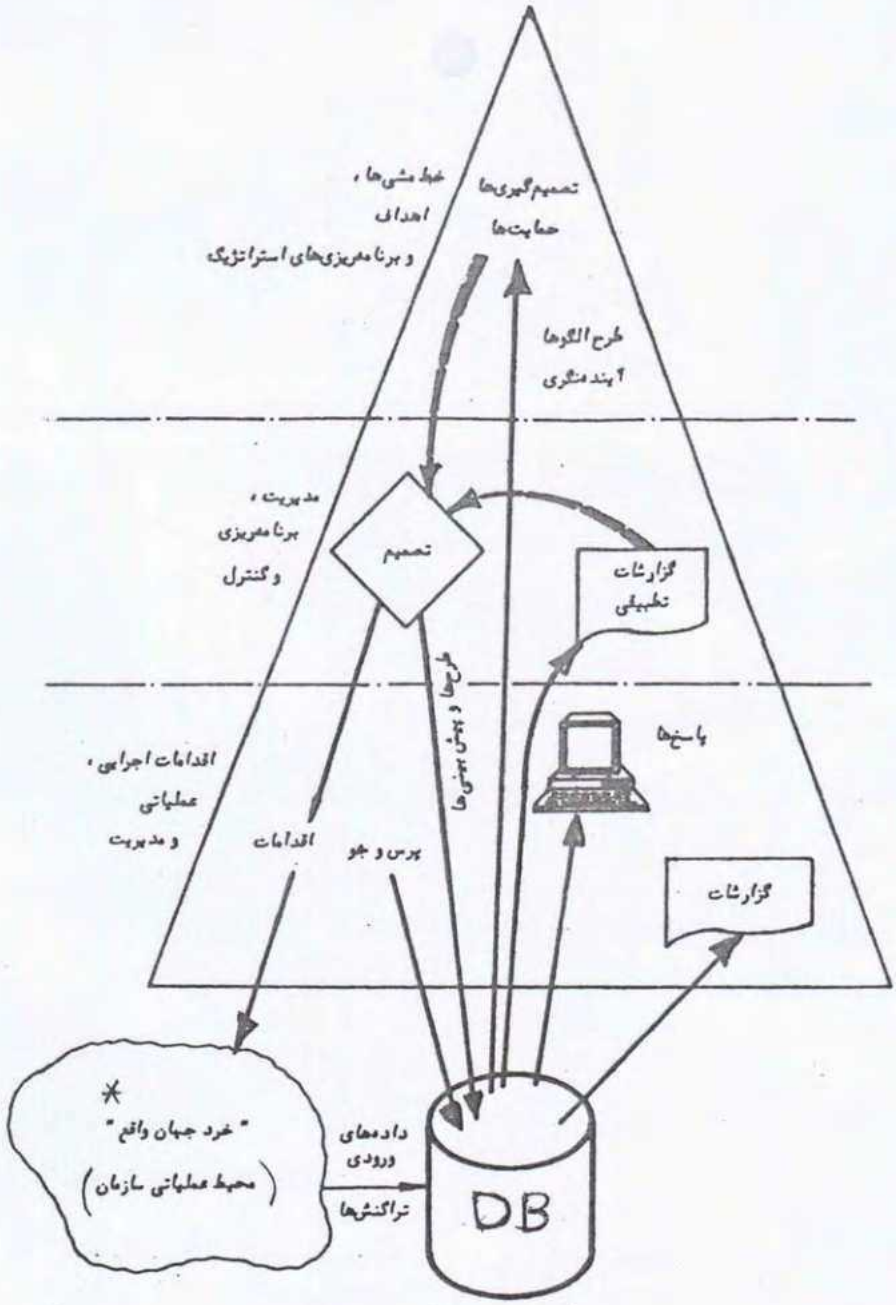


نقش اطلاع و پایگاه داده‌ها در سازمانها

* Micro real world (Miniworld یا Universe of Discourse (UoD))

منبع:
" مقدمه‌ای بر پایگاه داده‌ها ... "

۲- پایگاه داده‌ها در سازمانها

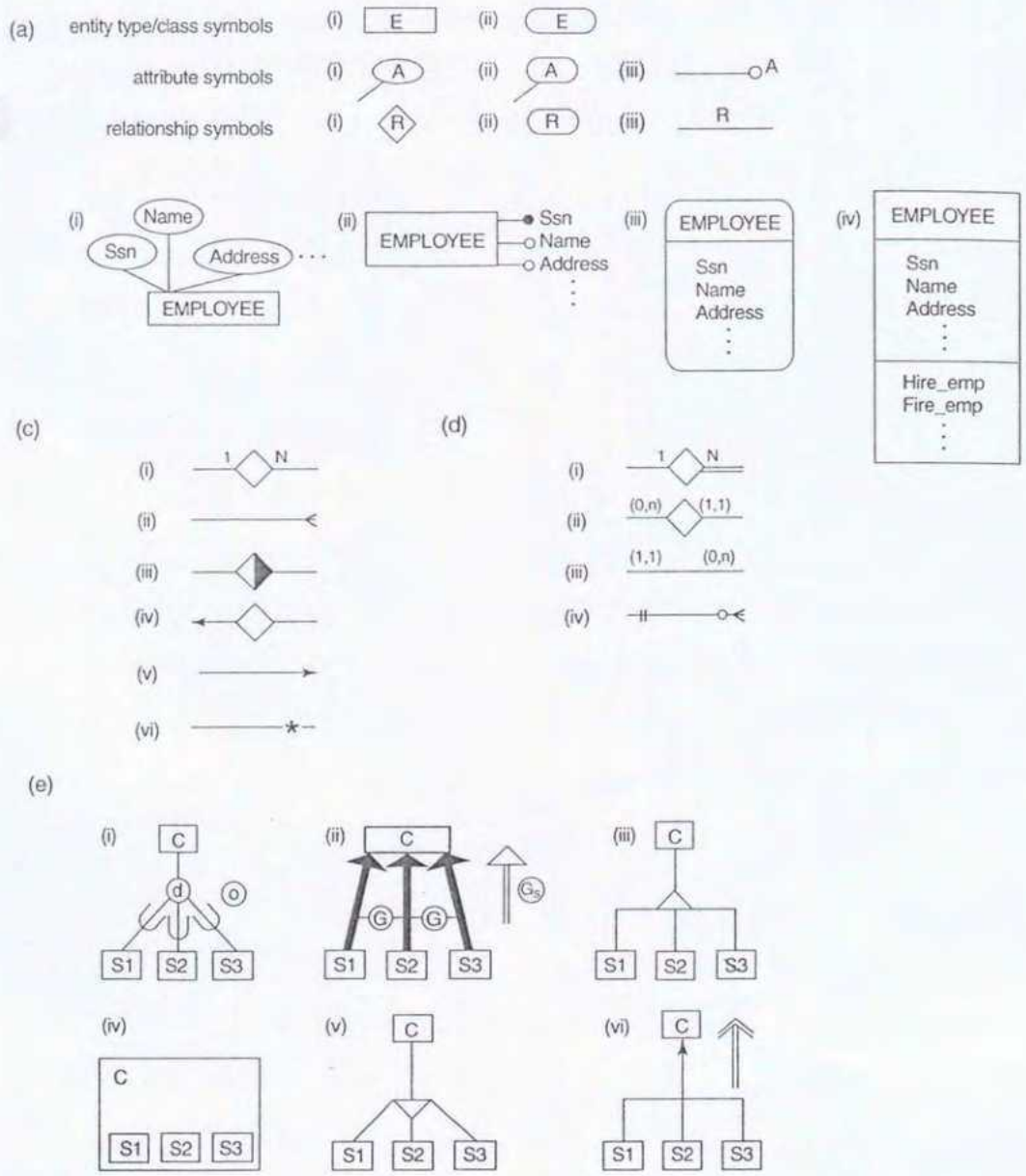


نقش اطلاع و پایگاه داده‌ها در سازمانها

* Micro real world (Miniworld یا Universe of Discourse (UoD))

منبع:
" مقدمه‌ای بر پایگاه داده‌ها ... "

نمادهای دیگر برای نمایش "مدلسازی معنایی" :



Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

* منبع : ELMA 03 (ضمیمه A از این کتاب)

B

۴- جدول مقایسه DSها
(از مبحث DS شناسی)

منبع: مفاهیم بنیادی پایگاه داده‌ها

ردیف	ویژگی ↓	TDS [RDS] (صرفنظر از بعضی ویژگی‌های دیگر RDM)	HDS [از HDM]	NDS [از NDM]
۱	وضوح نمایش	زیاد	کم	کمتر
۲	مبنای ریاضی ← میزان انتزاع	دارد ← زیاد	ندارد ← کم	ندارد ← کم
۳	تعداد عناصر ساختاری اساسی	یک: مفهوم رابطه (نوع جدول)	دو: نوع رکورد، PCL	دو: نوع رکورد، مجموعه کود اسیل
۴	قابلیت نمایش ارتباط با چندی 1:1، 1:N و M:N	دارد	در M:N مشکل دارد	دارد
۵	قابلیت نمایش ارتباط n-گانی $n>2$	دارد	با دشواری	دارد
۶	منطق جستجو در DS [و نه لزوماً در محیط فایلینگ]	ساده	پیچیده	پیچیده‌تر
۷	تعداد کلاس‌ها در ساخت منطقی عملگر ناوش	سه	چهار	پنج
۸	تعامل { رویه قرینه برای پرسش‌ها، قرینه	دارد	ندارد	دارد (به صحت فرونمایی)
۹	زبان پایگاه	نارویه‌ای	رویه‌ای	رویه‌ای
۱۰	ناوش (جستجو)	اتوماتیک	غیراتوماتیک	غیراتوماتیک
۱۱	عملگرهای ذخیره‌سازی [در کادر DS]	ساده	سادگی کمتر	باز هم کمتر
۱۲	قواعد جامعیت ذاتی	ندارد	دارد	دارد
۱۳	آنومالی در عملیات ذخیره‌سازی	ندارد (به شرط طراحی خوب)	دارد	ندارد (به شرط طراحی خوب)
۱۴	افزونگی ناشی از ماهیت	دارد	دارد	دارد

۵- جنبه‌های فایلینگ پایگاه داده‌ها (سطح داخلی و فیزیکی)

(* توجه! بسیار مهم در انتخاب DBMS)

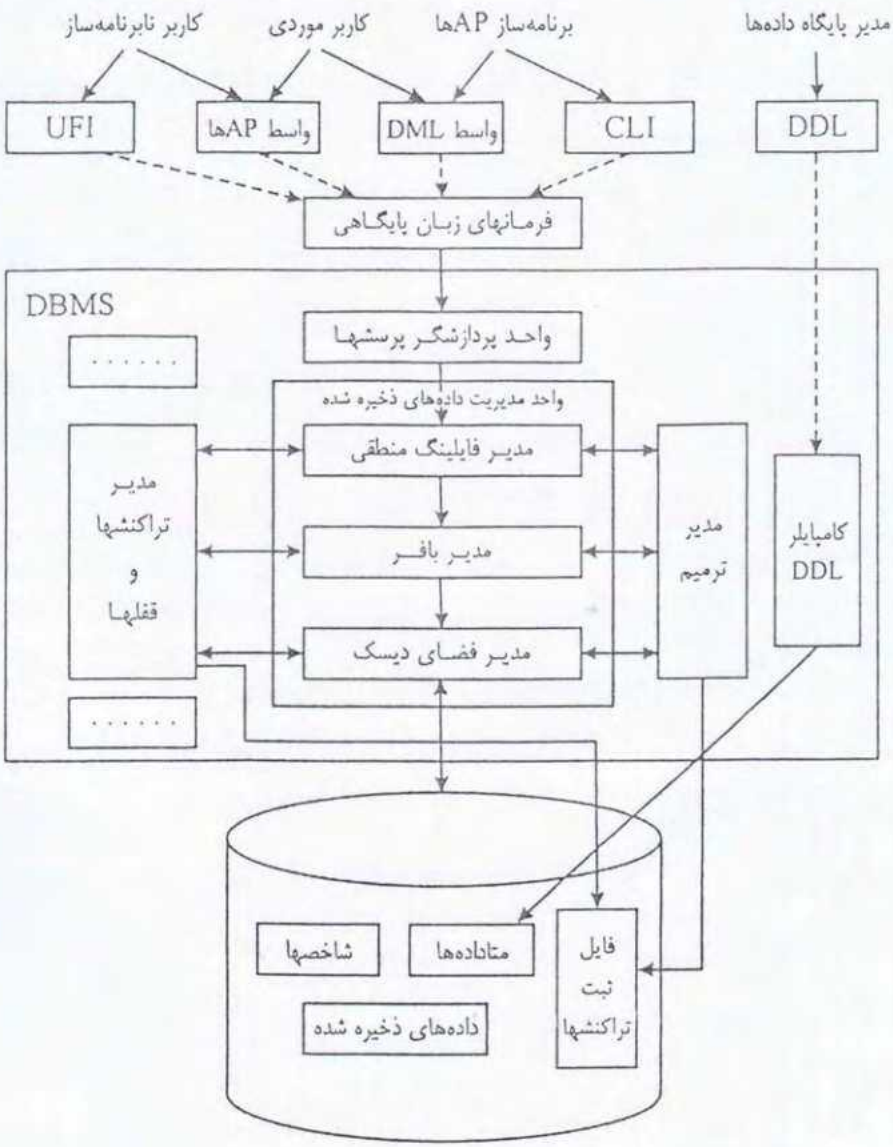
B

سطح داخلی و فیزیکی

نوع داده	نام فیلد	طول داده (حداکثر - پیش فرض)	حداکثر طول رکورد
حداکثر تعداد رکورد		ثابت یا متغیر بودن طول رکورد	حداکثر تعداد فیلد در هر نوع رکورد
حداکثر طول هر فیلد		حداکثر تعداد فایلهای باز در یک زمان	حداکثر تعداد ترانزاکشن‌ها
اندازه شاخص	هزینه ایجاد شاخص	شاخص، ساختمان درونی شاخص، درجه پویایی شاخص	حداکثر طول کلید
نوع کلید	نحوه انجام عملیات	ساختارهای فایل موجود	حداکثر تعداد کلید
نحوه نشست فایلها روی رسانه خارجی		ذخیره سازی در محیط داخلی - فیزیکی و سرعت انجام هر عمل	
چگونگی نگاشت (تناظر) بین فایلها و ساختارهای سطح ادراکی		نحوه پیاده سازی "هیچ مقدار" در این سطح	
فشرده سازی داده‌ها و نسبت آن		خوشه‌ای بودن داده‌ها (رکوردها، فایلها): خوشه‌ای بودن درون فایلی و بین فایلها	
اندازه و تعداد بافر مورد نیاز		فشرده سازی شاخصها و نسبت آن	
اندازه بلاک		اندازه صفحه (ثابت یا متغیر)	
وجود DSL داخلی و امکانات آن			
میزان دخالت DBA در سطح داخلی - فیزیکی و کنترل آن			

۶- درباره DBMS و تکنولوژی پایگاه داده‌ها

۶-۱: واحدهای سیستم، واسطهای تعامل با سیستم



ساختار یک سیستم پایگاهی

۶-۱-۱: طرح I:

(طرحهای II و III در صفحه بعد)

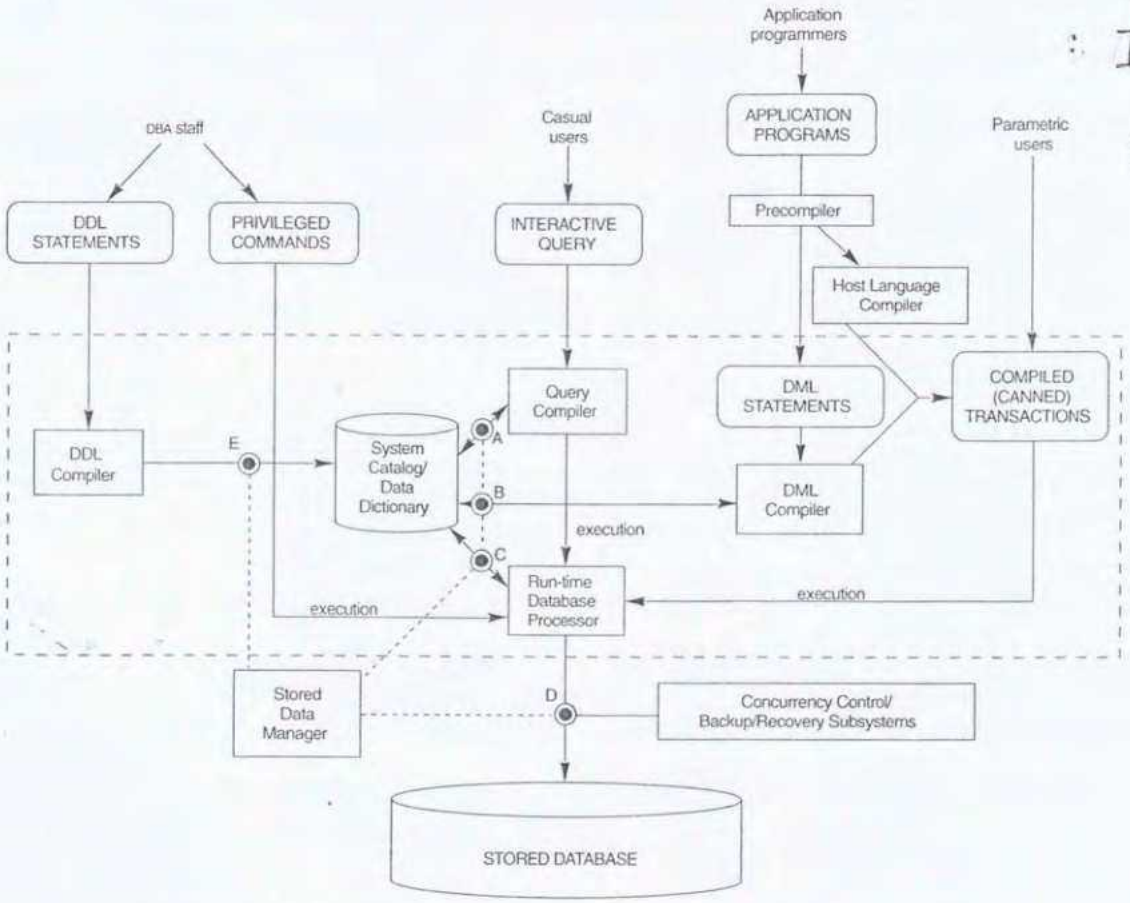
۶-۲: جایگاه DBMS

جایگاه DBMS در یک سیستم کامپیوتری

۶-۱-۲ :

طرح II :

(ELMA03

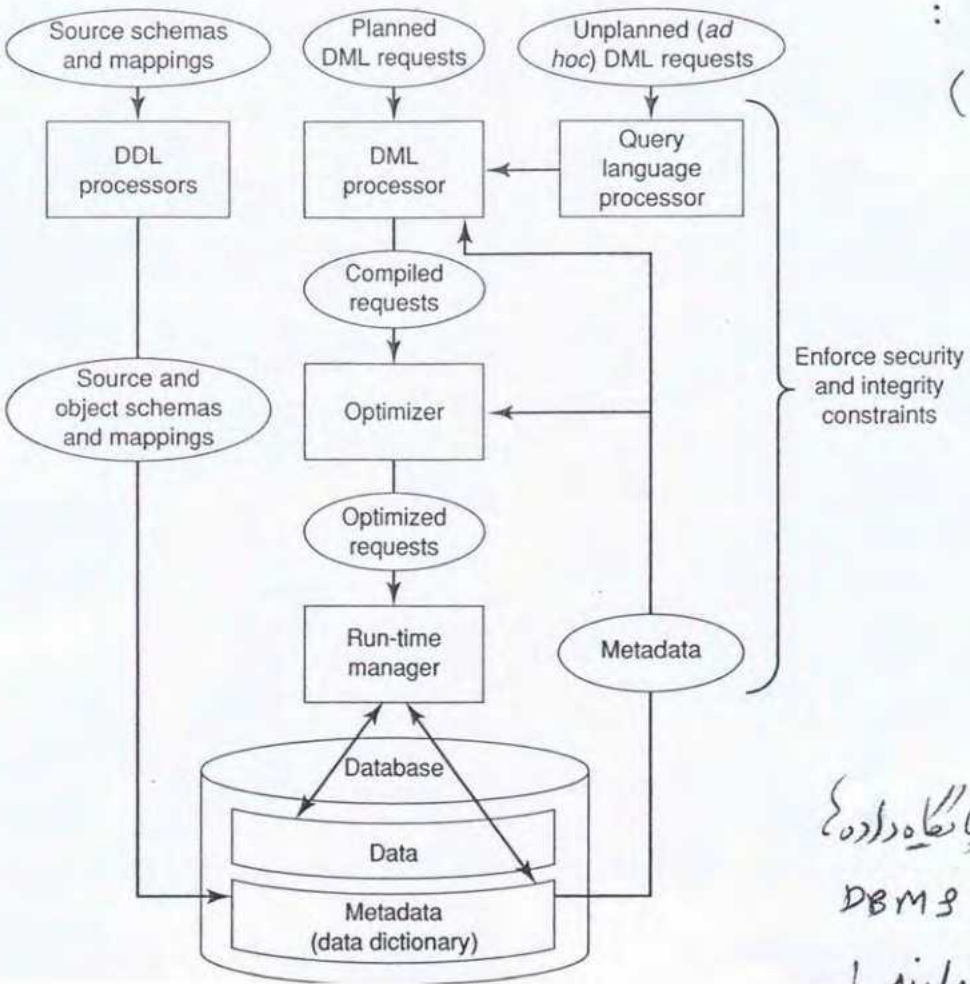


Component modules of a DBMS and their interactions.

۶-۱-۳ :

طرح III :

(DATE 03)



* تمرین: اجزاء DBMS حداقل در دو کتاب معتبر دیگر بررسی شود.

توجه!

مؤلفین متون پایگاه داده‌ها در مورد اجزاء DBMS

از نگاهی نزدیک به این نرم‌افزار، می‌توان دریافت که این نرم‌افزار از سه لایه[2] تشکیل شده است :

- لایه هسته (سیستم کنترل یا موتور[1] پایگاه‌داده‌ها)
- لایه مدیریت محیط پایگاه‌داده‌ها[2]
- لایه تسهیلات نرم‌افزاری[3] (ابزارها)

در هریک از سه لایه واحدهایی وجود دارد . نام واحدها ، تعداد آنها و وظیفه هر واحد در سیستم‌های مختلف ، فرق دارد . همچنین در متون آکادمیک خارجی نام واحدها و ارتباطات بین آنها ، یکسان نیست (برای بررسی این مطلب از جمله می‌توان به [DATE 2003] ، [ELMA 2003] ، [SILB 06] ، [CONN 99 , 2005] و [ULLM 97 , 2002] مراجعه کرد) . در اینجا بعضی از واحدهای دو لایه اصلی را نام می‌بریم . برای اطلاع از لایه تسهیلات نرم‌افزاری به یادداشت تکمیلی گفتار مراجعه شود . در یادداشت تکمیلی همچنین سه طرح ، نمایش اجزاء DBMS و ارتباط بین آنها ، وجود دارد .

۶-۳-۱ : واحدهای لایه هسته

۱. واحد دریافت درخواست کاربر و وارسی‌های اولیه
۲. واحد تولید شما[4]ها(گاه موسوم به پردازنده DDL و DCL)
۳. پیش کامپایلر(ها) برای DML (ادغام شده)
۴. کامپایلر (پردازنده DML)
۵. پردازشگر پرسش[5] و بهینه‌ساز پرسش[6]
۶. واحد مدیریت سطح داخلی (مدیریت فایلینگ منطقی)
۷. واحد مدیریت بافر (گاه موسوم به مدیریت حافظه نهان) .
۸. واحد مدیریت فضای دیسک (این واحد می‌تواند درخود سیستم عامل باشد) .
۹. واحد ناظر زمان اجرا[1] (مسئول اجرای دستورات پایگاهی گاه موسوم به مدیر زمان اجرا[2])
۱۰. واحد مدیریت همروندی تراکنش‌ها[3] (بویژه در سیستم‌های چند کاربری)
۱۱. واحد مدیریت انتقال داده‌ها[4]
۱۲. واحد مدیریت کاتالوگ[5]

در برخی سیستمها ممکن است دو یا بیش از دو واحد به صورت یک واحد و تحت یک نام خاص پیاده‌سازی شده باشند . ضمناً واحدهای برشمرده ، واحدهای مهمتر هستند .

۶-۳-۲ : واحدهای لایه مدیریت محیط پایگاه داده‌ها

۱. واحد کنترل جامعیت پایگاه داده‌ها
۲. واحد ترمیم (بازسازی) پایگاه داده‌ها
۳. واحد ایمنی و حفاظت[6] پایگاه داده‌ها
۴. واحد تولید نسخه‌های پشتیبان
۵. واحد تولید فایل‌های ثبت تراکنش‌ها

نکته : پیشتر هم اشاره شد که نسخه‌های پشتیبان و فایل‌های ثبت تراکنشها ، ابزارهای لازم برای مدیریت محیط پایگاه داده‌ها یعنی ترمیم ، ایمنی ، حفاظت و کنترل جامعیت هستند (به مباحث درس "پایگاه داده‌های پیشرفته" رجوع شود) .

1- Engine
2- Database envirnoment managment
3- Software facilities (tools)
4- Schema generator

1- Run time supervisor
2- Run time manager
3- Transactions concurrency management
4- DC manager
5- Catalog manager

B

۶-۴: رده‌بندی سیستم‌های مدیریت پایگاه‌داده‌ها

این نرم افزار را می‌توان از چندین نظر رده‌بندی کرد :

۶-۴-۱ : از نظر نوع ساختار داده‌ای

- سیستم رابطه‌ای
- سیستم سلسله‌مراتبی
- سیستم شبکه‌ای
- جز اینها

نکته ۱ : هرچند باتوجه به معماری پایگاه‌داده‌ها ، وجود سطوح انتزاعی و در نتیجه وجود یک ساختار داده‌ای تأمین کننده انتزاع ، الزامی است و ساختار داده‌ای در واقع هویت سیستم را مشخص می‌کند ، اما نرم‌افزارهایی وجود دارند که از نظر مجموعه توانشهای عملیاتی ، تا حدی مثل یک سیستم واقعی بوده ، کارایی آنها نیز قابل توجه است (مثل سیستم ADABAS ، سیستم NOMAD و ...) و جزء رده‌بندی ذکر شده هم نیستند.

نکته ۲ : اگر وجود مدل داده‌ای شیءگرا (ODM) را بپذیریم ، آنگاه در رده‌بندی بالا ، باید OODBMS را نیز وارد کرد.

۶-۴-۲ : از نظر محیط سخت افزاری

- وابسته به یک محیط خاص
- ناوابسته به یک محیط خاص

درگونه دوم می‌گوییم که سیستم از نظر سخت‌افزاری ، جابجایی پذیری[۲] دارد.

۶-۴-۳ : از نظر رده کامپیوتر

- خاص محیط کامپیوترهای شخصی
- خاص محیط کامپیوترهای متوسط
- خاص محیط کامپیوترهای بزرگ
- خاص محیط کامپیوترهای خیلی بزرگ
- اجراشونده درچند رده کامپیوتر

نکته ۳ : رده‌بندی ۳ با رده‌بندی ۲ لزوماً یکی نیست . در رده‌بندی ۲ سیستم می‌تواند جابجایی پذیری داشته باشد اما بین انواع کامپیوترهای از یک رده ، مثلاً رده کامپیوترهای شخصی.

B

۶-۴-۴ : از نظر محیط سیستم عامل

- وابسته به یک سیستم‌عامل خاص
- اجراشونده در محیط چند سیستم‌عامل

درگونه دوم می‌گوییم که سیستم از نظر سیستم‌عامل ، جابجایی پذیری دارد.

۶-۴-۵ : از نظر نوع معماری سیستم پایگاه داده‌ها

- با توانش ایجاد پایگاه متمرکز
- با توانش ایجاد پایگاه نامتمرکز

درگونه دوم می‌گوئیم که DDBMS داریم و به بیان دیگر سیستم دارای توانش ایجاد و مدیریت پایگاه‌داده‌های توزیع‌شده است و البته باید دید آیا سیستم فقط در معماری همگن عمل می‌کند یا در معماری ناهمگن هم می‌تواند عمل کند

۶-۴-۶ : از نظر معماری مشتری - خدمتگزار

- با توانش ایجاد معماری چند مشتری - یک خدمتگزار
- با توانش ایجاد معماری چند مشتری - چند خدمتگزار

رده‌بندی ۶ لزوماً همان رده‌بندی ۵ نیست. سیستم با معماری مشتری - خدمتگزار حالت خاصی از سیستم با معماری توزیع شده است (به گفتار نهم مراجعه شود).

نکته ۴ : درگونه اول سیستم را تک خدمتگزار و درگونه دوم چند خدمتگزار می‌گوییم. در گونه دوم تعداد خدمتگزارها (S) قابل توجه است و باید دید آیا سیستم فقط با معماری SSM عمل می‌کند یا در معماری MSM هم می‌تواند عمل کند.

۶-۴-۷ : از نظر زبان

- سیستم دارای SQL
- سیستم فاقد SQL

نکته ۵ : SQL در واقع زبان استانده سیستمهای رابطه‌ای است و از این رو خود مبنایی برای رده‌بندی است.

۶-۴-۸ : از نظر نوع زبان داده‌ای فرعی

- دارای I.DSL
- دارای E.DSL
- دارای E/I.DSL

در گونه E.DSL باید دید آیا سیستم دارای پیش‌کامپایلر است یا از طریق حکم فراخوانی موجود در زبان میزبان عمل می‌کند.

B

۶-۴-۹ : از نظر ماهیت زبان داده‌ای فرعی

- با زبان رویه‌ای
- با زبان نارویه‌ای

۶-۴-۱۰ : از نظر سیستم فایل

- خودکفا
- وابسته به سیستم فایل محیط سیستم عامل

نکته ۶ : در هر دوگونه ، تعداد "شیوه دستیابی " و تنوع ساختار فایل‌ها باید مورد توجه قرار گیرد.

کنجکاوی ۱ : چرا؟

۶-۴-۱۱ : از نظر نوع کاربرد

- تک منظوره
- همه منظوره

نکته ۷ : تک منظوره یعنی سیستم برای کاربرد خاصی طراحی و تولید شده است ، مثلاً برای کاربرد هواشناسی ، مهندسی ، هواپیمایی ، جغرافیایی و ... . سیستم همه منظوره ، می‌تواند درکاربردهای گوناگون استفاده شود. البته سیستم‌های موجود (از جمله RDBMS های متعارف) در بعضی از کاربردها (به‌ویژه کاربردهای جدید) ، نمی‌توانند به درستی عمل کنند ، از این رو در این گونه کاربردها باید از سیستم‌های پسارابطه‌ای استفاده کرد.

۶-۴-۱۲ : از نظر قیمت

از حدود ده هزار دلار تا صدهزار دلار و گاه بیشتر. سیستمهای اجراشونده در محیط کامپیوترهای شخصی بین صد تا سه هزار دلار قیمت دارند [ELMA 2000] .

۶-۴-۱۳ : از نظر طرز برپایی

- با محدودیت برپایی یکپارچه
- دارای امکان برپایی گزینشی

۶-۴-۱۴ : از نظر واسط کاربر

- با واسط زبانی
- با واسط غیرزبانی
- با هر دو واسط

B

۶-۴-۱۵ : از نظر رفتار در قبال رویدادها

- سیستم فعال
- سیستم غیرفعال

سیستم فعال سیستمی است که در قبال رویدادهای در پایگاه‌داده‌ها (مثلاً انجام یک عمل بهنگام سازی یا حذف یا پدید آمدن افزونگی و ...) عکس‌العمل مناسب خودکار دارد ،

۶-۴-۱۶ : از نظر متدولوژی زبان پایگاهی

- بدون متدولوژی شیئ‌گرایی
- دارای متدولوژی شیئ‌گرایی

۶-۴-۱۷ : از نظر بهینه‌سازی پرسش ۶-۴-۱۸ : از نظر نوع تراکنش

- دارای بهینه‌ساز متعارف
- پذیرنده تراکنشهای ساده و تک‌سطحی
- دارای بهینه‌ساز مبتنی بر قاعده ، مثلاً ..
- پذیرنده تراکنشهای با مدل پیشرفته (مثلاً تودرتو ، زنجیره‌ای و ...)

۶-۴-۱۹ : از نظر نوع پردازش

- با قابلیت پردازش بی‌درنگ
- فاقد این قابلیت

۶-۴-۲۰ : از نظر رسانه ذخیره‌سازی پایگاه داده‌ها

- با قابلیت ایجاد MMDB
- فاقد این قابلیت

۶-۴-۲۱ : از نظر قابلیت تعامل بین سیستمها

- فاقد این قابلیت
- دارای قابلیت تعامل با سیستمهای همگن
- دارای قابلیت تعامل با سیستمهای ناهمگن

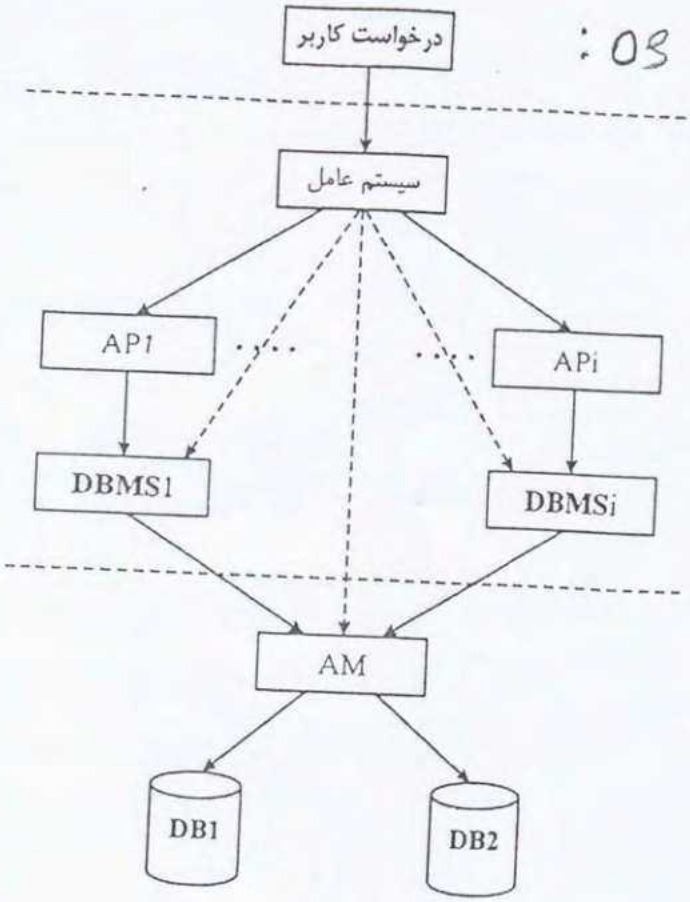
۶-۴-۲۲ : از نظر پردازش داده‌های زمانمند

- فاقد جنبه‌های یک سیستم زمانی (سیستم معمولی)
- سیستم مدیریت پایگاه داده‌های زمانی

۵-۶ :

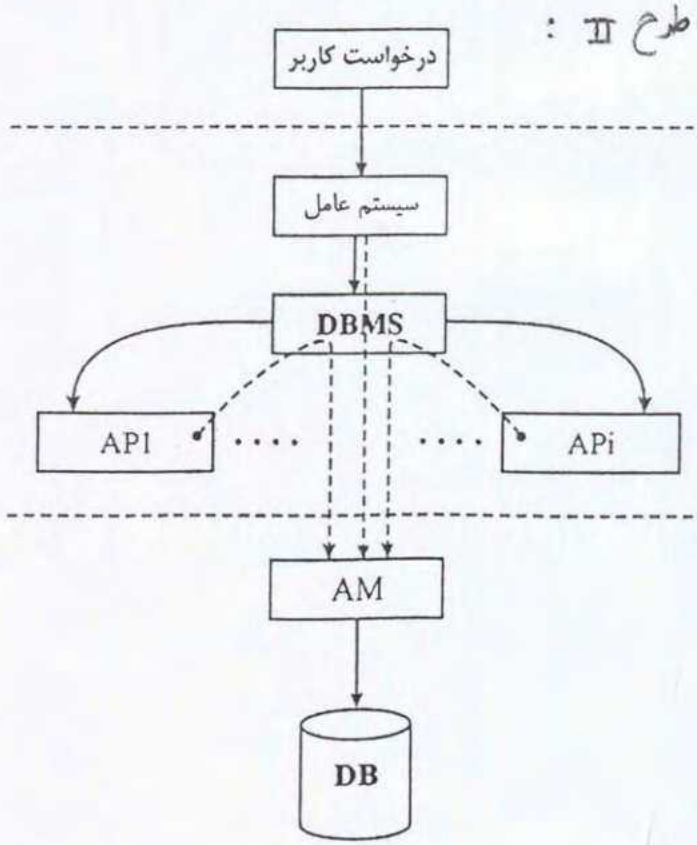
با DBMS ، AP و OS :

طرح I :



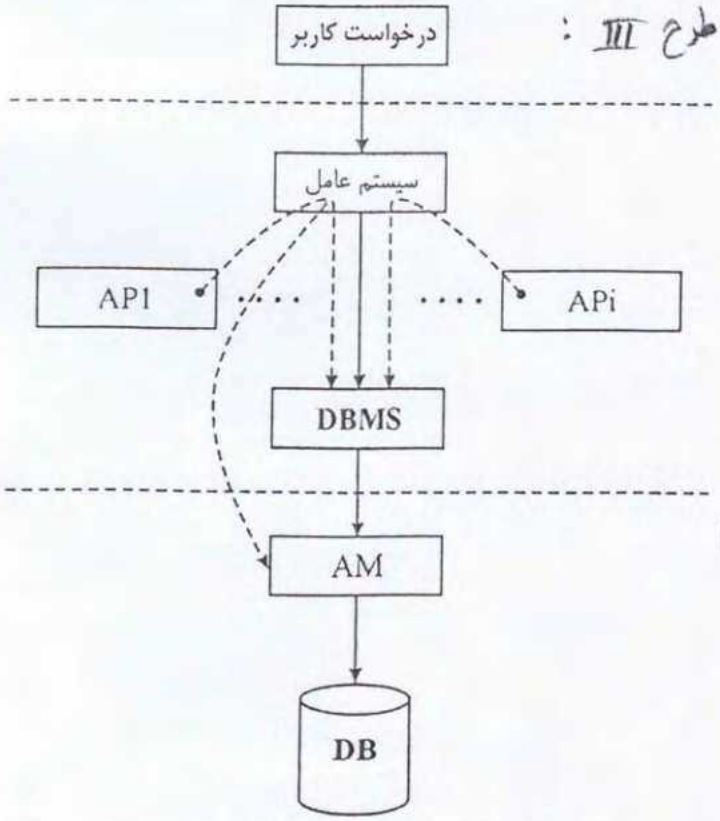
یک DBMS برای هر برنامه کاربردی

طرح II :



یک DBMS برای چند AP با اجرای تحت کنترل DBMS

طرح III :



یک DBMS برای چند AP با اجرای تحت کنترل OS

تمرین : درمورد مزایا و معایب هر یک از این طرح‌ها بحث کنید .

B

۶-۱ کاتالوگ سیستم و دیکشنری داده‌ها : مَتاداده‌ها

حاوی داده‌هایی است در مورد داده‌های ذخیره شده در پایگاه داده‌های کاربر [DATE 95,2000] و این داده‌ها به متاداده‌ها موسومند. به کاتالوگ سیستم گاه دیکشنری داده‌ها هم گفته می‌شود. اما در واقع دیکشنری داده‌ها حاوی اطلاعات بیشتری است. متاداده‌ها معمولاً از دید کاربر سطح خارجی نهان‌اند ، اما مسئول سیستم و یا کاربر مجاز، می‌تواند تا حدی از محتوای کاتالوگ آگاه شود.

دیکشنری داده‌ها معمولاً جزیی از خود سیستم است و به دو صورت فعال و غیرفعال تولید می‌شود. دیکشنری فعال آن است که هر بار که پایگاه داده‌ها مورد دستیابی قرار می‌گیرد، واحدهایی از سیستم ، بسته به نوع درخواست کاربر، آن را وارسی می‌کنند و براساس اطلاعات موجود در آن ، درخواست کاربر نهایتاً انجام می‌شود. اما دیکشنری غیرفعال فقط توسط طراح و کاربران مجاز پایگاه داده‌ها و نیز تیم مدیریت پایگاه داده‌ها استفاده می‌شود تا اطلاعاتی از آن بدست آورند و خود سیستم از آن استفاده نمی‌کند [ELMA 2000] .

ساختار و محتوای کاتالوگ و دیکشنری داده‌ها در سیستم‌های مختلف یکسان نیست ، اما بطور کلی ، اطلاعات زیر در آنها نگهداری می‌شود :

- شماهای خارجی
- شمای ادراکی
- شمای داخلی
- رویه‌های مربوط به تبدیلات بین سه سطح معماری (به قسمت ۶ از همین گفتار مراجعه شود)
- شرح سازمان فیزیکی داده‌های ذخیره‌شده (فایل‌ها ، رسانه‌ها ...)
- مشخصات کاربران و حقوق دستیابی آنها به داده‌های ذخیره‌شده
- مشخصات برنامه‌های کاربردی تولید شده و ارتباط آنها با درخواست‌های کاربران
- مشخصات پایانه‌های متصل به سیستم
- ارتباط بین برنامه‌های کاربردی و داده‌های ذخیره‌شده
- قواعد مربوط به کنترل صحت و دقت داده‌های ذخیره‌شده در پایگاه داده‌ها (موسوم به قواعد جامعیت . به بحث با همین عنوان در گفتار دهم مراجعه شود)
- ضوابط کنترل ایمنی داده‌ها
- مشخصات پیکربندی سخت افزاری سیستم و رسانه‌های ذخیره‌سازی
- اطلاعات متنوع آماری درمورد پایگاه داده‌ها و کاربران
- توابع تعریف شده توسط کاربران

و برخی اطلاعات دیگر ... .

B

* ۷-۶: محورهای اصلی مقایسه DBMS ها

در مقایسه این سیستمها باید تمام پارامترهای مطرح در مبحث " DBMS - شناسی " را در نظر داشت. اما با بررسی این پارامترها می‌توان محورهای اصلی مقایسه را بدست آورد. در زیر محورهای مهمتر را برمی‌شمریم (فرض بر این است که سیستمهای مقایسه‌شونده معماری ANSI را پشتیبانی می‌کنند).

۱. امکانات تعریف داده‌ها
۲. امکانات عملیات در داده‌ها
۳. امکانات کنترل داده‌ها
۴. نوع تراکنش‌ها (چند پرسشی یا برنامه ، تعداد آنها) و طرز مدیریت آنها
۵. امکانات پردازش پرسشها ، بهینه‌سازی آنها و زمان بهینه‌سازی
۶. امکانات ایجاد دیکشنری داده‌ها
۷. وضعیت سطح داخلی - فیزیکی پایگاه داده‌ها از نظر ساختار فایلها ، شیوه‌های دستیابی و جنبه‌های دیگر.
۸. معماری سیستم پایگاهی قابل ایجاد و مدیریت
۹. محیط سخت افزاری و حداقل امکانات لازم
۱۰. محیط سیستم عامل لازم
۱۱. امکانات مدیریت محیط پایگاه داده‌ها :کنترل جامعیت ، ترمیم ، ایمنی و حفاظت.
۱۲. تسهیلات طراحی پایگاه داده‌ها (طراحی منطقی و طراحی فیزیکی).
۱۳. مکانیسم ادغام زبان داده‌ای فرعی در زبان میزبان (وجود پیش کامپایلر یا فراخوانی توابع).
۱۴. تنوع دیدهایی که عملیات ذخیره‌سازی را می‌پذیرند (به گفتار دوازدهم مراجعه شود).
۱۵. مکانیسم مجازشماری کاربران (نامتمرکز یا متمرکز) .
۱۶. وجود مکانیسم حفاظت گذرواژه .
۱۷. امکانات نهان نگاری داده‌ها .
۱۸. طرز انجام عمل بهنگام‌سازی (درجا یا برون ازجا) .
۱۹. الگوریتم اجرای عملگر پیوند و سایر عملگرهای جبر رابطه‌ای (به گفتار دهم مراجعه شود).
۲۰. طرز نمایش نتایج عملیات (گرافیک ، گزارش) .
۲۱. پذیرش یا عدم پذیرش زبانهای نسل چهارم .
۲۲. امکانات تولید برنامه‌های کاربردی .
۲۳. امکانات پشتیبانی تصمیم .
۲۴. امکانات لازم برای واسطهای کاربری .
۲۵. تسهیلات نرم‌افزاری دیگر.

B

۶-۸ : طرز مطالعه سیستم

در مطالعه این نرم افزار، به منظور کسب آشنایی مقدماتی با آن (ونه چندان تخصصی)، باید موارد زیر بررسی شود :

۱. بررسی شرکت سازنده ، خانواده نرم‌افزارهای پایگاهی مشابه و تاریخچه سیستم
۲. تعیین وضع سیستم با توجه به رده‌بندی انجام شده
۳. حداقل پیکره بندی سخت‌افزاری و نرم‌افزاری لازم
۴. بررسی وجود اجزاء معماری ANSI برای پایگاه داده‌ها
۵. امکانات زبان داده‌ای فرعی سیستم
۶. امکانات زبان میزبان سیستم
۷. بررسی مولفه‌های مدل داده‌ای و میزان رابطه‌ای بودن سیستم
۸. اجزاء تشکیل‌دهنده
۹. روند کلی اجرای برنامه توسط سیستم
۱۰. نحوه کار با سیستم : برپاسازی ، راه‌اندازی و آماده‌سازی سیستم ، ورود به سیستم و کارهای لازم برای ایجاد پایگاه و انجام عملیات در آن
۱۱. تسهیلات جانبی سیستم از جمله واسطهای کاربری و ... .
۱۲. سایر جنبه‌های خاص

تأکید : مطالعه سیستم درموارد بالا صرفاً به منظور کسب حداقل شناخت لازم برای کار با این نرم افزار است . بدیهی است در مطالعه چنین سیستمی از دیدگاه تخصصی (DBMS-logy) باید به موارد مهمتری از جمله جنبه‌های مربوط به توانشهای عملیاتی ، کارایی و ... نیز پرداخت.

۶-۹ : رویّه‌های مستند برای کاربران

برای اینکه کاربران و اعضاء تیم مدیریت پایگاه داده‌ها بتوانند از سیستم استفاده کنند ، معمولاً مجموعه‌ای از دستورها و قواعد ، موسوم به رویه‌های مستند [CONN 99] توسط عرضه کنندگان سیستم در اختیار خریداران قرار داده می‌شود. در این رویه‌های مستند چگونگی انجام فعالیتهای زیر مشخص شده است :

۱. برپاسازی سیستم
۲. طرز ارتباط با سیستم
۳. طرز استفاده از سیستم
۴. طرز استفاده از تسهیلات و امکانات آن
۵. تولید نسخه‌هایی از پایگاه داده‌ها
۶. طرز تشخیص عیبهای سخت‌افزاری و نرم‌افزاری و چگونگی رفع آنها و ترمیم پایگاه داده‌ها
۷. تغییر ساختار پایگاه داده‌ها (سازماندهی مجدد)

۸. تنظیم سیستم
۹. بهبود بخشیدن کارایی پایگاه داده‌ها
۱۰. تولید نسخه‌های پشتیبان
و فعالیتهای دیگر از همین قبیل .

۶-۱۰: هزینه‌ها

استفاده از تکنولوژی پایگاه‌داده‌ها هزینه‌هایی دارد. برخی از اقلام مهمتر هزینه عبارتند از:

۱. هزینه خرید نرم‌افزار اصلی (DBMS)
۲. هزینه آموزش نرم‌افزار اصلی
۳. هزینه نگهداری و بهره‌برداری از آن
۴. هزینه تبدیل سیستم ناپایگاهی به سیستم پایگاهی
۵. هزینه تهیه ابزارهای نرم‌افزاری دیگر
۶. هزینه آموزش امکانات نرم‌افزاری
۷. هزینه تهیه بسته‌های کاربردی
۸. هزینه آموزش بسته‌های کاربردی
۹. هزینه تهیه مستندات خود سیستم
۱۰. هزینه تهیه مستندات امکانات نرم‌افزاری و بسته‌های کاربردی
۱۱. هزینه تنظیم مستندات سیستم پایگاه داده‌ها
۱۲. هزینه تهیه سخت‌افزار پردازشگر (کامپیوتر از رده‌های مختلف)
۱۳. هزینه تهیه سخت‌افزار ذخیره‌سازی
۱۴. هزینه تأمین شبکه‌های لازم
۱۵. هزینه نگهداری و بهره‌برداری از سیستم کاربردی
۱۶. هزینه بهینه‌سازی و گسترش سیستم
۱۷. حقوق و مزایای افراد تیم مدیریت و تیم‌های اجرایی

B

* ۶ - ۱۱: تسهیلات نرم افزاری کمکی DBMS

این تسهیلات و جنبه‌ها می‌توانند همراه خود سیستم و یا به نحوی قابل تأمین و استفاده در محیط سیستم باشند (به صورت نرم‌افزار جداگانه به کاربران سیستم عرضه شوند). هرچه سیستم از نظر پذیرش انواع تسهیلات و سازگاری و همایندی با آنها غنی‌تر باشد، مطلوبتر است و بهره‌برداری بهتری از سیستم امکان‌پذیرخواهد بود. برخی ازاین تسهیلات (ابزارها) عبارتند از :

۱. امکانات تولید نسخه‌های پشتیبان و میزان سهولت تولید آنها
۲. امکانات پرسش به کمک مثال و پرسش به کمک فرم
۳. امکانات بررسی‌ها و تحلیلهای آماری
۴. امکانات گرافیکی
۵. پردازشگر زبان طبیعی
۶. مولد گزارش
۷. انواع ویرایشگرها
۸. پردازشگر متن
۹. کاربریار
۱۰. ابزارهای ایجاد برنامه‌های کاربردی
۱۱. ابزارهای ایجاد انواع واسطهای کاربردی
۱۲. امکانات دستیابی به داده‌های دوردست
۱۳. امکانات تولید خروجی‌های کاربرپسند
۱۴. امکانات تنظیم کردن پایگاه
۱۵. امکانات تنظیم کردن خود سیستم
۱۶. امکانات بارگذاری ، بازبارگذاری ، خالی کردن پایگاه
۱۷. امکانات تبدیل یک ساختار داده از سطح خارجی معماری ANSI به ساختار داده دیگر از سطح ادراکی همان معماری
۱۸. امکانات گستر برگ
۱۹. امکانات آغازاندن پایگاه
۲۰. امکانات تهیه آمارهای مربوط به عملیات ذخیره‌سازی (درج ، حذف ، بهنگام‌سازی)
۲۱. امکانات سیستم در ایجاد پایگاه داده‌های چندرسانه‌ای (از نظر رسانه‌های تماس انسان با ماشین از جمله متن ، صدا ، تصویر و کلام)
۲۲. امکانات سیستم برای مدلسازی داده‌ها و طراحی منطقی و فیزیکی پایگاه
۲۳. امکانات سیستم برای گشت‌زنی (گذارگری) در پایگاه‌داده‌ها
۲۴. تسهیلات برپاسازی سیستم و به ویژه برپاسازی انتخابی
۲۵. امکان استفاده از زبانهای سطح بالا (نه به عنوان زبان میزبان) و مکانیسم پیوند به سیستم از طریق آنها
۲۶. امکانات فارسی‌پردازی درخود سیستم (در سطح واسط کاربری و در سطوح درونی‌تر

۲۷. امکان ایجاد مدولهای اجرایی برای برنامه‌های کاربردی در محیط سیستم به نحوی که خارج از محیط سیستم قابل اجرا باشند

۲۸. امکان استفاده از یک سیستم خبره در محیط سیستم

۲۹. مولد فرم

۳۰. مولد مِنو

۳۱. امکانات شبکه‌ای

۳۲. ابزار تولید واسط دیداری- شنوداری

۳۳. واسط 4GL

۳۴. ابزارهای تولید مستندات (در مراحل مختلف تولید یک سیستم کاربردی)

۳۵. امکانات نگهداری سیستم

۳۶. امکانات بهینه‌سازی برنامه‌های کاربردی

۳۷. امکانات یادگیری طرز کار با سیستم و بهره‌برداری آن

۳۸. انواع میان‌افزار ها (گاه موسوم به افزارگان پشت صحنه[۳])

۳۹. امکانات کار در محیط وب

۴۰. امکانات مدیریت پویای پرسشها

آنچه برشمرده شد ، فهرستی است از تسهیلات و جنبه‌ها ، و با توجه به پیشرفت مهندسی نرم‌افزار، امکانات دیگری نیز می‌توانند مطرح باشند. به‌ویژه که هر روز انواع گوناگونی از این قبیل ابزارها و تسهیلات تولید و به بازار مصرف عرضه می‌شوند. سیستم نه تنها باید بتواند با این امکانات تماس برقرار کند بلکه باید به سهولت با آنها همایندی داشته باشد.

۶-۱۲: مشخصات کلی سیستم

۱. نام نرم‌افزار ۲. عنوان شرکت سازنده و شرکت فروشنده

۳. شماره ویراست (نگارش) مورد بررسی ـ شماره آخرین ویراست

۴. تاریخ عرضه (اولین نگارش و آخرین نگارش)

۵. قیمت ۶. نام کشور سازنده و فروشنده

۷. شرایط کلی تحویل ۸. خدمات بعد از تحویل

۹.کمیت و کیفیت آموزش ۱۰. مستندات

۱۱. تعداد مشتری‌ها و ماهیت نیازهای اطلاعاتی و پردازشی آنان

۱۲. ضمانت(های) شرکت فروشنده

۱۳. امکانات شرکت فروشنده در گسترش یا ارتقاء سیستم

۱۴. در دسترس بودن فروشنده (هرگاه که لازم باشد)

۱۵. وجود تیم فنی پشتیبانی سیستم ، در شرکت فروشنده

۶-۱۳ : مزایا و معایب مشی پایگاهی

B

مزایای این تکنولوژی بستگی به نوع سیستم (DBMS) و معماری سیستم پایگاه داده‌ها و ماهیت کاربردها دارد. در این گفتار ابتدا مزایا و معایب پایگاه داده‌هایی که در محیط کامپیوتر شخصی و معمولاً با یک سیستم تک‌کاربری ایجاد می‌شود ، را بررسی می‌کنیم و سپس به سیستم جامع پایگاه داده‌های چندکاربری می‌پردازیم.

توجه : در این بحث ، وجود مدلسازی اصولی و جامع و نیز طراحی بهینه پایگاه داده‌ها فرض است. روشن است که در صورت مدلسازی و طراحی غیر اصولی و نادرست ، از قویترین نرم‌افزارها هم نمی‌توان بهره‌برداری بهینه کرد.

۶-۱۳-۱ : مزایا و معایب "سیستم" تک کاربری

توجه ! هرگاه تعدادی از این سیستم‌ها ، بویژه بی‌ارتباط با یکدیگر ، در سازمان وجود داشته باشند : "مشی بنیادین"

۶-۱۳-۱-۱ : مزایا

۱. هر بخش از سازمان ، داده‌های خود را نگهداری و پردازش می‌کند و در نتیجه سیستم کامپیوتری مرکزی سازمان می‌تواند به کارهای دیگر بپردازد.

۲. با استفاده از کامپیوترهای شخصی، حجم داده‌های سیستم مرکزی کاهش می‌یابد . در غیر اینصورت با افزایش حجم داده‌ها ، سیستم مرکزی را باید متناسباً گسترش داد.

۳. پایگاه داده‌هایی که روی کامپیوتر شخصی ایجاد می‌شود معمولاً کوچک و مدلسازی ، طراحی و پیاده‌سازی آن ساده است و کاربر می‌تواند پایگاه داده‌ها را چنان ایجاد کند که با نیازهایش دقیقاً تطبیق داشته باشد.

۴. کار با سیستمهای تک کاربری و برنامه‌سازی در محیط آنها ساده است و نیاز به تخصص چندان ندارد.

۵. با پیشرفت روزافزون قابلیت و قدرت کامپیوترهای شخصی ، این سیستم‌ها می‌توانند بسیاری از کارهای سیستم‌های کامپیوتری بزرگ را انجام دهند.

۶-۱۳-۱-۲ : معایب

۱. وجود تعدادی سیستم پایگاه داده‌های کوچک و پراکنده در یک سازمان منجر به بروز افزونگی ، ناسازگاری داده‌ها و تا حدی ناایمنی آنها می‌شود. (یعنی همان مشکلی که در مشی ناپایگاهی (فایلینگ) وجود دارد. به کنجکاوی ۷ از گفتار دوم توجه شود).

۲. محدودیت‌های سخت‌افزاری سبب محدود شدن اندازه فایل‌ها می‌شود و نیز محدودیت در سرعت پردازش ، باعث محدود شدن حجم پایگاه داده‌ها می‌شود.

۳. خود سیستم نمی‌تواند قوی و کارا باشد (زیرا سیستم قوی نیاز به سخت‌افزار کافی و سیستم عامل قوی دارد). در نتیجه بسیاری از خدماتی که یک سیستم قوی می‌تواند ارائه کند ، ارائه نمی‌شود.

۴. چون این سیستم‌های کوچک معمولاً تک‌کاربری هستند، میزان ایمنی و حفاظت داده‌ها و مکانیسم مجازشماری آنها ضعیف است.

۵. امکانات تولید نسخه‌های پشتیبان و ترمیم پایگاه داده‌ها معمولاً کم است.
۶. اشتراکی کردن این پایگاه‌های جداگانه و نامرتبط ، مشکلات تکنیکی جدی دارد.
۷. اعمال مجموعه واحدی از استانده‌ها در کل سازمان ناممکن است.
۸. معمولاً یک کاربر محیط شخصی مهارت کافی در مدلسازی و طراحی بهینه پایگاه داده‌ها ندارد و درنتیجه "سیستم کاربردی" تولید شده ،کارایی مطلوبی نداشته ، فاقد انعطاف‌پذیری است.

۶-۱۳-۲ : مزایا و معایب پایگاه داده‌های چندکاربری

مؤلفین منابع کلاسیک پایگاه داده‌ها ، مزایایی برای این نوع پایگاه داده‌ها برشمرده‌اند و نیز معایبی چند . آنچه در زیر می‌آید ، حاصل بررسی این متون است (در هر منبع چند مورد از موارد زیر ، حداکثر شاید ده مورد ، قید شده است).

۶-۱۳-۲-۱ : مزایا

۱. اشتراک داده‌ها
۲. کاهش حتی الامکان افزونگی
۳. تعدد شیوه‌های دستیابی و تسهیل دستیابی به داده‌ها
۴. اجتناب از ناسازگاری داده‌ای
۵. تامین سیستم کاراتر برای ذخیره‌سازی داده‌ها و دستیابی به آنها (از نظر زمان و حافظه و با توجه به خدماتی که سیستم ارائه می‌کند)
۶. تامین همروندی بهتر
۷. تسهیل پردازش تراکنشها
۸. تضمین جامعیت داده‌ها
۹. امکان اعمال ضوابط ایمنی دقیقتر
۱۰. ترمیم داده‌ها به طور کاراتر
۱۱. تامین استقلال داده‌ای (به بحث استقلال داده‌ای در همین گفتار رجوع شود)
۱۲. حفظ محرمانگی داده‌ها
۱۳. امکان اعمال استانده‌ها
۱۴. ایجاد سازش بین نیازهای گاه حتی متضاد کاربران
۱۵. امکان برنامه‌سازی با زبانهای نارویه‌ای
۱۶. تسهیل گسترش " کاربردها " و رشد محیط ذخیره‌سازی (به بحث استقلال داده‌ای در همین گفتار مراجعه شود).
۱۷. "تسریع" در دریافت پاسخ پرسشها
۱۸. تسهیل دریافت انواع گزارشات آماری (کم کردن حجم برنامه‌سازی برای تولید این گزارشات و زمان آن)

۱۹.وضوح بخشیدن به دید کاربران (مخصوصاً در پایگاه داده‌های رابطه‌ای (جدولی))
۲۰.تسهیل در ایجاد تغییرات (رجوع شود به بحث استقلال داده‌ای) برای هماهنگی با نیازهای جدید
۲۱.تعدد زبان‌ها
۲۲.کاهش حجم برنامه‌ها
۲۳.تنوع کاربران
۲۴.تسهیل سازماندهی مجدد سطح ادراکی یا تغییر طراحی منطقی (به بحث استقلال داده‌ای رجوع شود)
۲۵.ارائه تسهیلات سازماندهی مجدد محیط ذخیره‌سازی
۲۶.تسهیل مدیریت داده‌های سازمان
۲۷.استفاده بهتر از سخت‌افزار
۲۸-کاهش هزینه‌های سازمان در میان مدت و درازمدت (از طریق کاهش حجم برنامه‌سازی ، کاهش هزینه تولید نرم‌افزار و استفاده بهتر از سخت‌افزار و نرم‌افزار)

کنجکاوی ۱ : کدامیک از این مزایا ، هدفهای اصلی تکنولوژی پایگاه داده‌ها هستند؟

موضوع مطالعه بیشتر ۱ : درمورد هریک از این مزایا بیشتر مطالعه شود. (برای توضیح درباره برخی از این مزایا از جمله به [DATE 2000] ، [CONN 99] ، [ELMA 2000] و [روحا ۷۸- الف] می‌توان مراجعه کرد).

کنجکاوی ۲ : آیا مزایای دیگری هم وجود دارد؟

تأکید: همه این مزایا لزوماً در عمل توسط سیستمهای موجود تامین نمی‌شوند ، و فراموش نکنیم که داشتن دانش در مفاهیم بنیادی پایگاه داده‌ها و شناخت کافی از این تکنولوژی و به ویژه از ویژگی‌ها و توانش‌های سیستم انتخاب شده و استفاده آگاهانه و تخصصی از این تکنولوژی ، در تامین مزایای بر شمرده نقش اساسی دارد.

۶-۱-۲-۲ : معایب

۱. هزینه بالای نرم‌افزار
۲. هزینه بالای سخت‌افزار
۳. هزینه بیشتر برای برنامه‌سازی
۴. هزینه بالا برای انجام مهندسی دوباره به منظور تبدیل سیستم از مشی ناپایگاهی به مشی پایگاهی ،
۵. کند شدن اجرای بعضی از برنامه‌های کاربردی

کنجکاوی ۳ : چرا؟

۶. خطر بالقوه آسیب پذیری داده‌ها
۷. تاثیرات گسترده‌تر خرابی‌ها و دشواری بیشتر ترمیم آنها
۸. پیچیده بودن سیستم و نیاز به تخصص بیشتر

B

کنجکاوی ۴ : آیا می‌توانید معایب دیگری را بر شمرید؟

نکته ۱ : درست است که استفاده از این تکنولوژی در سازمان ، هزینه‌هایی دربردارد اما در میان مدت و دراز مدت ، هزینه‌های داده‌داری و داده‌پردازی سازمان کاهش می‌یابد (به شرط استفاده بجا و کارشناسانه از این تکنولوژی).

۶-۱۴: شرایط استفاده از تکنولوژی پایگاه داده‌ها

از فهرست مزایای بر شمرده برای تکنولوژی پایگاه داده‌ها می‌توان استنتاج کرد که بطور کلی در چه شرایطی ، از این تکنولوژی استفاده می‌شود . اما از دیدگاه تکنیکی و کاربردی ، چنانچه شرایط زیر برقرار باشد، بکارگیری این تکنولوژی در سازمان توصیه می‌شود :

۱. ایجاد یک سیستم یکپارچه اطلاعاتی در سازمان مورد نظر باشد (توجه داریم که ایجاد سیستم یکپارچه لزوماً به معنای وجود یک پایگاه داده‌های متمرکز در سازمان نیست . به گفتار نهم ، معماری سیستم پایگاه داده‌ها مراجعه شود) .
۲. حجم داده‌های سازمان زیاد بوده و مرتباً بطور پویا رشد یابند .
۳. تغییرات مداوم در داده‌های ذخیره شده زیاد باشد و استقلال داده‌ای (جدایی برنامه‌های کاربردی از داده‌ها) مورد نظر باشد .
۴. بسامد در خواستهای موردی کاربران بالا باشد .
۵. نیاز به اِعمال کنترل متمرکز و دقیق روی کل داده‌های سازمان و تامین ایمنی بالا .
۶. وجود ارتباطات پیچیده بین داده‌های سازمان .
۷. میزان داده‌های مشترک بین برنامه‌های کاربردی زیاد باشد .
۸. صحت ، دقت و سازگاری داده‌ها (جامعیت داده‌ها (به گفتار دهم مراجعه شود)) بطور پیوسته مورد نظر باشد .
۹. زیاد بودن گزارشها و گوناگونی آنها .
۱۰. نیاز به انجام پردازشهای تحلیلی بر خط .
۱۱. نیاز به سیستم داده‌کاوی و کشف دانش در سازمان

۷- درباره DBA (و ریز فعالیتهای طراحی و پیاده‌سازی پایگاه داده‌ها) B

۷-۱: تعریف

مدیر پایگاه داده‌ها فردی است متخصص در پایگاه داده‌ها و با مسئولیت علمی- فنی و نیز اداری در محدوده وظایفی که عهده‌دار است . این مدیر معمولاً همراه با یک تیم تخصصی کار می‌کند که به آن تیم مدیریت پایگاه داده‌ها می‌گویند. هریک از اعضا این تیم مسئولیت خاصی دارد و در حیطه اختیارات و وظایفش می‌تواند سرپرست یک تیم اجرایی باشد.

۷-۲: مسئولیت‌ها

برخی از مسئولیت‌های اصلی در این تیم تخصصی عبارتند از :

- مدیر پایگاه داده‌ها
- مدیر داده ها
- مدیر امور پژوهش- توسعه (درهمین حیطه مورد بحث یعنی دانش و تکنولوژی پایگاه داده‌ها)
- مدیر سیستم(های) کاربردی
- مسئول تیمهای برنامه‌سازی
- مسئول کنترل کارایی DBMS
- مسئول کنترل کارایی خود سیستم پایگاه داده‌ها
- مسئول نظارت بر عملیات روی پایگاه داده‌ها و انجام فعالیت‌های آماری مربوطه
- مسئول تماس با کاربران زیر محیط‌های سازمان
- مسئول تنظیم مستندات و وضع استانده‌ها

نکته ۱ : در یک محیط کاری برخوردار از سطح مطلوب دانش و تکنولوژی و عمل کننده براساس دیسیپلین‌های علمی و مهندسی و دارای مدیریت پویا ، وجود این تیم تخصصی اجتناب ناپذیر است. بعلاوه این تیم باید از مشاورانی در زمینه‌های دیگر مهندسی نرم‌افزار و مهندسی سخت‌افزار استفاده کند و حتی مطلوب این است که بعضی از مشاوران به نحوی عضو خود تیم باشند.

نکته ۲ : مدیر داده‌ها فردی است با دانش و تجربه در مدیریت و آشنا با دانش و تکنولوژی پایگاه داده‌ها . برای اطلاع از وظایف مدیر داده‌ها به منابع درس‌های "مهندسی نرم‌افزار" ، " تحلیل و طراحی سیستم‌ها " و نیز درس "سیستم اطلاعات مدیریت" مراجعه شود.

نکته ۳ : در اینجا فقط اشاره می‌کنیم که در مدیریت نوین سازمانها ، هر سازمان دارای پنج سرمایه است :

- سخت‌افزار
- نرم‌افزار
- داده
- بودجه
- تخصص

بنابراین داده هم از سرمایه‌های مهم است و چنین سرمایه‌ای نیاز به مدیریت پویا و کارا دارد.

کنکاوی : به نظر شما کدامیک مهم‌تر است ؟ چرا ؟ ترتیب نزولی اهمیت این سرمایه‌ها چگونه است

B

۷-۳ : وظایف DBA

۱. مشارکت در تفهیم اهمیت و نقش داده به مدیریت سازمان (با همکاری مدیر داده‌ها) .
۲. مشارکت در تفهیم اهمیت و مزایای تکنولوژی پایگاه داده‌ها .
۳. مشارکت در تصمیم‌گیری درمورد استفاده یا عدم استفاده از این تکنولوژی .
۴. مشارکت در توجیه علمی - فنی تصمیم استفاده از این تکنولوژی .
۵. مطالعه دقیق و همه جانبه محیط عملیاتی و برآورد خواسته‌ها و برآورد نیازهای کاربران (انجام اصولی مهندسی نیازها) .
۶. بررسی روند داده‌ها و روند رویدادها در محیط و رسم نمودار روند داده‌ها و روند رویدادها (یک یا هر دو بسته به شیوه مدلسازی سیستم مورد نظر) و تهیه و تنظیم مستندات لازم .
۷. مدلسازی معنایی داده‌ها (با مراحلی که دیده شد از جمله رسم نمودار EER) .
۸. تخمین حجم داده‌های ذخیره شدنی در پایگاه داده‌ها .
۹. تصمیم‌گیری درمورد تعیین معماری سیستم پایگاه داده‌ها (رجوع شود به گفتار نهم) و تعیین مشخصات سیستم(های) کاربردی مورد نیاز .
۱۰. مشارکت در انتخاب DBMS(ها) و پیکربندی سخت‌افزاری و نرم‌افزاری لازم درصورت لزوم (اگر انتخاب نشده باشد) .
۱۱. تصمیم‌گیری در انتخاب و انتساب اعضاء تیمهای اجرایی .
۱۲. تصمیم‌گیری در انتخاب ابزارهای نرم‌افزاری دیگر لازم برای تولید و گسترش سیستم مورد نظر .
۱۳. تصمیم‌گیری درمورد زبان(های) برنامه‌سازی مورد نیاز و متناسب با هر کاربرد .
۱۴. طراحی سطح ادراکی پایگاه داده‌ها (طراحی منطقی) .
۱۵. نوشتن شمای ادراکی و برنامه‌های لازم برای ایجاد پایگاه داده‌ها .
۱۶. تعیین مجموعه قواعد (محدودیتهای) جامعیت ناظر به پایگاه داده‌ها .
۱۷. نظارت بر تعیین دیدهای خارجی و نوشتن شماهای خارجی .
۱۸. تصمیم گیری درمورد مشخصات ساختار سطح داخلی پایگاه داده‌ها و تعیین ساختار فایل‌های مناسب ، استراتژی‌های دستیابی کارا و نوشتن شمای داخلی (طراحی فیزیکی) .

۱۹. انجام طراحی توزیع (در صورت معماری توزیع شده)
۲۰. طراحی "برنامه‌های کاربردی" ، تراکنش‌های لازم و رویه‌های عملیاتی لازم . (توجه داشته باشیم که هر برنامه‌ای ، نیاز به طراحی دارد و تنها پس ازطراحی اصولی برنامه ، می‌توان برنامه‌سازی کرد) ، و ایجاد ارتباط دائم با تولید کنندگان "سیستم کاربردی" .
۲۱. طراحی واسطهای کاربری .
۲۲. ایجاد نمونه نخست سیستم پایگاهی (نمونه‌سازی) و بارگذاری پایگاه با داده‌های تستی .
۲۳.نوشتن برنامه‌های لازم برای کنترل پایگاه داده‌ها بویژه اعمال محدودیتهای جامعیتی .
۲۴.نوشتن برنامه‌های لازم برای بهره برداری از پایگاه داده‌ها .
۲۵.ایجاد سیستم پایگاهی واقعی (و منطبق با نیازهای کاربران)

۲۶.نظارت بر وارد کردن داده‌ها (درحجم محدود) .

۲۷.انتخاب استراتژیهای تست مناسب و تست کردن "سیستم" با داده‌های تستی و نیز با داده‌های واقعی در حجم محدود (انجام دو مرحله تست) .

۲۸.نظارت بر وارد کردن داده‌های واقعی سازمان .

۲۹.تست کردن "سیستم" با داده‌های واقعی و در حجم واقعی (انجام تست مرحله سوم) .

۳۰.تنظیم دقیق قسمتهای مختلف سیستم و کل سیستم یکپارچه .

۳۱.تعیین ضوابط دستیابی کاربران به داده‌ها .

۳۲.نظارت در تهیه مستندات لازم در همه مراحل کار.

۳۳.وضع استانده‌های لازم در همه مراحل کار و نظارت بر اعمال آنها .

۳۴.تصمیم گیری درمورد چگونگی ترمیم پایگاه داده‌ها و درصورت لزوم تهیه یا توسعه ابزارهای این کار و انجام ترمیم پایگاه .

۳۵.کنترل مداوم کارایی DBMS وکارایی "سیستم پایگاه داده‌ها" وتلاش درافزایش کارایی .

۳۶.نظارت و کنترل دائم بر عملیاتی که در پایگاه داده‌ها انجام می‌شود .

۳۷.کنترل جامعیت پایگاه داده‌ها .

۳۸.تضمین محرمانگی داده‌ها .

۳۹. تصمیم گیری درمورد چندی و چگونگی رشد (گسترش) پایگاه داده‌ها .

۴۰. اتخاذ تدابیر لازم برای ایمنی و حفاظت داده‌ها و اعمال این تدابیر .

۴۱. مدیریت کاربران پایانی (ایجاد و تعریف کاربران ، گذرواژه‌ها ، امتیازها و ...) .

۴۲. تماس دائم با کاربران و شناخت نیازهای جدید آنها .

۴۳. تولید نسخه‌های پشتیبان بطور متناوب (با تناوب مناسب) .

۴۴. تعیین الگوهای استفاده از داده‌ها و بسامد (فرکانس) استفاده از داده‌ها .

۴۵. تصمیم‌گیری درمورد چگونگی سازماندهی مجدد پایگاه داده‌ها .

۴۶. انجام تبدیل و انتقال داده‌ها از "سیستم‌های موجود" به پایگاه داده‌های جدید و انجام تبدیل برنامه‌های کاربردی موجود به گونه‌ای که قابل اجرا در "سیستم کاربردی" جدید باشند .

۴۷. تلاش در جهت ارتقاء سطح دانش و فن اعضاء تیم و کاربران (بویژه در زمینه تکنولوژی اطلاعات و سیستمهای اطلاعاتی) .

۴۸. تلاش در جهت شناسایی امکانات جدید ، گسترش ، ارتقاء و کاراتر کردن سیستم با استفاده از این امکانات .

۴۹. تهیه و تنظیم انواع آمارها و گزارشات کنترلی و مدیریتی درمورد سیستم پایگاه داده‌ها وکاربران .

۵۰. تضمین انجام و اتمام "پروژه پایگاهی" در مدت زمان پیش‌بینی شده و با توجه به محدودیت بودجه .

توجه : با دقت درفهرست وظایف DBA ، می‌توان تفصیل کارهای لازم برای طراحی و ایجاد یک

۸- کاربر :

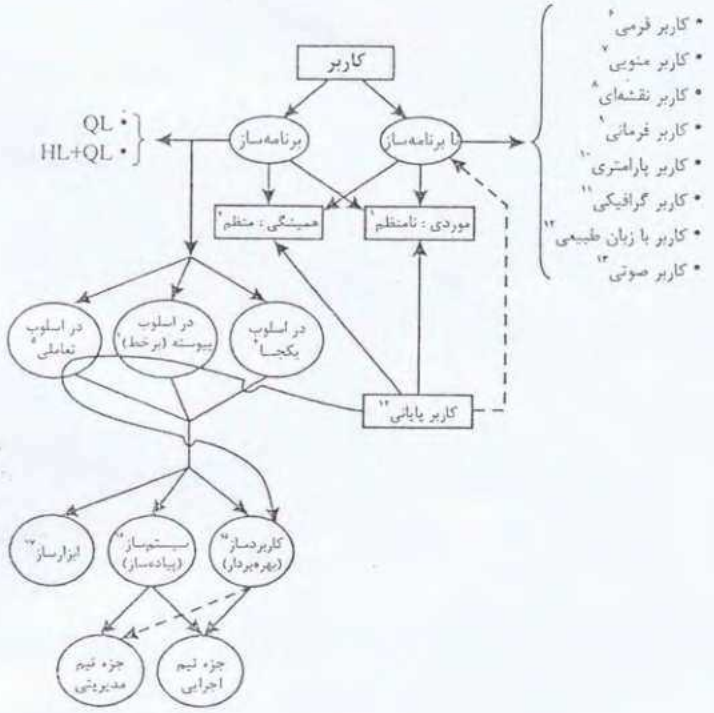
در معنای عام ، هر استفاده‌کننده از پایگاه داده‌ها را کاربر گوییم . کاربر رده‌هایی دارد که در نمودار شکل زیر نشان داده شده‌اند .

برای اطلاع بیشتر درموردکاربر، به منابع ذیربط (از جمله به [روحا ۷۸- الف]) مراجعه شود .

نکته : برنامه‌ساز "سیستم" ، برنامه‌های ایجاد[2] و کنترل پایگاه داده‌ها را می‌نویسد . اما برنامه‌ساز "کاربردی" معمولاً برنامه‌های بهره‌برداری از پایگاه داده‌ها را می‌نویسد .

توجه : اصطلاح " کاربرپایانی" ، در معنای عام ، به هر دوگونه کاربر نابرنامه‌ساز (که از طریق یک واسط غیرزبانی از "سیستم" استفاده می‌کند) و کاربر برنامه‌ساز کاربردی گفته می‌شود ، اما معمولاً برنامه‌ساز بهره‌بردار (کاربرد ساز) است : هر کاربری که "کارش[3]" نیاز به دستیابی به پایگاه داده‌ها به منظور بازیابی یا ذخیره‌سازی دارد و گزارشهایی تولید می‌کند (و نیز رجوع شود به [ELMA 2003]) .

1- Developement
2- Job



1- Ad hoc (Unplanned)
2- Permanent (planned)
3- Batch mode
4- Online mode
5- Interactive mode
6- Form driven
7- Menu driven
8- Map driven
9- Command driven
10- Parameter driven
11- Graphic driven
12- Natural language driven
13- Voice driven
14- End user
15- Application developer
16- System developer
17- Tool developer

کنجکاوی : کاربر موردی (یکباری) ؟

تمرین : مدلسازی معنایی این محیط را انجام دهید .

B

1

به نام آن که جان را فکرت آموخت

یادداشت‌های تکمیلی

(سری II)

۱- تفاوت‌های رابطه با جدول

۲-رابطه‌ی درجه‌ی صفر

۳-درباره‌ی طراحی

۳-۱ مراحل اساسی

۳-۲ خصوصیات طراحی خوب

۴-مثال‌هایی از قواعد جامعیت

۵- جبر رابطه‌ای

۵-۱ خواص عملگرها

۵-۲ مجموعه‌ی کامل عملگرها

۵-۳ مثال‌ها

۵-۴ مثال‌های دیگر

۶-حساب رابطه‌ای

۶-۱ روابط هم‌ارزی

۶-۲ مثال‌ها

۶-۳ مثال‌های دیگر

۷- تمرینات تکمیلی

۷-۱ صورت پرسش‌ها

۷-۲ پاسخ در جبر رابطه‌ای

۷-۳ پاسخ در حساب رابطه‌ای

۸- درباره‌ی تئوری وابستگی

۹- دلایل بروز افزونگی

در باره رابطه

Relations *vs.* Tables

۱- تفاوتهای رابطه با جدول

For purposes of reference, we present in this subsection a list of some of the main differences between (a) the formal object that is a relation as such and (b) the informal object that is a table, which is an informal picture on paper of that formal object:

1. Each attribute in the heading of a relation involves a type name, but those type names are usually omitted from tabular pictures.
2. Each component of each tuple in the body of a relation involves a type name and an attribute name, but those type and attribute names are usually omitted from tabular pictures.
3. Each attribute value in each tuple in the body of a relation is a value of the applicable type, but those values are usually shown in some abbreviated form—for example, S1 instead of S#('S1')—in tabular pictures.
4. The columns of a table have a left-to-right ordering, but the attributes of a relation do not. *Note:* One implication of this point is that columns might have duplicate names, or even no names at all. For example, consider this SQL query:

   ```
   SELECT S.CITY, S.STATUS * 2, P.CITY
   FROM   S, P ;
   ```

   What are the column names in the result of this query?
5. The rows of a table have a top-to-bottom ordering, but the tuples of a relation do not.
6. A table might contain duplicate rows, but a relation does not contain duplicate tuples.

The foregoing is not an exhaustive list of the differences. Others include:

- The fact that tables are usually regarded as having at least one column, while relations are not required to have at least one attribute (see the subsection "Relations with No Attributes" later in this section)
- The fact that tables—at least in SQL—are allowed to include nulls, while relations are certainly not (see Chapter 19)
- The fact that tables are "flat" or two-dimensional, while relations are *n*-dimensional (see Chapter 22)

It follows from all of the foregoing that, in order to agree that a tabular picture can properly be regarded as representing a relation, *we have to agree on how to "read" such a picture;* in other words, we have to agree on certain **rules of interpretation** for such pictures. To be specific, we have to agree that there is an underlying type for each column; that each attribute value is a value of the relevant type; that row and column orderings are irrelevant; and that duplicate rows are not allowed. If we can agree on all of these rules of interpretation, then—*and only then*—we can agree that a table is a reasonable picture of a relation.

So we can now see that a table and a relation are indeed not quite the same thing (even though it is often convenient to pretend they are). Rather, a **relation** is what the definition says it is, a rather abstract kind of object, and a **table** is a concrete picture, typically on

DATE 03 : منبع *

paper, of such an abstract object. They are not (to repeat) quite the same. Of course, they are very similar . . . and in informal contexts, at least, it is usual, and perfectly acceptable, to say they are the same. But when we are trying to be precise—and right now we *are* trying to be precise—then we do have to recognize that the two concepts are not exactly identical.

That said, it is worth pointing out too that in fact it is a major advantage of the relational model that its basic abstract object, the relation, does have such a simple representation on paper. It is that simple representation that makes relational systems easy to use and easy to understand, and makes it easy to reason about the way relational systems behave. Nevertheless, it is unfortunately also the case that that simple representation does suggest some things that are not true (e.g., that there is a top-to-bottom tuple ordering).

Relations with No Attributes

۲ : رابطه درجه صفر

Every relation has a set of attributes; and, since the empty set is certainly a set, it follows that it must be possible for a relation to have the empty set of attributes, or in other words no attributes at all. (Do not be confused: We often talk about "empty relations," meaning relations whose body is an empty set of tuples, but here, by contrast, we are talking about relations whose *heading* is an empty set of *attributes*.) Thus, relations with no attributes are at least respectable from a mathematical point of view. What is perhaps more surprising is that they turn out to be extremely important from a practical point of view as well!

In order to examine this notion more closely, we first need to consider the question of whether a relation with no attributes can contain any tuples. The answer (again perhaps surprisingly) is *yes*. To be more specific, such a relation can contain *at most one* tuple:

namely, the 0-tuple (i.e., the tuple with no components; it cannot contain more than one such tuple, because all 0-tuples are duplicates of one another). There are thus precisely two relations of degree zero—one that contains just one tuple, and one that contains no tuples at all. So important are these two relations that, following Darwen [6.5], we have pet names for them: We call the first TABLE_DEE and the other TABLE_DUM, or DEE and DUM for short (DEE is the one with one tuple, DUM is the empty one). *Note:* It is hard to draw pictures of these relations! Thinking of relations as conventional tables breaks down, somewhat, in the case of DEE and DUM.

Why are DEE and DUM so important? There are several more or less interrelated answers to this question. One is that they play a role in the relational algebra—see Chapter 7—that is akin, somewhat, to the role played by the empty set in set theory or zero in ordinary arithmetic. Another has to do with what the relations mean (see reference [6.5]): essentially, DEE means TRUE, or *yes*, and DUM means FALSE, or *no*. In other words, they have *the most fundamental meanings of all*. (A good way to remember which is which is that the "E"s in DEE match the "e" in *yes*.)

In **Tutorial D**, the expressions TABLE_DEE and TABLE_DUM can be used as shorthand for the relation selector invocations

```
RELATION { } { TUPLE { } }
```

and

```
RELATION { } { }
```

respectively.

It is not possible to go into more detail on this topic at this juncture; suffice it to say that you will encounter DEE and DUM many times in the pages ahead. For further discussion, see reference [6.5].

مثال‌هایی از قواعد جامعیت

- دستور تعریف دامنه (میدان)

```
CREATE DOMAIN domain-name datatype
              [default-definition]
              [domain-constraint-definition-list]
```

- مثال

```
CREATE DOMAIN DEGREE CHAR(3) DEFAULT '???'
          CONSTRAINT VALID-DEGREES
               CHECK VALUE IN ('bs','ms','doc','???')
```

- دستور تعریف "اظهار"

```
CREATE ASSERTION constraint-name
[BEFORE COMMIT | AFTER{INSERT|DELETE|UPDATE[OF(column-name)]}
ON table-name ...]
CHECK condition(s)
[FOR [EACH ROW OF] table-name]
```

- مثال۱:

```
CREATE ASSERTION GRADECHECK
BEFORE INSERT ON STCOT CHECK (0≤GRADE≤20);
```

- مثال۲: رابطه‌ی R (A,B, ...)

```
CREATE ASSERTION BFDA
CHECK (NOT EXISTS (SELECT * FROM R
                   GROUP BY A HAVING MAX(B) ≠ MIN(B)));
```

- مثال۳: رابطه‌ی R (A, B, C, D) مفروض است.

```
CREATE ASSERTION K
CHECK (NOT EXISTS (SELECT * FROM R
                   WHERE A = B AND C ≠ D));
```

- مثال۴:

```
CREATE ASSERTION C
CHECK ((SELECT COUNT(*) FROM(
(SELECT STID FROM STT WHERE STDEID = 'D11') EXCEPT
(SELECT STID FROM STCOT WHERE COID = 'C11')) AS N) = 0);
```

- مثال۵:

```
CREATE ASSERTION A
CHECK (NOT EXISTS (SELECT * FROM STT WHERE STJ = 'comp'
                   AND NOT EXISTS(SELECT * FROM STCOT
                   WHERE STCOT.STID = STT.STID
                   AND STCOT.COID = 'C12')));
```

• دستور تعریف رهانا (TRIGGER)

```
CREATE TRIGGER name
{ BEFORE |AFTER| INSTEAD OF }
{ INSERT |DELETE| UPDATE OF Column }
ON Table name [ORDER value]
[REFERENCING {NEW |OLD| NEW- TABLE|OLD- TABLE } AS name]
{ WHEN Condition (n)}
(SQL 3 Procedure
[FOR EACH {ROW|STATEMENT}])
```

• مثال ۱: رابطه حاوی مشخصات کارمند و میزان حقوق او را در نظر می‌گیریم :

EMPREL (EMPID , EMPNAME , ... , EMPSAL)
C.K.

محدودیت موردنظر چنین است : "حقوق کارمند هیچگاه کاهش نمی‌یابد" . رهانای زیر این محدودیت را اعمال می‌کند :

```
CREATE TRIGGER EMPREL- UPDATE- TRIG
BEFORE UPDATE OF EMPSAL ON EMPREL
REFERENCING OLD AS OEMPREL , NEW AS NEMPREL ,
(WHEN OEMPREL.EMPSAL > NEMPREL . EMPSAL
SIGNAL.SQL State ' 7005' ('salary can not be decreased')
FOR EACH ROW) ;
```

• مثال ۲: در این مثال فرض می‌کنیم علاوه بر رابطه EMPREL ، رابطه زیر را هم داریم :

EMPSALAUG (EMPID , AUGM)
C.K.

صفت AUGM مقدار افزایش حقوق کارمند است .

رهانای زیر سازگاری داده‌ای پایگاه را پس از عمل بهنگام سازی تضمین می‌کند .

```
CREATE TRIGGER DBUPDATETRIG
AFTER UPDATE OF EMPSAL ON EMPREL
REFERENCING OLD AS OEMPREL , NEW AS NEMPREL
(UPDATE EMPSALAUG
   SET AUGM = AUGM + NEMPREL.EMPSAL – OEMPREL.EMPSAL
   WHERE EMPSALAUG.EMPID = EMPREL.EMPID
   FOR EACH ROW) ;
```

• مثال ۳: این مثال برگرفته از [ELMA 2000] است . رابطه‌های زیر مفروضند :

EMPLOYEE (NAME , SSN , SALARY , DNO , SUPERVISOR – SSN)
DEPARTMENT (DNAME , DNO , TOTAL-SAL , MANAGER-SSN)

(a)

```
R1: CREATE TRIGGER TOTALSAL1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.DNO IS NOT NULL)
       UPDATE DEPARTMENT
       SET TOTAL_SAL = TOTAL_SAL + NEW.SALARY
```

ادامه مثال ۳ :

```
R2 : CREATE TRIGGER TOTALSAL2
     AFTER UPDATE OF SALARY ON EMPLOYEE
     FOR EACH ROW
     WHEN (NEW.DNO IS NOT NULL)
        UPDATE DEPARTMENT
        SET TOTAL_SAL = TOTAL_SAL + NEW.SALARY- OLD.SALARY
        WHERE  DNO = NEW.DNO ;

R3 : CREATE TRIGGER TOTALSAL3
     AFTER UPDATE OF DNO ON EMPLOYEE
     FOR EACH ROW
          BEGIN
           UPDATE DEPARTMENT
           SET TOTAL_SAL = TOTAL_SAL + NEW.SALARY
           WHERE DNO = NEW.DNO ;
           UPDATE DEPARTMENT
           SET TOTAL_SAL = TOTAL_SAL - OLD.SALARY
           WHERE DNO = OLD.DNO ;
          END ;

R4 : CREATE TRIGGER TOTALSAL 4
     AFTER DELETE ON EMPLOYEE
     FOR EACH ROW
     WHEN (OLD.DNO IS NOT NULL)
        UPDATE DEPARTMENT
        SET TOTAL_SAL = TOTAL_SAL - OLD.SALARY
        WHERE  DNO = OLD.DNO ;
```

(b)

```
R5 : CREATE TRIGGER INFORM_SUPERVISOR1
     BEFORE INSERT OR UPDATE OF SALARY , SUPERVISOR_SSN
     ON EMPLOYEE
     FOR EACH ROW
     WHEN
        (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
                       WHERE  SSN = NEW.SUPERVISOR_SSN))
      INFORM_SUPERVISOR (NEW.SUPERVISOR_SSN, NEW.SSN) ;
```

تمرین : عملکرد هر یک از این برهانا "ها را توضیح دهید.

■ مثال ۴: کاربرد رهانا در عملیات از دید در پایگاه داده‌ها: اعمال "محدودیت دید":

فرض می‌کنیم دید V چنین تعریف شده باشد:

```
CREATE VIEW V
  AS SELECT STID, STNAME, STDEG, STDEID
    FROM STT
    WHERE STMJR = 'comp';
```

اگر کاربر بخواهد تاپلی از این دید در پایگاه داده‌ها درج کند، در رابطه مبنای STT برای صفت STMJR، "هیچ‌مقدار" و یا "مقدار پیش‌نهاده" وارد می‌شود و چنانچه مقدار پیش‌نهاده "comp" نباشد، در بازیابی از دید V، کاربر تاپل درج شده را نخواهد دید. برای رفع مشکل از رهانا استفاده می‌کنیم، به صورت زیر:

```
CREATE TRIGGER IT001
  INSTEAD OF INSERT ON V
  REFERENCING NEW ROW AS R
  FOR EACH ROW
  INSERT INTO STT
    VALUES( R.STID, R.STNAME, R.STDEG, 'Comp', R.STDEID).
```

با این تکنیک، درج یک تاپل، مثلاً <999, stg, bs, D19> در دید V، منجر به درج تاپل زیر در رابطه STT می‌شود:

<999, stg, 'bs', 'Comp', D19>

و به ترتیب، تاپل جدید، در بازیابی از دید V، دیده خواهد شد.

مثال ۵:

در این مثال قاعده C_2 در عمل حذف اعمال می‌شود.

```
CREATE TRIGGER COTDELTRIG
BEFORE DELETE ON COT
REFERENCING OLD AS OLCO
(DELETE FROM STCOT
   WHERE STCOT.COID = OLCO.COID
   FOR EACH ROW);
```

■ مثال ۶:

در این مثال، قاعده C_2 در عمل درج اعمال می‌شود.

```
CREATE TRIGGER STCOTITRIG
  BEFORE INSERT ON STCOT
  REFERENCING NEW AS NSTCOT
  (WHEN ( NOT EXISTS ( SELECT * FROM COT WHERE COT.COID =
                       NSTCOT.COID))
ABORT TRANSACTION FOR EACH ROW );
```

مثال ۷ : گزینهٔ FOR EACH STATEMENT

رابطه EMPL (EMID , ENAME , ... , EJOB , ESAL)

را در نظر می‌گیریم. محدودیت زیر مفروض است:

"میانگین حقوق کارمندان نباید از 'a' ریال کمتر باشد".

چون با یک دستور درج یا بهنگام‌سازی ممکن است تعدادی سطر درج شوند و یا بهنگام در آیند، در اثنای این تغییرات در رابطه EMPL ممکن است میانگین حقوق از 'a' ریال موقتاً کمتر شود ولی پس از پایان اجرای دستور درج یا بهنگام‌سازی، از 'a' ریال بیشتر شود.

بنابراین اگر بخواهیم محدودیت داده شده را به کمک رهانا کنترل کنیم، باید از گزینهٔ FOR EACH STATEMENT استفاده کنیم. رهانای زیر این کار را در عمل بهنگام‌سازی انجام می‌دهد.

```
CREATE TRIGGER AVSALTRIG
  AFTER UPDATE OF ESAL ON EMPL
  REFERENCING
      OLD AS OEMPL , NEW AS NEMPL
  FOR EACH STATEMENT
    WHEN 'a' > (SELECT AVG(ESAL)
                FROM EMPL)
    BEGIN
      DELETE FROM EMPL
        WHERE (EMID , ENAME , ..... , EJOB , ESAL)
          IN NEMPL;
      INSERT INTO EMPL
          SELECT * FROM OEMPL ;
    END;
```

جبر رابطه ای

۱- خواص عملگرها

رابطه های R ، S و T به ترتیب با مجموعه صفات زیر مفروضند:

$H_R = \{A_1, A_2, ..., A_n\}$

$H_S = \{B_1, B_2, ..., B_n\}$

$H_T = \{C_1, C_2, ..., C_n\}$

و نیز فرض کنیم : p، q، r مسند و p_{Atts}، q_{Atts} و r_{Atts} به ترتیب مجموعه صفات استفاده شده در این مسندها باشند و L_1، L_2، L_3، M_1، M_2 و N، هریک،یک مجموعه از صفات باشند.

خواص زیر در عملگرهای جبر رابطه ای برقرارند[CONN2002]:

1) $\sigma_{p \wedge q \wedge r}(R) = \sigma_p(\sigma_q(\sigma_r(R))$

2) $\sigma_p(\sigma_q(R)) = \sigma_q(\sigma_p(R))$

3) $\prod_{<L>} \prod_{<M>} ..... \prod_{<N>}(R) = \prod_{<L>}(R)$, $L \subset M \subset N$

4) $\prod_{<A_1,...,A_n>}(\sigma_p(R)) = \sigma_p(\prod_{<A_1,...,A_n>}(R))$ $\quad p_{Atts} \in \{A_1, A_2, ..., A_n\}$

5) $R \bowtie_p S = S \bowtie_p R$

$R \times S = S \times R$

6) $\sigma_p(R \bowtie_r S) = (\sigma_p(R) \bowtie_r S)$

$\sigma_p(R \times S) = (\sigma_p(R)) \times S$

$p_{Atts} \in \{A_1, A_2, ..., A_n\}$

7) $\sigma_{p \wedge q}(R \bowtie_r S) = (\sigma_p(R) \bowtie_r \sigma_q(S))$

$\sigma_{p \wedge q}(R \times S) = (\sigma_p(R) \times \sigma_q(S))$

B

8) $\prod_{<L_1 \cup L_2>}(R \bowtie_r S) = (\prod_{<L_1>}(R)) \bowtie_r (\prod_{<L_2>}(S))$

به شرطی که: L_1 فقط شامل صفاتی از H_R و L_2 فقط شامل صفاتی از H_S باشند و در شرط پیوند فقط صفاتی از $L_1 \cup L_2$ به کار رفته باشد.

این خاصیت در حالت ضرب کارتزین R و S نیز برقرار است.

اما اگر در شرط پیوند صفات دیگری خارج از صفات $L_1 \cup L_2$ به کار رفته باشد، مثلا مجموعه صفات $M_1 \cup M_2$ به نحوی که M_1 فقط شامل صفاتی از H_R و M_2 فقط شامل صفاتی از H_S باشد در این صورت:

$$\prod_{<L_1 \cup L_2>}(R \bowtie_r S) = \prod_{<L_1 \cup L_2>}(\prod_{<L_1 \cup M_1>}(R)) \bowtie_r (\prod_{<L_2 \cup M_2>}(S))$$

9) $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

توجه! در اینجا منظور عملگر پیوند طبیعی است.

$(R \times S) \times T = R \times (S \times T)$

اگر شرط پیوند q تنها شامل صفاتی از H_S و H_T باشد در این صورت داریم:

$(R \bowtie_p S) \bowtie_{q \wedge r} T = R \bowtie_{p \wedge r} (S \bowtie_q T)$

10) $R \cup S = S \cup R$

$R \cap S = S \cap R$

11) $\sigma_p(R \cup S) = \sigma_p(S) \cup \sigma_p(R)$

$\sigma_p(R \cap S) = \sigma_p(S) \cap \sigma_p(R)$

$\sigma_p(R - S) = \sigma_p(R) - \sigma_p(S)$

12) $\prod_{<L>}(R \cup S) = \prod_{<L>}(S) \cup \prod_{<L>}(R)$

13) $(R \cup S) \cup T = S \cup (R \cup T)$

$(R \cap S) \cap T = S \cap (R \cap T)$

۲- مجموعه کامل عملگرهای جبر رابطه‌ای:

- σ : RESTRICT
- Π : PROJECT
- $\cup$: UNION
- $-$: MINUS
- $\times$: TIMES

RDB :
- S(S#, SNAME, STATUS, CITY)
- P(P#, PNAME, COLOR, WEIGHT, CITY)
- SP(S#, P#, QTY)

۳- مثالهایی از جبر رابطه‌ای :

• Q_1 : نام تهیه کنندگانی را بدهید که قطعه P_2 را تهیه می‌کنند.

```
((SP JOIN S) where P# = 'P2')[SNAME]
```

و با نمادهای ریاضی :

$$\Pi_{\langle SNAME \rangle}\left(\sigma_{\langle P\# = 'P2' \rangle}(SP \bowtie S)\right)$$

• Q_2 : نام تهیه کنندگانی را بدهید که حداقل یک قطعه قرمز رنگ تهیه می‌کنند.

```
(((P where COLOR='Red') JOIN SP)[S#] JOIN S)[SNAME]
```

• Q_3 : نام تهیه کنندگانی را بدهید که تمام قطعات را تهیه می‌کنند :

```
((SP[S#, P#] DIVIDE BY P[P#]) JOIN S)[SNAME]
```

• Q_4 : شماره تهیه کنندگانی را بدهید که حداقل تمام قطعات تهیه شده توسط 'S_2' را تهیه می‌کنند

```
SP[S#, P#] DIVIDE BY (SP where S# = 'S2') [P#]
```

• Q_5 : نام تهیه کنندگانی را بدهید که قطعه 'P_2' را تهیه نمی‌کنند.

```
((S[S#] MINUS (SP where P# = 'P2')[S#]) JOIN S)[SNAME]
```

⊛ عملگر EXTEND

```
1) EXTEND S ADD 'supplier' AS TAG.
2) (EXTEND S ADD CITY AS SCITY)[S#, SNAME,
                                STATUS, SCITY].
```

معادل است با : S RENAME CITY AS SCITY

```
3) EXTEND (P JOIN SP) ADD (WEIGHT*QTY)
                                AS SHIPWT.
```

⊛ عملگر SUMMARIZE

```
1) SUMMARIZE SP BY (P#) ADD SUM(QTY) AS TOTQTY.
```

2) SUMMARIZE SP BY (S#) ADD COUNT AS NP.

3) SUMMARIZE (P JOIN SP) BY (CITY) ADD COUNT AS NSP.

جدول جواب می‌تواند چنین باشد:

CITY	NSP
C_1	5
C_2	6
C_3	1

4) SUMMARIZE SP BY (P#) ADD SUM (QTY) AS TOTQTY, AVG (QTY) AS AVGQTY.

• عملگرهای جمعی Aggregate operators :

از جمله: COUNT ، SUM ، MIN ، MAX ، AVG ، .....

شکل کلی: <op name> (relation exp.) [. <att. name>])

در عملگر COUNT نیازی به نام صفت نیست.

• مثال: SUM (SP WHERE S# = S1 , QTY)

← کنجکاوی [اختیاری]: عملگرهای GROUP و UNGROUP چه عملکردی دارند؟

← توجه:

1) $(R_1 \div R_2) \times R_2 \subseteq R_1$

2) $R_1(X,Y) \div R_2(Y) = \Pi_{<X>}(R_1) - \left(\Pi_{<X>}\left(\Pi_{<X>}(R_1) \times R_2 - R_1\right)\right)$

B

$$RDB: \begin{cases} STT(\underline{STID}, STNAME, STDEG, STMJR, STDEID) \\ COT(\underline{COID}, COTITLE, COTYPE, CREDIT, CODEID) \\ STCOT(\underline{STID}, \underline{COID}, TR, YRYR, GRADE) \end{cases}$$

۴- مثالهای دیگر از جبر رابطه ای (با نمادهای ریاضی)

Q_1 - عنوان درسهایی را بدهید که دانشجوی با شماره 'N' در ترم اول ۸۳-۸۲ انتخاب کرده است:

$$\Pi_{\langle COTITLE \rangle}\left(\sigma_{STID='N' \wedge TR='1' \wedge YRYR='82-83'}(STCOT) \bowtie COT\right)$$

Q_2 - نام دانشجویانی را بدهید که در درس با عنوان 'C' در ترم دوم ۸۲-۸۱ مردود شده اند.

$$\Pi_{\langle STNAME \rangle}\left(\sigma_{COTITLE='C' \wedge TR='2' \wedge YRYR='81-82' \wedge GRADE<10}(STCOT \bowtie COT) \bowtie STT\right)$$

Q_3 - نام و رشته دانشجویانی را بدهید که حداقل یک درس عملی در ترم اول ۸۳-۸۲ انتخاب کرده اند.

$$\Pi_{\langle STNAME, STMJR \rangle}\left(\left(\sigma_{COTYPE='p'}(COT) \bowtie \sigma_{TR='1' \wedge YRYR='82-83'}(STCOT)\right) \bowtie STT\right)$$

Q_4 - عنوان درسهای عملی را بدهید که دانشجویان گروه آموزشی 'D1' در سال ۸۲-۸۱ انتخاب نکرده اند.

$$\Pi_{\langle COTITLE \rangle}\left(\left(\Pi_{\langle COID \rangle}\left(\sigma_{COTYPE='p'}(COT)\right) - \left(\Pi_{\langle COID \rangle}\left(\sigma_{STDEID='D1' \wedge YRYR='81-82'}(STT \bowtie STCOT)\right)\right)\right) \bowtie COT\right)$$

Q_5 - عنوان درسهایی را بدهید که تمام دانشجویان رشته کامپیوتر در ترم دوم ۸۲-۸۱ در آنها قبول شده باشند.

$$\Pi_{\langle COTITLE \rangle}\left(\Pi_{\langle STID, COID \rangle}\left(\sigma_{TR='2' \wedge YRYR='81-82' \wedge GRADE \geq 10}(STCOT)\right) \div \left(\Pi_{\langle STID \rangle}\left(\sigma_{STMJR='comp'}\right)\right) \bowtie COT\right)$$

تمرین:

۱- پرسش Q_4 را با عملگر SEMIMINUS بنویسید.

۲- عنوان هر رشته و تعداد دانشجویان هر رشته را بدهید.

۳- پرسش Q_5 را بدون استفاده از عملگر تقسیم بدهید.

حساب رابطه‌ای

۱- روابط هم‌ارزی بین سورهای وجودی و عمومی :

- FORALL T (f) ≡ NOT EXISTS T (NOT f)
- EXISTS T (f) ≡ NOT (FORALL T (NOT f))
- FORALL T ((f) AND (g)) ≡ NOT EXISTS T(NOT (f) OR NOT (g))
- FORALL T ((f) OR (g)) ≡ NOT EXISTS T(NOT (f) AND NOT (g))
- EXISTS T ((f) OR (g)) ≡ NOT FORALL T (NOT (f) AND NOT (g))
- EXISTS T((f) AND (g)) ≡ NOT FORALL T (NOT (f) OR NOT (g))

و نیز :

- FORALL T (f) ⇒ EXISTS T (f)
- NOT EXISTS T (f) ⇒ NOT FORALL T (f)

← کنجکاوی [اختیاری] :

می‌توان سور EXISTS را به صورت یک "iterated OR" و سور FORALL را به صورت یک "iterated AND" تعریف کرد. این مطلب یعنی چه؟

← برای مطالعه بیشتر [اختیاری] : الگوریتم کاهش کاد (CODD's reduction alg.) مطالعه شود (این الگوریتم می‌تواند هر عبارت دلخواه حساب رابطه‌ای را به عبارت معادل آن در جبر رابطه‌ای تبدیل کند).

۴- مثال هایی از حساب رابطه ای:

* SX, PX, SPX, SPY : متغیرهای تاپلی

• Q_1 : شماره تهیه کنندگان ساکن 'c2' با وضعیت بیش از 20 را بدهید:

• SX.S# where SX.CITY = 'c2' AND SX.STATUS > 20.

• Q_2 : نام تهیه کنندگان قطعه 'p2' را بدهید:

• SX.SNAME where EXISTS SPX (SPX.S# = SX.S#
AND
SPX.P# = 'p2').

• Q_3 : نام تهیه کنندگانی که حداقل یک قطعه قرمز رنگ تهیه می کنند:

• SX.SNAME where EXISTS SPX (SX.S# = SPX.S#
AND
EXISTS PX (PX.P# = SPX.P#
AND
PX.COLOR = 'Red'))

• Q_4 : نام تهیه کنندگانی که حداقل یک قطعه تهیه شده توسط 's2' را تهیه می کنند:

• SX.SNAME where EXISTS SPX (EXISTS SPY (SX.S# = SPX.S#
AND
SPX.P# = SPY.P#
AND
SPY.S# = 's2'))

• Q_5 : نام تهیه کنندگانی که تمام قطعات را تهیه می کنند:

• SX.SNAME where FORALL PX (EXISTS SPX (SPX.S# = SX.S#
AND
SPX.P# = PX.P#)

• SX.SNAME where NOTEXISTS PX (NOTEXISTS SPX :یا
(SPX.S# = SX.S#
AND
SPX.P# = PX.P#)).

• Q_6 : نام تهیه کنندگانی که قطعه 'p2' را تهیه نمی کنند:

• SX.SNAME where NOTEXISTS SPX (SPX.S# = SX.S#
AND
SPX.P# = 'p2').

۳۰- مثال‌های دیگری از حساب رابطه‌ای

متغیرهای تاپلی ST ، CO ، STCO تعریف شده به ترتیب روی رابطه‌های STT ، COT و STCOT مفروضند.

Q_1 - شماره و عنوان درس‌های دو واحدی را بدهید.

```
(CO.COID , CO.COTITLE ) where CO.CREDIT = '2';
```

Q_2 - عنوان درس‌هایی را بدهید که دانشجوی با شماره 'N' در ترم اول ۸۳-۸۲ انتخاب کرده است.

```
CO.COTITLE where EXISTS STCO ( STCO.COID = CO.COID
AND STCO.STID = 'N' AND STCO.TR = '1' AND STCO.YRYR = '82-83')
```

Q_3 - نام دانشجویانی را بدهید که درس با شماره 'c' را انتخاب نکرده‌اند:

```
ST.STNAME where NOTEXISTS STCO( STCO.STID = ST.STID
                                AND
                                STCO.COID = 'c')
```

Q_4 - نام و رشته دانشجویانی را بدهید که حداقل یک درس عملی در ترم اول ۸۳-۸۲ انتخاب کرده‌اند.

```
ST.STNAME , ST.STMJR where EXISTS STCO ( ST.STID = STCO.STID
AND STCO.TR = '1' AND STCO.YRYR = '82-83' AND EXISTS CO
( CO.COID = STCO.COID AND CO.COTYPE = 'p'))
```

Q_5 - عنوان درس‌هایی را بدهید که تمام دانشجویان رشته کامپیوتر در ترم دوم ۸۲-۸۱ در آنها قبول شده باشند.

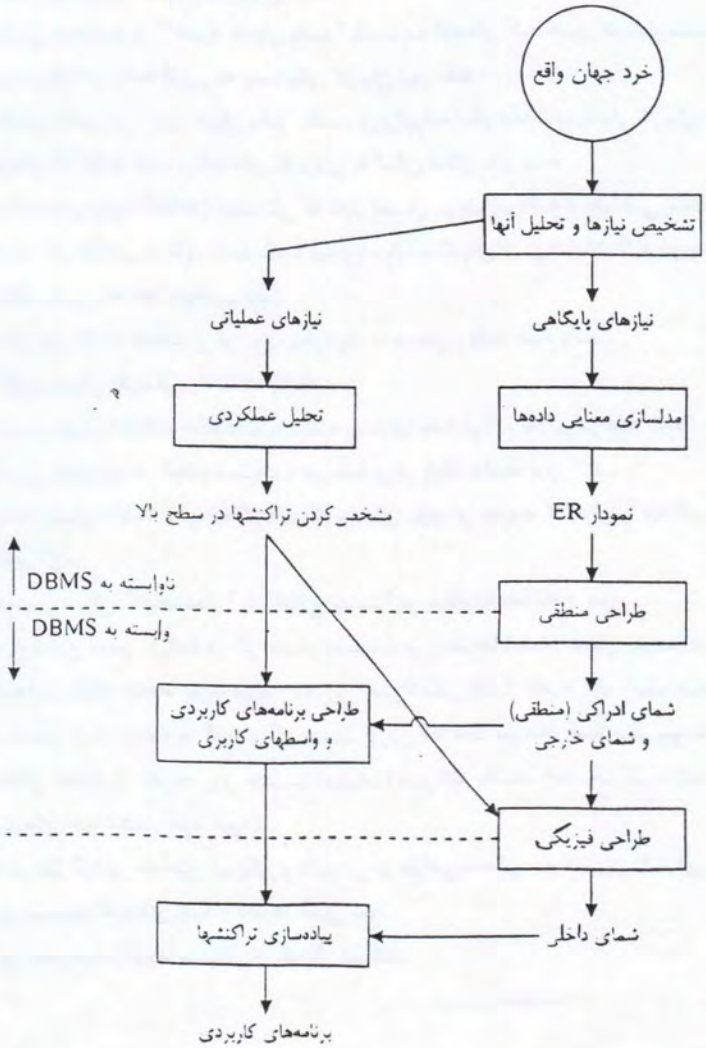
```
CO.COTITLE where FORALL ST(EXISTS STCO (
STCO.STID = ST.STID AND STCO.COID = CO.COID
AND STCO.TR = '2' AND STCO.YRYR = '82-83'
AND STCO.GRADE ≥ '10' AND ST.STMJR = 'comp'))
```

• تمرین: پرسش Q_5 بدون استفاده از FORALL نوشته شود.

طراحی ...

۱- مراحل اساسی طراحی پایگاه داده‌ها (ساده شده)

برای جزئیات کار به وظایف DBA (در یادداشت‌های تکمیلی) مراجعه شود.



مأخذ: ELMASRI --- 2003

۲ـ خصوصیات طراحی خوب

طراحی منطقی خوب باید خصوصیات زیر را داشته باشد :

۱- نمایشی واضح از " خرد جهان واقع" باشد .

۲- نمایشی صحیح از " خرد جهان واقع" باشد به گونه‌ای که هیچ اشتباه معنایی (سمنتیک) در پاسخگویی به پرسشهای کاربران بروز نکند .

۳- نمایشی جامع از " خرد جهان واقع" باشد . دربرگیرنده تمام صفات موردنیاز کاربران ، به گونه‌ای که تولید همه برنامه‌های کاربردی به آسانی امکان پذیر باشد .

۴- تمام محدودیتها (قواعد) جامعیتی که قابل اعمال در هر مرحله از طراحی منطقی باشند ، در طراحی منظور شده باشند (به ویژه وابستگیهای بین صفات که در بحث نرمالترسازی رابطه‌ها خواهیم دید).

۵- معنای هر یک از صفات از هر نوع موجودیت به درستی رعایت شده باشد .

۶- کمترین میزان افزونگی را داشته باشد .

۷- کمترین میزان اختلاط اطلاعات را داشته باشد (به بحث نرمالترسازی مراجعه شود) .

۸- کمترین دشواری در انجام عملیات ذخیره‌سازی در پایگاه داده‌ها بروز کند .

۹- انعطاف‌پذیری داشته باشد به گونه‌ای که بتوان نیازهای جدید کاربران را به آسانی منظور کرد .

۱۰- کمترین میزان "هیچمقدار" یا " اطلاع نیست" در پایگاه داده‌ها ایجاد شود .

۱۱- هیچ اطلاع حشو (زائد) در اثر انجام عملیات در رابطه‌ها (مثلاً عمل پیوند دو رابطه) در پایگاه داده‌ها پدید نیاید . بنابراین حتی‌الامکان باید از تجزیه یک رابطه به دو یا بیش از دو رابطه به گونه‌ای که بدست آوردن اطلاعات موردنظر نیازمند پیوند رابطه‌ای حاصل از تجزیه روی صفت (صفات) غیرکلید باشد . اجتناب کرد (به گفتارهای چهاردهم رجوع شود) .

۱۲- با در نظر گرفتن طراحی فیزیکی و تأثیر آن در طراحی منطقی . بیشترین کارآیی برای سیستم کاربردی پایگاه داده‌ها تأمین شود .

بعضی از این خصوصیات لزوماً مستقل از یکدیگر نیستند .

کنجکاوی : آیا خصوصیات دیگری هم مطرح است ؟

B - course(1) تمرینات تکمیلی در جبر و حساب

Query Exercises

۱- صورت پرسش ها (پرسش و جواب)

The remaining exercises are all based on the suppliers-parts-projects database (see Fig. 4.5 in the "Exercises" section in Chapter 4 and the answer to Exercise 5.4 in Chapter 5). In each case you are asked to write a relational algebra expression for the indicated query. (By way of an interesting variation, you might like to try looking at some of the answers first and stating what the given expression means in natural language.) For convenience we show the structure of the database (in outline) below:

```
S    { S#, SNAME, STATUS, CITY }
     PRIMARY KEY { S# }
P    { P#, PNAME, COLOR, WEIGHT, CITY }
     PRIMARY KEY { P# }
J    { J#, JNAME, CITY }
     PRIMARY KEY { J# }
SPJ  { S#, P#, J#, QTY }
     PRIMARY KEY { S#, P#, J# }
     FOREIGN KEY { S# } REFERENCES S
     FOREIGN KEY { P# } REFERENCES P
     FOREIGN KEY { J# } REFERENCES J
```

6.13 Get full details of all projects.

6.14 Get full details of all projects in London.

6.15 Get supplier numbers for suppliers who supply project J1.

6.16 Get all shipments where the quantity is in the range 300 to 750 inclusive.

6.17 Get all part-color/part-city combinations. *Note:* Here and subsequently, the term "all" is to be taken to mean "all currently represented in the database," not "all possible."

6.18 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are all colocated (i.e., all in the same city).

6.19 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are not all colocated.

6.20 Get all supplier-number/part-number/project-number triples such that no two of the indicated supplier, part, and project are colocated.

6.21 Get full details for parts supplied by a supplier in London.

6.22 Get part numbers for parts supplied by a supplier in London to a project in London.

6.23 Get all pairs of city names such that a supplier in the first city supplies a project in the second city.

6.24 Get part numbers for parts supplied to any project by a supplier in the same city as that project.

6.25 Get project numbers for projects supplied by at least one supplier not in the same city.

6.26 Get all pairs of part numbers such that some supplier supplies both the indicated parts.

6.27 Get the total number of projects supplied by supplier S1.

6.28 Get the total quantity of part P1 supplied by supplier S1.

6.29 For each part being supplied to a project, get the part number, the project number, and the corresponding total quantity.

6.30 Get part numbers of parts supplied to some project in an average quantity of more than 350.

6.31 Get project names for projects supplied by supplier S1.

6.32 Get colors of parts supplied by supplier S1.

6.33 Get part numbers for parts supplied to any project in London.

6.34 Get project numbers for projects using at least one part available from supplier S1.

Q-1 DATE 2000 : مأخذ

6.35 Get supplier numbers for suppliers supplying at least one part supplied by at least one supplier who supplies at least one red part.

6.36 Get supplier numbers for suppliers with a status lower than that of supplier S1.

6.37 Get project numbers for projects whose city is first in the alphabetic list of such cities.

6.38 Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.

6.39 Get supplier numbers for suppliers supplying some project with part P1 in a quantity greater than the average shipment quantity of part P1 for that project.

6.40 Get project numbers for projects not supplied with any red part by any London supplier.

6.41 Get project numbers for projects supplied entirely by supplier S1.

6.42 Get part numbers for parts supplied to all projects in London.

6.43 Get supplier numbers for suppliers who supply the same part to all projects.

6.44 Get project numbers for projects supplied with at least all parts available from supplier S1.

6.45 Get all cities in which at least one supplier, part, or project is located.

6.46 Get part numbers for parts that are supplied either by a London supplier or to a London project.

6.47 Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.

6.48 Get all pairs of supplier numbers, S*x* and S*y* say, such that S*x* and S*y* supply exactly the same set of parts each. (Thanks to a correspondent, Fatma Mili of Oakland University, Rochester, Michigan, for this problem. For simplicity, you might want to use the original suppliers and parts database for this exercise, instead of the expanded suppliers-parts-projects database.)

6.49 Get a "grouped" version of all shipments showing, for each supplier-number/part-number pair, the corresponding project numbers and quantities in the form of a binary relation.

6.50 Get an "ungrouped" version of the relation produced in Exercise 6.49.

۲- پاسخ در جبر رابطه‌ای

6.13 J

6.14 J WHERE CITY = 'London'

6.15 (SPJ WHERE J# = J# ('J1')) { S# }

6.16 SPJ WHERE QTY ≥ QTY (300) AND QTY ≤ QTY (750)

6.17 P { COLOR, CITY }

6.18 (S JOIN P JOIN J) { S#, P#, J# }

6.19
```
( ( ( S RENAME CITY AS SCITY ) TIMES
    ( P RENAME CITY AS PCITY ) TIMES
    ( J RENAME CITY AS JCITY ) )
WHERE SCITY ≠ PCITY
OR    PCITY ≠ JCITY
OR    JCITY ≠ SCITY ) { S#, P#, J# }
```

6.20
```
( ( ( S RENAME CITY AS SCITY ) TIMES
    ( P RENAME CITY AS PCITY ) TIMES
    ( J RENAME CITY AS JCITY ) )
WHERE SCITY ≠ PCITY
AND   PCITY ≠ JCITY
AND   JCITY ≠ SCITY ) { S#, P#, J# }
```

6.21 P SEMIJOIN (SPJ SEMIJOIN (S WHERE CITY = 'London'))

6.22 Just to remind you of the possibility, we show a step-at-a-time solution to this exercise:

```
WITH ( S WHERE CITY = 'London' ) AS T1,
     ( J WHERE CITY = 'London' ) AS T2,
     ( SPJ JOIN T1 ) AS T3,
     T3 { P#, J# } AS T4,
     ( T4 JOIN T2 ) AS T5 :
     T5 { P# }
```

Here is the same query without using WITH:

```
( ( SPJ JOIN ( S WHERE CITY = 'London' ) ) { P#, J# }
                JOIN ( J WHERE CITY = 'London' ) ) { P# }
```

We will give a mixture of solutions (some using WITH, some not) to the remaining exercises.

6.23
```
( ( S RENAME CITY AS SCITY ) JOIN SPJ JOIN
  ( J RENAME CITY AS JCITY ) ) { SCITY, JCITY }
```

6.24 (J JOIN SPJ JOIN S) { P# }

6.25
```
( ( ( J RENAME CITY AS JCITY ) JOIN SPJ JOIN
    ( S RENAME CITY AS SCITY ) )
                    WHERE JCITY ≠ SCITY ) { J# }
```

6.26
```
WITH ( SPJ { S#, P# } RENAME S# AS XS#, P# AS XP# ) AS T1,
     ( SPJ { S#, P# } RENAME S# AS YS#, P# AS YP# ) AS T2,
     ( T1 TIMES T2 ) AS T3,
     ( T3 WHERE XS# = YS# AND XP# < YP# ) AS T4 :
     T4 { XP#, YP# }
```

6.27
```
( SUMMARIZE SPJ { S#, J# }
  PER RELATION { TUPLE { S# S# ( 'S1' ) } }
  ADD COUNT AS N ) { N }
```

We remind you that the expression in the PER clause here is a relation selector invocation (in fact, it is a relation literal).

6.28
```
( SUMMARIZE SPJ { S#, P#, QTY }
  PER RELATION { TUPLE { S# S# ( 'S1' ), P# P# ( 'P1' ) } }
  ADD SUM ( QTY ) AS Q ) { Q }
```

6.29
```
SUMMARIZE SPJ PER SPJ { P#, J# } ADD SUM ( QTY ) AS Q
```

6.30
```
WITH ( SUMMARIZE SPJ PER SPJ { P#, J# }
                     ADD AVG ( QTY ) AS Q ) AS T1,
       ( T1 WHERE Q > QTY ( 350 ) ) AS T2 :
       T2 { P# }
```

6.31
```
( J JOIN ( SPJ WHERE S# = S# ( 'S1' ) ) ) { JNAME }
```

6.32
```
( P JOIN ( SPJ WHERE S# = S# ( 'S1' ) ) ) { COLOR }
```

6.33
```
( SPJ JOIN ( J WHERE CITY = 'London' ) ) { P# }
```

6.34
```
( SPJ JOIN ( SPJ WHERE S# = S# ( 'S1' ) ) { P# } ) { J# }
```

6.35
```
( ( ( SPJ JOIN
      ( P WHERE COLOR = COLOR ( 'Red' ) ) { P# } ) { S# }
                        JOIN SPJ ) { P# } JOIN SPJ ) { S# }
```

6.36
```
WITH ( S { S#, STATUS } RENAME S# AS XS#,
                               STATUS AS XSTATUS ) AS T1,
       ( S { S#, STATUS } RENAME S# AS YS#,
                               STATUS AS YSTATUS ) AS T2,
       ( T1 TIMES T2 ) AS T3,
       ( T3 WHERE XS# = S# ( 'S1' ) AND
                  XSTATUS > YSTATUS ) AS T4 :
       T4 { YS# }
```

6.37
```
( ( EXTEND J ADD MIN ( J, CITY ) AS FIRST )
                                 WHERE CITY = FIRST ) { J# }
```

What does this query return if relvar J is empty?

6.38
```
WITH ( SPJ RENAME J# AS ZJ# ) AS T1,
       ( T1 WHERE ZJ# = J# AND P# = P# ( 'P1' ) ) AS T2,
       ( SPJ WHERE P# = P# ( 'P1' ) ) AS T3,
       ( EXTEND T3 ADD AVG ( T2, QTY ) AS QX ) AS T4,
       T4 { J#, QX } AS T5,
       ( SPJ WHERE J# = J# ( 'J1' ) ) AS T6,
       ( EXTEND T6 ADD MAX ( T6, QTY ) AS QY ) AS T7,
       ( T5 TIMES T7 { QY } ) AS T8,
       ( T8 WHERE QX > QY ) AS T9 :
       T9 { J# }
```

6.39
```
WITH ( SPJ WHERE P# = P# ( 'P1' ) ) AS T1,
       T1 { S#, J#, QTY } AS T2,
       ( T2 RENAME J# AS XJ#, QTY AS XQ ) AS T3,
       ( SUMMARIZE T1 PER SPJ { J# }
                      ADD AVG ( QTY ) AS Q ) AS T4,
       ( T3 TIMES T4 ) AS T5,
       ( T5 WHERE XJ# = J# AND XQ > Q ) AS T6 :
       T6 { S# }
```

6.40
```
WITH ( S WHERE CITY = 'London' ) { S# } AS T1,
       ( P WHERE COLOR = COLOR ( 'Red' ) ) AS T2,
       ( T1 JOIN SPJ JOIN T2 ) AS T3 :
       J { J# } MINUS T3 { J# }
```

6.41
```
J { J# } MINUS ( SPJ WHERE S# ≠ S# ( 'S1' ) ) { J# }
```

6.42
```
WITH ( ( SPJ RENAME P# AS X ) WHERE X = P# ) { J# } AS T1,
       ( J WHERE CITY = 'London' ) { J# } AS T2,
       ( P WHERE T1 ≥ T2 ) AS T3 :
       T3 { P# }
```

6.43
```
S { S#, P# } DIVIDEBY J { J# } PER SPJ { S#, P#, J# }
```

6.44
```
( J WHERE
    ( ( SPJ RENAME J# AS Y ) WHERE Y = J# ) { P# } ≥
    ( SPJ WHERE S# = S# ( 'S1' ) ) { P# } ) { J# }
```

6.45
```
S { CITY } UNION P { CITY } UNION J { CITY }
```

6.46
```
( SPJ JOIN ( S WHERE CITY = 'London' ) ) { P# }
  UNION
( SPJ JOIN ( J WHERE CITY = 'London' ) ) { P# }
```

6.47
```
( S TIMES P ) { S#, P# } MINUS SP { S#, P# }
```

6.48 We show two solutions to this problem.

```
WITH ( SP RENAME S# AS SA ) { SA, P# } AS T1,
     /* T1 {SA,P#} : SA supplies part P# */

     ( SP RENAME S# AS SB ) { SB, P# } AS T2,
     /* T2 {SB,P#} : SB supplies part P# */

     T1 { SA } AS T3,
     /* T3 {SA} : SA supplies some part */

     T2 { SB } AS T4,
     /* T4 {SB} : SB supplies some part */

     ( T1 TIMES T4 ) AS T5,
     /* T5 {SA,SB,P#} : SA supplies some part and
                        SB supplies part P# */

     ( T2 TIMES T3 ) AS T6,
     /* T6 {SA,SB,P#} : SB supplies some part and
                        SA supplies part P# */

     ( T1 JOIN T2 ) AS T7,
     /* T7 {SA,SB,P#} : SA and SB both supply part P# */

     ( T3 TIMES T4 ) AS T8,
     /* T8 {SA,SB} : SA supplies some part and
                     SB supplies some part */

     SP { P# } AS T9,
     /* T9 {P#} : part P# is supplied by some supplier */

     ( T8 TIMES T9 ) AS T10,
     /* T10 {SA,SB,P#} :
        SA supplies some part,
        SB supplies some part, and
        part P# is supplied by some supplier */

     ( T10 MINUS T7 ) AS T11,
     /* T11 {SA,SB,P#} : part P# is supplied,
                         but not by both SA and SB */

     ( T6 INTERSECT T11 ) AS T12,
     /* T12 {SA,SB,P#} : part P# is supplied by SA
                         but not by SB */

     ( T5 INTERSECT T11 ) AS T13,
     /* T13 {SA,SB,P#} : part P# is supplied by SB
                         but not by SA */

     T12 { SA, SB } AS T14,
     /* T14 {SA,SB} :
        SA supplies some part not supplied by SB */

     T13 { SA, SB } AS T15,
     /* T15 {SA,SB} :
        SB supplies some part not supplied by SA */

     ( T14 UNION T15 ) AS T16,
     /* T16 {SA,SB} : some part is supplied by SA or SB
                      but not both */
```

```
    T7 { SA, SB } AS T17,
    /* T17 {SA,SB} :
       some part is supplied by both SA and SB */

    ( T17 MINUS T16 ) AS T18,
    /* T18 {SA,SB} :
       some part is supplied by both SA and SB,
       and no part supplied by SA is not supplied by SB,
       and no part supplied by SB is not supplied by SA
       -- so SA and SB each supply exactly the same parts */

    ( T18 WHERE SA < SB ) AS T19 :
    /* tidy-up step */

T19
```

The second solution—which is much more straightforward!—makes use of the relational comparisons introduced in Section 6.9.

```
WITH ( S RENAME S# AS SA ) { SA } AS RA ,
     ( S RENAME S# AS SB ) { SB } AS RB :
     ( RA TIMES RB )
         WHERE ( SP WHERE S# = SA ) { P# } =
               ( SP WHERE S# = SB ) { P# }
         AND   SA < SB
```

6.49 `SPJ GROUP ( J#, QTY ) AS JQ`

6.50 Let SPQ denote the result of the expression shown in the answer to Exercise 6.49. Then:

```
SPQ UNGROUP JQ
```

۳- پاسخ در حساب رابطه ای

7.13.13 JX

7.13.14
```
JX WHERE JX.CITY = 'London'
```

7.13.15
```
SPJX.S# WHERE SPJX.J# = J# ( 'J1' )
```

7.13.16
```
SPJX WHERE SPJX.QTY ≥ QTY ( 300 ) AND
           SPJX.QTY ≤ QTY ( 750 )
```

7.13.17 (PX.COLOR, PX.CITY)

7.13.18
```
( SX.S#, PX.P#, JX.J# ) WHERE SX.CITY = PX.CITY
                        AND   PX.CITY = JX.CITY
                        AND   JX.CITY = SX.CITY
```

7.13.19
```
( SX.S#, PX.P#, JX.J# ) WHERE SX.CITY ≠ PX.CITY
                        OR    PX.CITY ≠ JX.CITY
                        OR    JX.CITY ≠ SX.CITY
```

7.13.20
```
( SX.S#, PX.P#, JX.J# ) WHERE SX.CITY ≠ PX.CITY
                        AND   PX.CITY ≠ JX.CITY
                        AND   JX.CITY ≠ SX.CITY
```

7.13.21
```
SPJX.P# WHERE EXISTS SX ( SX.S# = SPJX.S# AND
                          SX.CITY = 'London' )
```

7.13.22
```
SPJX.P# WHERE EXISTS SX EXISTS JX
        ( SX.S# = SPJX.S# AND SX.CITY = 'London' AND
          JX.J# = SPJX.J# AND JX.CITY = 'London' )
```

7.13.23
```
( SX.CITY AS SCITY, JX.CITY AS JCITY )
WHERE EXISTS SPJX ( SPJX.S# = SX.S# AND SPJX.J# = JX.J# )
```

7.13.24
```
SPJX.P# WHERE EXISTS SX EXISTS JX
              ( SX.CITY = JX.CITY AND
                SPJX.S# = SX.S# AND
                SPJX.J# = JX.J# )
```

7.13.25
```
SPJX.J# WHERE EXISTS SX EXISTS JX
              ( SX.CITY ≠ JX.CITY AND
                SPJX.S# = SX.S# AND
                SPJX.J# = JX.J# )
```

7.13.26
```
( SPJX.P# AS XP#, SPJY.P# AS YP# )
WHERE SPJX.S# = SPJY.S# AND SPJX.P# < SPJY.P#
```

7.13.27
```
COUNT ( SPJX.J# WHERE SPJX.S# = S# ( 'S1' ) ) AS N
```

7.13.28
```
SUM ( SPJX WHERE SPJX.S# = S# ( 'S1' )
               AND SPJX.P# = P# ( 'P1' ), QTY ) AS Q
```

Note: The following "solution" is *not* correct (why not?):

```
SUM ( SPJX.QTY WHERE SPJX.S# = S# ( 'S1' )
                     AND SPJX.P# = P# ( 'P1' ) ) AS Q
```

Answer: Because duplicate QTY values will now be eliminated before the sum is computed.

7.13.29
```
( SPJX.P#, SPJX.J#,
  SUM ( SPJY WHERE SPJY.P# = SPJX.P#
                 AND   SPJY.J# = SPJX.J#, QTY ) AS Q )
```

7.13.30
```
SPJX.P# WHERE
AVG ( SPJY WHERE SPJY.P# = SPJX.P#
               AND   SPJY.J# = SPJX.J#, QTY ) > QTY ( 350 )
```

7.13.31
```
JX.JNAME WHERE EXISTS SPJX ( SPJX.J# = JX.J# AND
                             SPJX.S# = S# ( 'S1' ) )
```

7.13.32
```
PX.COLOR WHERE EXISTS SPJX ( SPJX.P# = PX.P# AND
                             SPJX.S# = S# ( 'S1' ) )
```

7.13.33
```
SPJX.P# WHERE EXISTS JX ( JX.CITY = 'London' AND
                          JX.J# = SPJX.J# )
```

7.13.34
```
SPJX.J# WHERE EXISTS SPJY ( SPJX.P# = SPJY.P# AND
                            SPJY.S# = S# ( 'S1' ) )
```

7.13.35
```
SPJX.S# WHERE EXISTS SPJY EXISTS SPJZ EXISTS PX
                ( SPJX.P# = SPJY.P# AND
                  SPJY.S# = SPJZ.S# AND
                  SPJZ.P# = PX.P# AND
                  PX.COLOR = COLOR ( 'Red' ) )
```

7.13.36
```
SX.S# WHERE EXISTS SY ( SY.S# = S# ( 'S1' ) AND
                        SX.STATUS < SY.STATUS )
```

7.13.37
```
JX.J# WHERE FORALL JY ( JY.CITY ≥ JX.CITY )
```

Or:
```
JX.J# WHERE JX.CITY = MIN ( JY.CITY )
```

7.13.38
```
SPJX.J# WHERE SPJX.P# = P# ( 'P1' ) AND
              AVG ( SPJY WHERE SPJY.P# = P# ( 'P1' )
                     AND   SPJY.J# = SPJX.J#, QTY ) >
              MAX ( SPJZ.QTY WHERE SPJZ.J# = J# ( 'J1' ) )
```

7.13.39
```
SPJX.S# WHERE SPJX.P# = P# ( 'P1' )
              AND   SPJX.QTY >
                    AVG ( SPJY
                          WHERE SPJY.P# = P# ( 'P1' )
                          AND   SPJY.J# = SPJX.J#, QTY )
```

7.13.40
```
JX.J# WHERE NOT EXISTS SPJX EXISTS SX EXISTS PX
                ( SX.CITY = 'London' AND
                  PX.COLOR = COLOR ( 'Red' ) AND
                  SPJX.S# = SX.S# AND
                  SPJX.P# = PX.P# AND
                  SPJX.J# = JX.J# )
```

7.13.41
```
JX.J# WHERE FORALL SPJY ( IF SPJY.J# = JX.J#
                          THEN SPJY.S# = S# ( 'S1' )
                          END IF )
```

7.13.42
```
PX.P# WHERE FORALL JX
          ( IF JX.CITY = 'London' THEN
            EXISTS SPJY ( SPJY.P# = PX.P# AND
                          SPJY.J# = JX.J# )
            END IF )
```

7.13.43
```
SX.S# WHERE EXISTS PX FORALL JX EXISTS SPJY
                ( SPJY.S# = SX.S# AND
                  SPJY.P# = PX.P# AND
                  SPJY.J# = JX.J# )
```

7.13.44
```
JX.J# WHERE FORALL SPJY ( IF SPJY.S# = S# ( 'S1' ) THEN
                          EXISTS SPJZ
                        ( SPJZ.J# = JX.J# AND
                          SPJZ.P# = SPJY.P# )
                          END IF )
```

7.13.45
```
RANGEVAR VX RANGES OVER
         ( SX.CITY ), ( PX.CITY ), ( JX.CITY ) ;

VX.CITY
```

7.13.46
```
SPJX.P# WHERE EXISTS SX ( SX.S# = SPJX.S# AND
                          SX.CITY = 'London' )
          OR    EXISTS JX ( JX.J# = SPJX.J# AND
                            JX.CITY = 'London' )
```

7.13.47
```
( SX.S#, PX.P# )
WHERE NOT EXISTS SPJX ( SPJX.S# = SX.S# AND
                        SPJX.P# = PX.P# )
```

7.13.48
```
( SX.S# AS XS#, SY.S# AS YS# )
WHERE FORALL PZ
  ( ( IF   EXISTS SPJX ( SPJX.S# = SX.S# AND
                         SPJX.P# = PZ.P# )
      THEN EXISTS SPJY ( SPJY.S# = SY.S# AND
                         SPJY.P# = PZ.P# )
      END IF )
  AND
    ( IF   EXISTS SPJY ( SPJY.S# = SY.S# AND
                         SPJY.P# = PZ.P# )
      THEN EXISTS SPJX ( SPJX.S# = SX.S# AND
                         SPJX.P# = PZ.P# )
      END IF ) )
```

7.13.49
```
( SPJX.S#, SPJX.P#, ( SPJY.J#, SPJY.QTY WHERE
                      SPJY.S# = SPJX.S# AND
                      SPJY.P# = SPJX.P# ) AS JQ )
```

7.13.50 Let R denote the result of evaluating the expression shown in the previous solution. Then:

```
RANGEVAR RX RANGES OVER R ,
RANGEVAR RY RANGES OVER RX.JQ ;

( RX.S#, RX.P#, RY.J#, RY.QTY )
```

We are extending the syntax and semantics of <*range var definition*> slightly. The idea is that the definition of RY depends on that of RX (note that the two definitions are separated by a comma, not a semicolon, and are thereby bundled into a single operation). See reference [3.3] for further discussion.

در باره تئوری وابستگی

I- قواعد (اکسیومهای) آرمسترانگ در مورد FD :

۱- قاعده انعکاس : ▪ if $B \subseteq A$ Then $A \rightarrow B$

و : $A \rightarrow A$

۲- قاعده تعدی (ترانگذری) : ▪ if $A \rightarrow B$ and $B \rightarrow C$ Then $A \rightarrow C$

۳- قاعده افزایش : ▪ if $A \rightarrow B$ Then $(A, C) \rightarrow (B, C)$

۴- قاعده تجزیه : ▪ if $A \rightarrow (B, C)$ Then $A \rightarrow B$ and $A \rightarrow C$

۵- قاعده ترکیب : ▪ if $A \rightarrow B$ and $C \rightarrow D$ Then $(A, C) \rightarrow (B, D)$

۶- قاعده اجتماع : ▪ if $A \rightarrow B$ and $A \rightarrow C$ Then $A \rightarrow (B, C)$

۷- قاعده شبه تعدی : ▪ if $A \rightarrow B$ and $(B, C) \rightarrow D$ Then $(A, C) \rightarrow D$

۸- قاعده یگانگی عمومی (توسط Hugh DARWEN اثبات شده است) :

▪ if $A \rightarrow B$ and $C \rightarrow D$ Then $A \cup (C - B) \rightarrow (B, D)$

نکته : قواعد ۱، ۲ و ۳ استوار و کامل هستند، به این معنا که با داشتن یک مجموعه از وابستگی های تابعی F، تمام وابستگی تابعی منطقاً قابل استنتاج از F، با همین سه قاعده به دست می آیند و هیچ وابستگی تابعی دیگر (که قابل استنتاج از F نباشد) نیز به دست نمی آید.

توجه: سه قاعده اول به آسانی قابل اثبات اند و قواعد دیگر از روی همانها اثبات می شوند.

II - کاربردهای قواعد آمسترانگ

- یافتن بستار یک صفت : A^+
- یافتن بستار یک مجموعه از وابستگی های تابعی : F^+
- یافتن مجموعه کاهش ناپذیر وابستگی های تابعی یک رابطه (مجموعه کمینه)
- یافتن پوش کانونیک
- یافتن کلید (های) رابطه
- اثبات خوب بودن تجزیه رابطه

III - مجموعه کاهش ناپذیر :

۱- هیچ FD در آن افزونه نباشد.

۲- سمت راست هر FD، صفت ساده باشد.

۳- سمت چپ هر FD، کاهش ناپذیر باشد.

IV - پوشش کانونیک :

مجموعه‌ای است از FD ها که در آن

۱- سمت چپ هر FD، کاهش ناپذیر باشد.

۲- در هیچ دو FD، سمت چپ یکسان نباشد.

V : کاربرد بعض مفاهیم :

۱: کاربرد A^+ :

۱- یافتن سوپر کلید و کلید کاندید

۲- اثبات وجود یک FD مفروض در F^+

۲: کاربرد F^+ :

۱- تشخیص معادل بودن دو مجموعه از FD ها : شرط : $G^+ = F^+$

۲- تشخیص افزونه بودن یک FD

۳- تشخیص پوشش یک مجموعه از FD ها : F پوشش G - هرگاه: هر FD در G در F^+ باشد.

۴- در تعریف بعضی از فرمهای نرمال

۵- و بطور غیر مستقیم : در محاسبه مجموعه کاهش ناپذیر و پوشش کانونیک

کنجکاوی: ۱- کاربردهای مجموعه کاهش ناپذیر ، پوشش کانونیک ؟

۲- آیا مجموعه کاهش ناپذیر و پوشش کانونیک یکتا هستند ؟

توجه : تعداد FD های ممکن در یک رابطه درجه n ، برابر است با 2^{n+1} (چرا؟)

توجه : می‌توان FD بین دو صفت را به کمک مکانیسم "اظهار" بیان کرد :

فرض : $R(A, B, C, \ldots)$ و $B \rightarrow C$

```
CREATE ASSERTION CTOBFD
CHECK (NOTEXISTS (SELECT B
                  FROM R
                  GROUP BY B
                  HAVING MAX(C) ≠ MIN(C)));
```

تمرین: الگوریتم یافتن سوپر کلید و الگوریتم یافتن کلید کاندید را بنویسید.

توجه: وابستگی تابعی $B \to C$ را می‌توان با حساب رابطه‌ای هم بیان کرد:

فرض: R_1 و R_2: دو متغیر تاپلی تعریف شده روی رابطه R

```
CONSTRAINT  CTOBFD
  FORALL  R1 ( FORALL R2 (
    IF  R1.B = R2.B  THEN
        R1.C = R2.C )) ;
```

■ تمرین: الگوریتم یافتن A^+ و الگوریتم حذف FD افزونه را بنویسید.

B-1

■ مثال: در رابطه $R(A,B,C,D)$ داریم:

$$F = \{A \to (B,C),\ B \to C,\ A \to B,\ (A,B) \to C,\ (A,C) \to D\}$$

مجموعه کاهش ناپذیر FD ها چنین است: $\{A \to B,\ B \to C,\ A \to D\}$

زیرا:

۱- از $A \to (B,C)$ داریم: $A \to B$ و $A \to C$

۲- از $A \to C$ داریم: $(A,A) \to (A,C)$، پس: $A \to (A,C)$ و با $C \to D$ داریم: $A \to D$

۳- از $A \to C$ بدست می‌آید: $(A,B) \to C$

۴- از $A \to B$ و $B \to C$ داریم: $A \to C$ پس وابستگی‌های تابعی $(A,C) \to D$ و $(A,B) \to C$ و $A \to C$ افزونه‌اند و حذف می‌شوند.

• تمرین: در هر مورد، مجموعه کاهش ناپذیر را بیابید:

$$F_1: \left\{\begin{array}{l} B \to D \\ E \to C \\ (A,C) \to D \\ (C,D) \to A \\ (B,E) \to A \end{array}\right\} \qquad F_2: \left\{\begin{array}{l} A \to (C,D,E) \\ B \to (C,E) \\ (A,D) \to E \\ (C,D) \to F \\ (B,D) \to A \\ (C,E,D) \to (A,B,D) \end{array}\right\}$$

• تمرین: در رابطه $R(A,B,C,D)$ داریم: $(A,D) \to C$ و $(C,D) \to B$ و $A \to D$. مطلوبست محاسبه $(A,D)^+$. بررسی شود آیا (A,D) سوپرکلید است؟

توجه: وابستگی تابعی $B \to C$ را می‌توان با حساب رابطه‌ای هم بیان کرد:

فرض: R_1 و R_2: دو متغیر تاپلی تعریف شده روی رابطه R

```
CONSTRAINT  CTOBFD
    FORALL  R1 ( FORALL R2 (
        IF  R1·B = R2·B  THEN
            R1·C = R2·C )) ;
```

■ تمرین: الگوریتم یافتن A^+ و الگوریتم حذف FD افزونه را بنویسید.

B-1

■ مثال: در رابطه $R(A,B,C,D)$ داریم:

$$F = \{A \to (B,C),\ B \to C,\ A \to B,\ (A,B) \to C,\ (A,C) \to D\}$$

مجموعه کاهش ناپذیر FD ها چنین است: $\{A \to B,\ B \to C,\ A \to D\}$

زیرا:

۱- از $A \to (B,C)$ داریم: $A \to B$ و $A \to C$

۲- از $A \to C$ داریم: $(A,A) \to (A,C)$، پس: $A \to (A,C)$ و با $C \to D$ داریم: $A \to D$

۳- از $A \to C$ بدست می‌آید: $(A,B) \to C$

۴- از $A \to B$ و $B \to C$ داریم: $A \to C$ پس وابستگی‌های تابعی $(A,C) \to D$ و $(A,B) \to C$ و $A \to C$ افزونه‌اند و حذف می‌شوند.

• تمرین: در هر مورد، مجموعه کاهش ناپذیر را بیابید:

$$F_1: \left\{\begin{array}{l} B \to D \\ E \to C \\ (A,C) \to D \\ (C,D) \to A \\ (B,E) \to A \end{array}\right\} \qquad F_2: \left\{\begin{array}{l} A \to (C,D,E) \\ B \to (C,E) \\ (A,D) \to E \\ (C,D) \to F \\ (B,D) \to A \\ (C,E,D) \to (A,B,D) \end{array}\right\}$$

• تمرین: در رابطه $R(A,B,C,D)$ داریم: $(A,D) \to C$ و $(C,D) \to B$ و $A \to D$. مطلوبست محاسبه $(A,D)^+$. بررسی شود آیا (A,D) سوپرکلید است؟

توجه: وابستگی تابعی $B \to C$ را می‌توان با حساب رابطه‌ای هم بیان کرد:

فرض: R_1 و R_2: دو متغیر تاپلی تعریف شده روی رابطه R

```
CONSTRAINT  CTOBFD
  FORALL  R1 ( FORALL R2 (
    IF  R1·B = R2·B  THEN
        R1·C = R2·C )) ;
```

■ تمرین: الگوریتم یافتن A^+ و الگوریتم حذف FD افزونه را بنویسید.

B-1

■ مثال: در رابطه $R(A,B,C,D)$ داریم:

$$F = \{A \to (B,C),\ B \to C,\ A \to B,\ (A,B) \to C,\ (A,C) \to D\}$$

مجموعه کاهش ناپذیر FD ها چنین است: $\{A \to B,\ B \to C,\ A \to D\}$

زیرا:

۱- از $A \to (B,C)$ داریم: $A \to B$ و $A \to C$

۲- از $A \to C$ داریم: $(A,A) \to (A,C)$، پس: $A \to (A,C)$ و با $C \to D$ داریم: $A \to D$

۳- از $A \to C$ بدست می‌آید: $(A,B) \to C$

۴- از $A \to B$ و $B \to C$ داریم: $A \to C$ پس وابستگی‌های تابعی $(A,C) \to D$ و $(A,B) \to C$ و $A \to C$ افزونه‌اند و حذف می‌شوند.

• تمرین: در هر مورد، مجموعه کاهش ناپذیر را بیابید:

$$F_1: \left\{\begin{array}{l} B \to D \\ E \to C \\ (A,C) \to D \\ (C,D) \to A \\ (B,E) \to A \end{array}\right\} \qquad F_2: \left\{\begin{array}{l} A \to (C,D,E) \\ B \to (C,E) \\ (A,D) \to E \\ (C,D) \to F \\ (B,D) \to A \\ (C,E,D) \to (A,B,D) \end{array}\right\}$$

• تمرین: در رابطه $R(A,B,C,D)$ داریم: $(A,D) \to C$ و $(C,D) \to B$ و $A \to D$. مطلوبست محاسبه $(A,D)^+$. بررسی شود آیا (A,D) سوپرکلید است؟

قواعد استنتاج آرمسترانگ برای MVD

در رابطه $R(X, Y, Z, \ldots)$:

1) $Y \subset X \Rightarrow X \rightarrow\rightarrow Y$

2) $X \rightarrow\rightarrow Y \,,\, Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow Z - Y$

3) $X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow (R(H) - (X, Y))$

4) $X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$

5) $X \rightarrow\rightarrow Y \,,\, X \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Y, Z)$

6) $X \rightarrow\rightarrow Y \,,\, (Y, Z) \rightarrow\rightarrow W \Rightarrow (X, Z) \rightarrow\rightarrow (W - (Y, Z))$

7) $X \rightarrow\rightarrow Y \,,\, Z \subseteq W \Rightarrow (X, W) \rightarrow\rightarrow (Y, Z)$

8) $X \rightarrow\rightarrow (Y, Z) \Rightarrow X \rightarrow\rightarrow Y \cap Z \,,\, X \rightarrow\rightarrow Y - Z \,,\, X \rightarrow\rightarrow Z - Y$

9) $X \rightarrow\rightarrow Y \,,\, Z \rightarrow W \,,\, W \subset Y \,,\, Y \cap Z = \emptyset \Rightarrow X \rightarrow W$

تعریف فرمال MVD :

در رابطه $R(X, Y, Z)$ ، وابستگی تابعی چندمقداری $X \rightarrow\rightarrow Y$ وجود دارد اگر : در هر مقدار از رابطه R (در گسترده R در تمام لحظات) ، به ازاء تمام جفت تاپلهای t_1 و t_2 ، چنانچه $t_1[X] = t_2[X]$ باشد ، آنگاه تاپلهای t_3 و t_4 نیز وجود داشته باشند به نحوی که :

$$\begin{pmatrix} t_1[X] = t_2[X] = t_3[X] = t_4[X] \\ t_3[Y] = t_1[Y] \\ t_3[R(H) - Y] = t_2[R(H) - Y] \\ t_4[Y] = t_2[Y] \\ t_4[R(H) - Y] = t_1[R(H) - Y] \end{pmatrix}$$

دو تعریف دیگر از رابطه 4NF

- رابطه R با مجموعه وابستگیهای تابعی و چندمقداری F ، در 4NF است اگر برای هر وابستگی تابعی در F^+ به صورت $X \rightarrow\rightarrow Y$ $(Y \subseteq R(H) \,,\, X \subseteq R(H))$ حداقل یکی از دو وضع زیر برقرار باشد [RICA 90] :
  - $X \rightarrow\rightarrow Y$ یک وابستگی تابعی چندمقداری نامهم باشد .
  - X سوپرکلید R باشد .
- رابطه R در 4NF است اگر درصورت وجود وابستگی چندمقداری $X \rightarrow\rightarrow Y$ در F^+ ، $(Y \subseteq R(H) \,,\, X \subseteq R(H))$ ، X کلید کاندید R باشد [SIMO 95]

DT-4

B-1

۹- دلایل بروز افزونگی

در سیستم‌های ISR در فضای عام

(از جمله در سیستم‌های پایگاهی)

۱- ماهیت داده‌های سازمان (افزونگی طبیعی)

۲- تکنیک استفاده شده در تأمین استراتژی دستیابی کارآتر (مثلاً نمایه‌سازی ...)

۳- مشی طراحی و تولید سیستم کاربردی (مثلاً در مشی فایلینگ: ناپایگاهی)

۴- ماهیت ساختار داده‌ای (مثلاً کلید خارجی در ساختار داده رابطه‌ای ...)

۵- طراحی بد

۶- اضافه کردن صفت (صفات) به رابطه برای افزایش سرعت اجرای برنامه

۷- تولید نسخه(های) دیگر از داده (به دلیل عملیاتی یا بردارشی ...)

۸- وابستگی‌های بین صفات

۹- کاهش درجه نرمالیتی رابطه

۱۰- تولید نسخه‌ای از داده برای تخصیص به سایت‌ها (در سیستم توزیع شده)

۱۱- تکنیک طراحی (مثلاً پذیرش افزونگی برای کاهش پیچیدگی، ...)

۱۲- تکرار ذخیره‌سازی بخشی از داده‌ها (مثلاً بخشی از یک یا بیش از یک رابطه)

برای سرعت اجرای تراکنش‌ها در بازیابی ... (مثل دید ذخیره شده)

۱۳- نرمالترسازی رابطه‌ها (تجزیه عمودی رابطه به دو یا بیش از دو رابطه در صورت انجام تجزیه خوب)

توجه: معایب افزونگی: ✓ مصرف حافظه بیشتر

✓ فزونکاری (سربار) سیستم برای انجام بهنگام‌سازی منتشر شونده

مزیت افزونگی: ✓ افزایش کارایی سیستم در بازیابی

✓ امکان لازم برای ترمیم، ایمنی و ...

R-1 کنجکاوی: دلایل دیگر؟

مأخذ: مفاهیم بنیادی پایگاه داده‌ها

B - course(1)

تمرینات تکمیلی در : SQL

۱- صورت پرسش‌ها (درس دوم) :

Query Exercises

The remaining exercises are all based on the suppliers-parts-projects database (see Fig. 4.5 in the "Exercises" section in Chapter 4 and the answer to Exercise 5.4 in Chapter 5). In each case you are asked to write a relational algebra expression for the indicated query. (By way of an interesting variation, you might like to try looking at some of the answers first and stating what the given expression means in natural language.) For convenience we show the structure of the database (in outline) below:

```
S    { S#, SNAME, STATUS, CITY }
     PRIMARY KEY { S# }
P    { P#, PNAME, COLOR, WEIGHT, CITY }
     PRIMARY KEY { P# }
J    { J#, JNAME, CITY }
     PRIMARY KEY { J# }
SPJ  { S#, P#, J#, QTY }
     PRIMARY KEY { S#, P#, J# }
     FOREIGN KEY { S# } REFERENCES S
     FOREIGN KEY { P# } REFERENCES P
     FOREIGN KEY { J# } REFERENCES J
```

6.13 Get full details of all projects.

6.14 Get full details of all projects in London.

6.15 Get supplier numbers for suppliers who supply project J1.

6.16 Get all shipments where the quantity is in the range 300 to 750 inclusive.

6.17 Get all part-color/part-city combinations. *Note:* Here and subsequently, the term "all" is to be taken to mean "all currently represented in the database," not "all possible."

6.18 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are all colocated (i.e., all in the same city).

6.19 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are not all colocated.

6.20 Get all supplier-number/part-number/project-number triples such that no two of the indicated supplier, part, and project are colocated.

6.21 Get full details for parts supplied by a supplier in London.

6.22 Get part numbers for parts supplied by a supplier in London to a project in London.

6.23 Get all pairs of city names such that a supplier in the first city supplies a project in the second city.

6.24 Get part numbers for parts supplied to any project by a supplier in the same city as that project.

6.25 Get project numbers for projects supplied by at least one supplier not in the same city.

6.26 Get all pairs of part numbers such that some supplier supplies both the indicated parts.

6.27 Get the total number of projects supplied by supplier S1.

6.28 Get the total quantity of part P1 supplied by supplier S1.

6.29 For each part being supplied to a project, get the part number, the project number, and the corresponding total quantity.

6.30 Get part numbers of parts supplied to some project in an average quantity of more than 350.

6.31 Get project names for projects supplied by supplier S1.

6.32 Get colors of parts supplied by supplier S1.

6.33 Get part numbers for parts supplied to any project in London.

6.34 Get project numbers for projects using at least one part available from supplier S1.

6.35 Get supplier numbers for suppliers supplying at least one part supplied by at least one supplier who supplies at least one red part.

6.36 Get supplier numbers for suppliers with a status lower than that of supplier S1.

6.37 Get project numbers for projects whose city is first in the alphabetic list of such cities.

6.38 Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.

6.39 Get supplier numbers for suppliers supplying some project with part P1 in a quantity greater than the average shipment quantity of part P1 for that project.

6.40 Get project numbers for projects not supplied with any red part by any London supplier.

6.41 Get project numbers for projects supplied entirely by supplier S1.

6.42 Get part numbers for parts supplied to all projects in London.

6.43 Get supplier numbers for suppliers who supply the same part to all projects.

6.44 Get project numbers for projects supplied with at least all parts available from supplier S1.

6.45 Get all cities in which at least one supplier, part, or project is located.

6.46 Get part numbers for parts that are supplied either by a London supplier or to a London project.

6.47 Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.

6.48 Get all pairs of supplier numbers, S*x* and S*y* say, such that S*x* and S*y* supply exactly the same set of parts each. (Thanks to a correspondent, Fatma Mili of Oakland University, Rochester, Michigan, for this problem. For simplicity, you might want to use the original suppliers and parts database for this exercise, instead of the expanded suppliers-parts-projects database.)

6.49 Get a "grouped" version of all shipments showing, for each supplier-number/part-number pair, the corresponding project numbers and quantities in the form of a binary relation.

6.50 Get an "ungrouped" version of the relation produced in Exercise 6.49.

: SQL باب

7.15 We have numbered the following solutions as 7.15.*n*, where 6.*n* is the number of the original exercise in Chapter 6.

7.15.13
```
SELECT *
FROM   J ;
```

Or simply:

```
TABLE J ;
```

7.15.14
```
SELECT J.*
FROM   J
WHERE  J.CITY = 'London' ;
```

7.15.15
```
SELECT DISTINCT SPJ.S#
FROM   SPJ
WHERE  SPJ.J# = 'J1' ;
```

7.15.16
```
SELECT SPJ.*
FROM   SPJ
WHERE  SPJ.QTY >= 300
AND    SPJ.QTY <= 750 ;
```

7.15.17
```
SELECT DISTINCT P.COLOR, P.CITY
FROM   P ;
```

7.15.18
```
SELECT S.S#, P.P#, J.J#
FROM   S, P, J
WHERE  S.CITY = P.CITY
AND    P.CITY = J.CITY ;
```

7.15.19
```
SELECT S.S#, P.P#, J.J#
FROM   S, P, J
WHERE  NOT ( S.CITY = P.CITY AND
             P.CITY = J.CITY ) ;
```

7.15.20
```
SELECT S.S#, P.P#, J.J#
FROM   S, P, J
WHERE  S.CITY <> P.CITY
AND    P.CITY <> J.CITY
AND    J.CITY <> P.CITY ;
```

7.15.21
```
SELECT DISTINCT SPJ.P#
FROM   SPJ
WHERE  ( SELECT S.CITY
         FROM   S
         WHERE  S.S# = SPJ.S# ) = 'London' ;
```

Select P.* from S, P, SPJ
where SPJ.S# = S.S#
And SPJ.P# = P.P#
And S.City = 'LONDON'

7.15.22
```
SELECT DISTINCT SPJ.P#
FROM   SPJ
WHERE  ( SELECT S.CITY
         FROM   S
         WHERE  S.S# = SPJ.S# ) = 'London'
AND    ( SELECT J.CITY
         FROM   J
         WHERE  J.J# = SPJ.J# ) = 'London' ;
```

7.15.23
```
SELECT DISTINCT S.CITY AS SCITY, J.CITY AS JCITY
FROM   S, J
WHERE  EXISTS
       ( SELECT *
         FROM   SPJ
         WHERE  SPJ.S# = S.S#
         AND    SPJ.J# = J.J# ) ;
```

7.15.24
```
SELECT DISTINCT SPJ.P#
FROM   SPJ
WHERE  ( SELECT S.CITY
         FROM   S
         WHERE  S.S# = SPJ.S# ) =
       ( SELECT J.CITY
         FROM   J
         WHERE  J.J# = SPJ.J# ) ;
```

7.15.25
```
SELECT DISTINCT SPJ.J#
FROM   SPJ
WHERE  ( SELECT S.CITY
         FROM   S
         WHERE  S.S# = SPJ.S# ) <>
       ( SELECT J.CITY
         FROM   J
         WHERE  J.J# = SPJ.J# ) ;
```

7.15.26
```
SELECT DISTINCT SPJX.P# AS PA, SPJY.P# AS PB
FROM   SPJ AS SPJX, SPJ AS SPJY
WHERE  SPJX.S# = SPJY.S#
AND    SPJX.P# < SPJY.P# ;
```

7.15.27
```
SELECT COUNT ( DISTINCT SPJ.J# ) AS N
FROM   SPJ
WHERE  SPJ.S# = 'S1' ;
```

7.15.28
```
SELECT SUM ( SPJ.QTY ) AS X
FROM   SPJ
WHERE  SPJ.S# = 'S1'
AND    SPJ.P# = 'P1' ;
```

7.15.29
```
SELECT SPJ.P#, SPJ.J#, SUM ( SPJ.QTY ) AS Y
FROM   SPJ
GROUP  BY SPJ.P#, SPJ.J# ;
```

7.15.30
```
SELECT DISTINCT SPJ.P#
FROM   SPJ
GROUP  BY SPJ.P#, SPJ.J#
HAVING AVG ( SPJ.QTY ) > 350 ;
```

7.15.31
```
SELECT DISTINCT J.JNAME
FROM   J, SPJ
WHERE  J.J# = SPJ.J#
AND    SPJ.S# = 'S1' ;
```

7.15.32
```
SELECT DISTINCT P.COLOR
FROM   P, SPJ
WHERE  P.P# = SPJ.P#
AND    SPJ.S# = 'S1' ;
```

7.15.33
```
SELECT DISTINCT SPJ.P#
FROM   SPJ, J
WHERE  SPJ.J# = J.J#
AND    J.CITY = 'London' ;
```

7.15.34
```
SELECT DISTINCT SPJX.J#
FROM   SPJ AS SPJX, SPJ AS SPJY
WHERE  SPJX.P# = SPJY.P#
AND    SPJY.S# = 'S1' ;
```

7.15.35
```
SELECT DISTINCT SPJX.S#
FROM   SPJ AS SPJX, SPJ AS SPJY, SPJ AS SPJZ
WHERE  SPJX.P# = SPJY.P#
AND    SPJY.S# = SPJZ.S#
AND  ( SELECT P.COLOR
       FROM   P
       WHERE  P.P# = SPJZ.P# ) = 'Red' ;
```

7.15.36
```
SELECT S.S#
FROM   S
WHERE  S.STATUS < ( SELECT S.STATUS
                    FROM   S
                    WHERE  S.S# = 'S1' ) ;
```

7.15.37
```
SELECT J.J#
FROM   J
WHERE  J.CITY = ( SELECT MIN ( J.CITY )
                  FROM   J ) ;
```

7.15.38
```
SELECT DISTINCT SPJX.J#
FROM   SPJ AS SPJX
WHERE  SPJX.P# = 'P1'
AND  ( SELECT AVG ( SPJY.QTY )
       FROM   SPJ AS SPJY
       WHERE  SPJY.J# = SPJX.J#
       AND    SPJY.P# = 'P1' ) >
     ( SELECT MAX ( SPJZ.QTY )
       FROM   SPJ AS SPJZ
       WHERE  SPJZ.J# = 'J1' ) ;
```

7.15.39
```
SELECT DISTINCT SPJX.S#
FROM   SPJ AS SPJX
WHERE  SPJX.P# = 'P1'
AND    SPJX.QTY > ( SELECT AVG ( SPJY.QTY )
                    FROM   SPJ AS SPJY
                    WHERE  SPJY.P# = 'P1'
                    AND    SPJY.J# = SPJX.J# ) ;
```

7.15.40
```
SELECT J.J#
FROM   J
WHERE  NOT EXISTS
     ( SELECT *
       FROM   SPJ, P, S
       WHERE  SPJ.J# = J.J#
       AND    SPJ.P# = P.P#
       AND    SPJ.S# = S.S#
       AND    P.COLOR = 'Red'
       AND    S.CITY = 'London' ) ;
```

7.15.41
```
SELECT J.J#
FROM   J
WHERE  NOT EXISTS
     ( SELECT *
       FROM   SPJ
       WHERE  SPJ.J# = J.J#
       AND    NOT ( SPJ.S# = 'S1' ) ) ;
```

7.15.42
```
SELECT P.P#
FROM   P
WHERE  NOT EXISTS
     ( SELECT *
       FROM   J
       WHERE  J.CITY = 'London'
       AND    NOT EXISTS
            ( SELECT *
              FROM   SPJ
              WHERE  SPJ.P# = P.P#
              AND    SPJ.J# = J.J# ) ) ;
```

7.15.43
```
SELECT S.S#
FROM   S
WHERE  EXISTS
     ( SELECT *
       FROM   P
       WHERE  NOT EXISTS
            ( SELECT *
              FROM   J
              WHERE  NOT EXISTS
                   ( SELECT *
                     FROM   SPJ
                     WHERE  SPJ.S# = S.S#
                     AND    SPJ.P# = P.P#
                     AND    SPJ.J# = J.J# ) ) ) ;
```

7.15.44
```
SELECT J.J#
FROM   J
WHERE  NOT EXISTS
     ( SELECT *
       FROM   SPJ AS SPJX
       WHERE  SPJX.S# = 'S1'
       AND    NOT EXISTS
            ( SELECT *
              FROM   SPJ AS SPJY
              WHERE  SPJY.P# = SPJX.P#
              AND    SPJY.J# = J.J# ) ) ;
```

7.15.45
```
SELECT S.CITY FROM S
UNION
SELECT P.CITY FROM P
UNION
SELECT J.CITY FROM J ;
```

7.15.46
```
SELECT DISTINCT SPJ.P#
FROM   SPJ
WHERE  ( SELECT S.CITY
         FROM   S
         WHERE  S.S# = SPJ.S# ) = 'London'
OR     ( SELECT J.CITY
         FROM   J
         WHERE  J.J# = SPJ.J# ) = 'London' ;
```

7.15.47
```
SELECT S.S#, P.P#
FROM   S, P
EXCEPT
SELECT SPJ.S#, SPJ.P#
FROM   SPJ ;
```

7.15.48 Solution omitted.

7.15.49–7.15.50 Cannot be done.

B

View updating in SQL-standard

بهنگام سازی دیدها در SQL استاندارد

- اصطلاح بهنگام سازی در اینجا یعنی: عملیات درج، حذف و تغییر (بهنگام سازی در معنای خاص)
- در SQL استاندارد، موضوع دیدهای قابل عملیات بهنگام سازی (دیدهای پذیرا) چندان روشن نیست. اما گذشته از جزئیات می توان گفت که دیدهایی که تمام شرایط زیر را داشته باشند، قطعاً قابل عملیات بهنگام سازی هستند (توجه! ممکن است برخی دیگر از دیدها هم قابل عملیات بهنگام سازی باشند):

۱- عبارت تعریف کننده دید، یک عبارت SELECT ساده باشد (یعنی شامل عملگرهای JOIN، UNION، INTERSECT و EXCEPT نباشد).

۲- در عبارت SELECT گزینه DISTINCT وجود نداشته باشد.

۳- در کلاز FROM عبارت SELECT، فقط یک جدول وجود داشته باشد.

۴- جدول قید شده در کلاز FROM، یک جدول مبنا یا یک دید قابل بهنگام سازی باشد.

۵- در لیست نام ستون‌ها در عبارت SELECT، ستون‌های مورد نظر باید در جدول مبنا متناظر داشته باشند و به یک ستون از جدول مبنا بیش از یک بار ارجاع وجود نداشته باشد.

۶- در عبارت SELECT، کلاز GROUP BY و/یا کلاز HAVING وجود نداشته باشد.

۷- کلاز WHERE در عبارت SELECT حاوی کلاز FROM نباشد به گونه‌ای که در آن به همان جدولی ارجاع داده شده باشد که در کلاز FROM ذکر شده در شرط ۴.

نتیجه اینکه عملاً دیدهایی که یک زیرمجموعه افقی - عمودی دارای کلید از یک جدول مبنا (یا از یک دید قابل بهنگام سازی) باشند، قطعاً قابل بهنگام سازی هستند (توجه! به شرط رعایت محدودیت‌های جامعیتی، مثل یکتایی کلید، تمامیت ارجاعی و ...)

* [DATE 03] از نقطه نظر "جامعیت داده‌ها" در پایگاه داده‌های رابطه‌ای

عملیات بازگشتی در SQL

۱- آیا می‌توان با SQL عملیات بازگشتی را برنامه‌سازی کرد؟

پاسخ: بله.

■ مثال: جدول COPRECO(COID, PRECOID) را در نظر می‌گیریم (شماره درس، شماره درس پیشنیاز). این جدول ارتباط "پیشنیازی" بین نوع موجودیت درس و خودش را نشان می‌دهد. سؤال کاربر چنین است:

Q_1: شماره درس تمام درس‌های پیشنیاز درس شماره 'COM222' را بدهید.

قطعه "برنامه" زیر برای این نیاز پاسخ می‌دهد (در SQL استاندارد):

```
WITH RECURSIVE
  PRECORS(COID, PRECOID) AS
  (SELECT COID, PRECOID
    FROM COPRECO
      WHERE COID='COM222'
    UNION ALL
   SELECT COPRECO(COID), COPRECO(PRECOID)
    FROM PRECORS, COPRECO
      WHERE COPRECO.COID=PRECORS(PRECOID))
   SELECT * FROM PRECORS
    ORDER BY COID, PRECOID;
```

تمرین ۱: این قطعه "برنامه" را را با دست بگردانید.

تمرین ۲: جدول PP(MAJP#, MINP#, Q) مفروض است. (MAJP#: شماره قطعه اصلی، MINP#: شماره قطعه فرعی، Q: تعداد)

درخواست زیر را SQL پاسخ دهید:

Q: شماره تمام قطعات تشکیل دهنده قطعه شماره 'P7' را بدهید.

* مأخذ مثال: "مفاهیم بنیادی پایگاه داده‌ها"
* مأخذ مثال تمرین ۲: UPDATE 03

در SQL استاندارد امکان انجام عملیات بازگشتی وجود دارد. برای این منظور کلاز WITH RECURSIVE بکار می‌رود. با این کلاز یک دید (دید موقت) تعریف می‌شود. در تعریف این دید، از خود این دید هم به نحوی استفاده می‌شود. در واقع یک نوع دید بازگشتی باید تعریف شود. این دید به صورت اجتماع دو پرسش تعریف می‌شود: یک پرسش مبنا[1] که نابازگشتی است و یک پرسش بازگشتی[2] که در آن از دید بازگشتی استفاده می‌شود. در زیر ساختار کلی بازیابی از یک جدول با عملیات بازگشتی دیده می‌شود: [SILBO 08]

```
WITH RECURSIVE Temp1 ( Col(s)) AS
    ( SELECT col(s)
        FROM table
    UNION
      SELECT table.col , Temp1.col
        FROM table , Temp1
          WHERE ----- )
    SELECT *  FROM Temp1 ;
```

در اینجا، پرسش مبنا، SELECT از جدول table است و پرسش بازگشتی، SELECT از پیوند دو جدول table و Temp1 است.

مثال ۶:

1- Base query 2- Recursive query