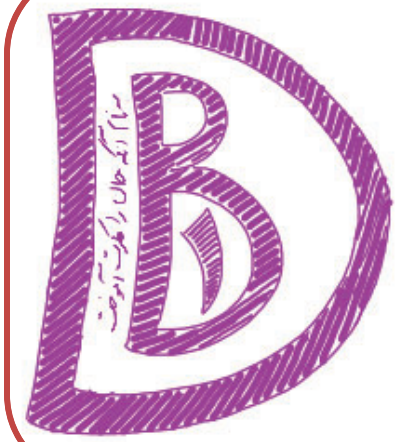
به نام آنکه جان را فکرت آموخت



معرفی درس:

طراحی پایگاه داده‌ها (۴۰۳۸۴)

مرتضی امینی

نیمسال دوم ۹۴-۹۵

تولید سیستم پایگاهی

- **سوال:** برای تولید یک سیستم پایگاهی در یک محیط عملیاتی چه باید کرد؟

سرفصل‌های درس

۶ معرفی درس طراحی پایگاه داده‌ها

۱- کلیات

- تعریف پایگاه داده‌ها، مشی فایلینگ و مشی پایگاهی، عناصر محیط پایگاه داده، انواع معماری سیستم پایگاهی

۲- مدلسازی معنایی داده‌ها با روش ER و EER

- نمودار ER و اجزای آن، انواع دام‌ها، تکنیک‌های تخصیص، تعمیم، تجزیه، ترکیب و تجمیع، ویژگی‌های روش مدلسازی معنایی

۳- آشنایی با ساختار داده‌ای جدولی (رابطه‌ای)

- ساختار جدولی و اجزای آن، پایگاه داده جدولی و طراحی آن، زبان پایگاه داده جدولی (SQL)

۴- معماری سه سطحی پایگاه (پیشنهادی ANSI)

- دید (نمای) ادراکی، دید داخلی، دید خارجی، تبدیلات بین سطوح، عملیات از دید خارجی و مشکلات آن، استقلال داده‌ای فیزیکی و منطقی

۵- سیستم مدیریت پایگاه داده‌ها (DBMS)

- ریزفعالیت‌های ایجاد سیستم پایگاهی، مزایا و معایب تکنولوژی پایگاهی، وظایف، اجزا و رده‌بندی سمپادها، تیم مدیریت پایگاه داده‌ها (DBA)

سرفصل‌های درس (ادامه)

۶- مفاهیم اساسی مدل داده رابطه‌ای

- رابطه و مفاهیم مربوطه، میدان (دامنه)، انواع رابطه، رابطه‌های نرمال و غیرنرمال، انواع کلید در مدل رابطه‌ای

۷- اصول طراحی پایگاه داده‌های رابطه‌ای به روش بالا به پایین

- تکنیک‌های تبدیل مدلسازی معنایی به طراحی منطقی

۸- اصول طراحی پایگاه داده‌های رابطه‌ای به روش سنتز

- روش سنتز (نرمال‌ترسازی رابطه‌ها)، مفاهیمی از تئوری وابستگی، شرح فرم‌های نرمال، تجزیه مطلوب

۹- جامعیت در مدل رابطه‌ای

- قواعد کاربری، مکانیزم‌های اِعمال قواعد جامعیت کاربری، قواعد جامعیت موجودیتی و ارجاعی (C1 و C2)

۱۰- عملیات در پایگاه رابطه‌ای

- جبر رابطه‌ای، حساب رابطه‌ای

نکته: یادگیری زبان SQL به عهده دانشجو است.

(در کلاس به شکل مختصر معرفی می‌شود)

مراجع درس

- **مفاهیم بنیادی پایگاه داده‌ها** نوشته سیدمحمدتقی روحانی رانکوهی، ویراست چهارم، ۱۳۹۰.
- **An Introduction to Database Systems**, By C.J. Date, 8th Edition, 2003.
- **Fundamental of Database Systems**, By R. Elmasri, 7th Edition, 2015.
- **Database Systems**, By T. Connolly and C. Begg, 6th Edition, 2014.
- **Database Management Systems**, By R. Ramakrishnan and J. Gehrke, 3nd Edition, 2002.
- **Database System Concepts**, By A. Silberschartz, H.F. Korth and S. Sudarshan, 6th Edition, 2010.

به نام آنکه جان را فکرت آموخت

بخش اول :
مقدمه

مرتضی امینی

نیمسال دوم ۹۴-۹۵

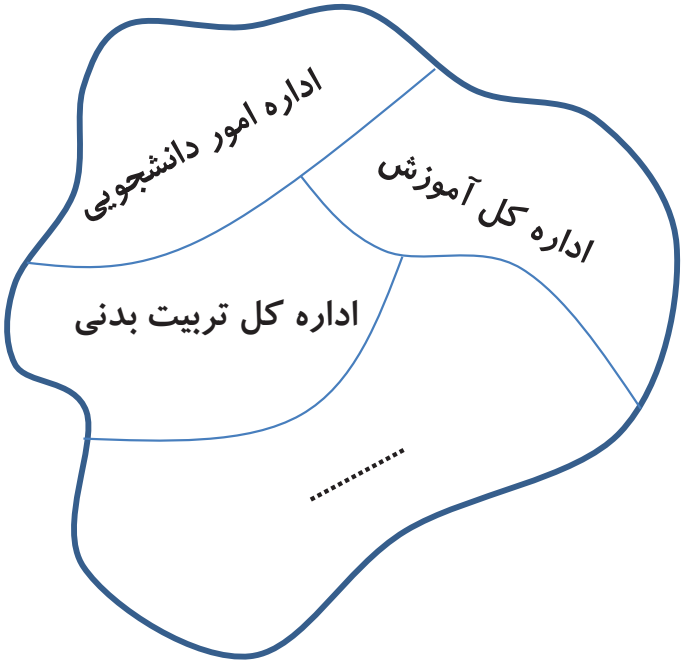
(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)

مثال کاربردی

محیط عملیاتی: دانشگاه

بخشی از جهان واقعی که قصد ایجاد سیستم برای آن را داریم.

Micro Real World (خرد جهان واقع)
Mini World
Universe of Discourse (جهان مطرح)



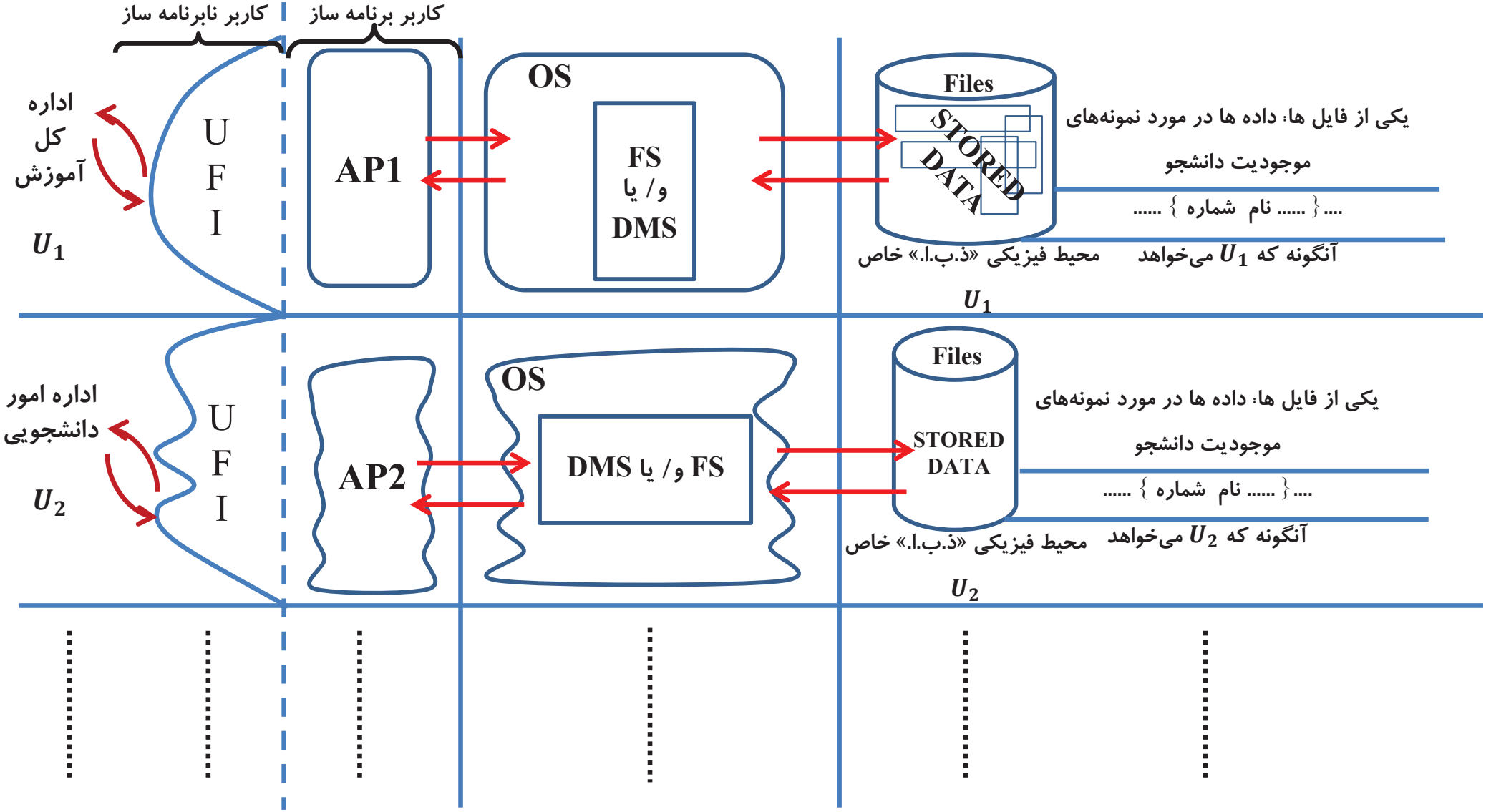
- نکته: هر محیط از تعدادی زیر محیط تشکیل شده است.
- در هر محیط مجموعه‌ای از نوع موجودیت‌ها وجود دارند که نیازهای کاربران ناظر به آنهاست (یعنی داده‌ای پردازشی به داده‌هایی در مورد آنها نیاز دارند).

- **نکته:** زیرمحیط های یک محیط *معمولا* با هم اشتراک دارند در نوع موجودیت‌ها (Entity Type یا Object Type)
  - مثال: در محیط دانشگاه دانشجو، استاد، درس، کلاس، و ...
  - مثال: نوع موجودیت دانشجو در هر سه زیر محیط مطرح است.

- **مسئله (خواسته): ایجاد سیستم(های) کاربردی برای این زیر محیط‌ها**
  - برای این منظور در اساس دو مَشی-روش (approach) وجود دارد.
    - مشی فایلینگ [سنتی یا کلاسیک] یا ناپایگاهی
    - مشی پایگاهی Database Approach

    یعنی ممکن است مشی‌های بینابینی نیز وجود داشته باشد.

نمایش شماتیک مشی فایلینگ



برخی از معایب مشی فایلینگ:

- وجود سیستم های نامجتمع در یک سازمان [محیط] و نامرتبط به هم
- عدم وجود یک سیستم کنترل متمرکز روی کل داده‌های سازمان
- وجود افزونگی زیاد
- خطر بروز ناسازگاری داده ها (Data Inconsistency) ← کنجکاوی: جنبه های بروز ناسازگاری کدامند؟
- عدم امکان اعمال ضوابط حفظ امنیت داده‌ها (Data Security)
- عدم امکان اشتراکی شدن داده ها (Data Sharing) [یا در حداقل و یا با دشواری]
- مصرف نابهینه سخت افزار (به ویژه سخت افزار ذخیره‌ساز)
- وابسته بودن برنامه ها به جنبه های فایلینگ محیط ذخیره‌سازی، به گونه‌ای که اگر قرار باشد در فایلینگ تغییراتی ایجاد شود، برنامه ها هم متناسبا باید تغییر یابد. (به طور مثال فرمت ساختار یا نحوه دسترسی (Access Strategy) را تغییر دهیم)

- **افزونگی در معنای گسترده** (یعنی برون‌فایلی- در مباحث پایگاه داده)
  - عبارت است از تکرار ذخیره‌سازی داده‌ها در مورد نمونه‌های یک یا بیش از یک نوع موجودیت از یک محیط.
    - این نوع افزونگی نه از نوع طبیعی و نه از نوع تکنیکی است بلکه ناشی از رهیافت انتخاب شده برای طراحی و تولید سیستم‌های کاربردی است.
    - به طور مثال تکرار اطلاعات دانشجویان در دو زیرسیستم اداره کل آموزش و زیرسیستم اداره امور دانشجویی.
  - **نکته:** افزونگی از نوع طبیعی و تکنیکی در پایگاه داده هم می‌تواند وجود داشته باشد.

دلایل بروز افزونگی در سیستم های ISR به ویژه سیستم های پایگاهی کدامند؟

مثال مقدماتی : اما در مشی پایگاهی

❑ **کارهای لازم در انجام یک «پروژه پایگاهی»:** (فعلا نه در جزئیات)

به منظور ایجاد یک سیستم یکپارچه مجتمع

۱- مطالعه و شناخت محیط

۲- انجام مهندسی نیاز ها Requirement Engineering ← ۱- تشخیص نیاز های داده ای ۲- تشخیص نیاز های پردازشی ۳- مستندسازی نیاز ها ۴- دریافت تایید سازمان

۳- تعیین مشخصات سیستم یکپارچه

[۴- انتخاب (حداقل) یک پیکربندی H/S]

[۵- انتخاب (حداقل) یک DBMS (یک تصمیم گیری حیاتی است)]

۶- مدلسازی معنایی داده ها (data semantic modeling) ← در مشی فایلینگ انجام نمی‌شود.

۷- طر احی منطقی پایگاه داده Logical Data Base Design

۸- طراحی فیزیکی پایگاه داده Physical Data Base Design

۹- طراحی UFI ها

۱۰- طراحی AP ها [ضمن تعریف تراکنش ها (Transactions)]

ادامه مثال مقدماتی (مشی پایگاهی)

ادامه:...

۱۱- تولید برنامه‌های تعریف (ایجاد) و کنترل DB

۱۲- تولید برنامه‌های عملیات در داده‌ها (پرداز‌ش داده‌ها)

۱۳- ایجاد محیط فیزیکی «ذ.ب.ا.» با داده های تستی و رفع اشکال ها (تست مرحله اول)

۱۴- ایجاد محیط فیزیکی «ذ.ب.ا.» با داده های واقعی اما حجم محدود و انجام تست مرحله دوم

۱۵- ایجاد محیط فیزیکی «ذ.ب.ا.» با داد های واقعی و حجم واقعی و انجام تست مرحله سوم

۱۶- تنظیم سیستم پایگاهی (Data Base System Tuning)

۱۷- آغاز بهره برداری و نگهداری از سیستم

۱۸- گسترش سیستم

۱۹- رفع معایب و بهینه‌سازی سیستم

کنجکاوی؟ مزایا و معایب جداسازی این دو دسته برنامه

تعریف و کنترل و عملیات در داده‌ها چیست؟

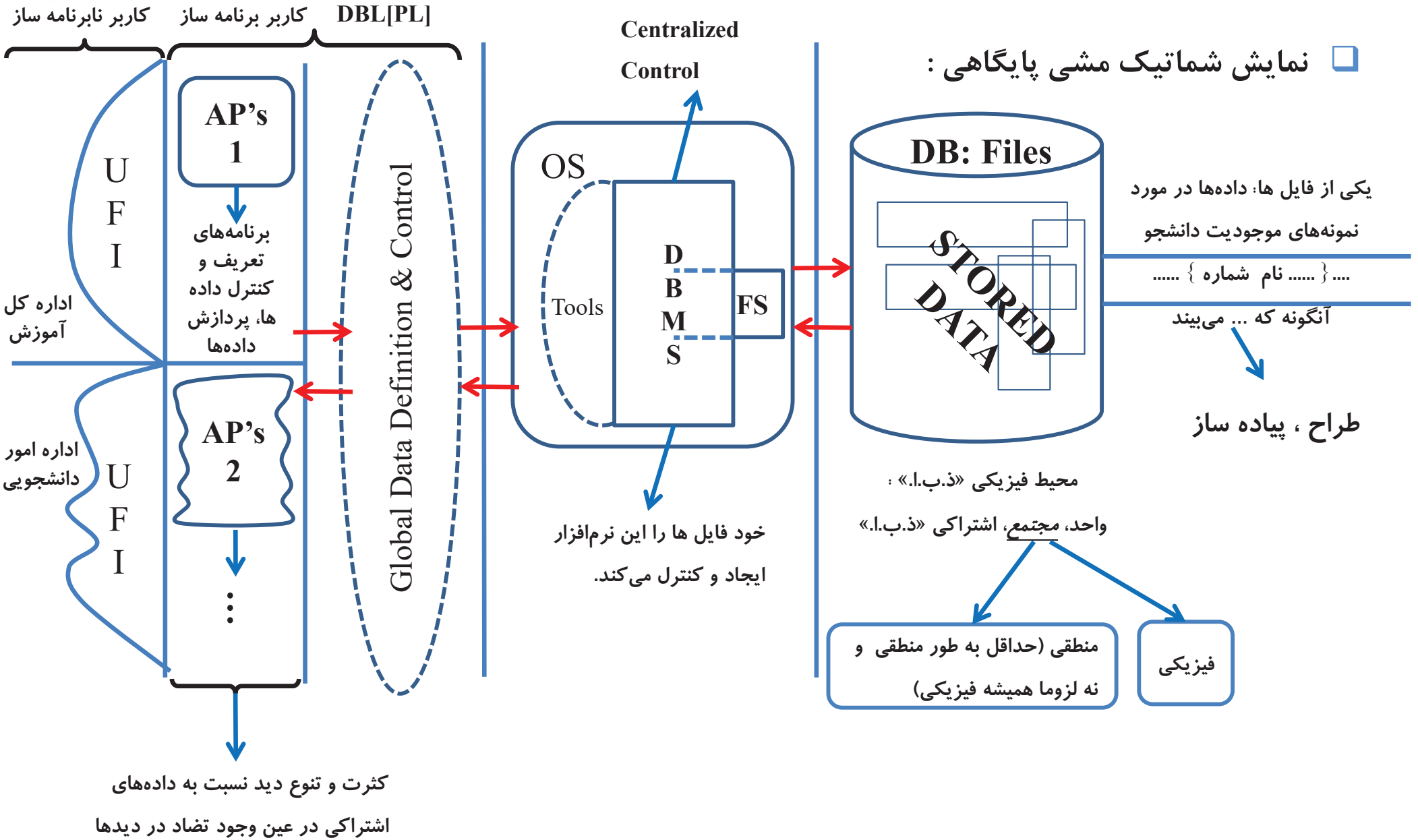
۱- از دیدگاه عملیات در داده‌ها

۲- از دیدگاه زبان‌های برنامه‌سازی

به طور مثال به منظور افزایش کارایی

یکی از ویژگی های DBMS گسترش پذیری سیستم است.

ادامه مثال مقدماتی (مشی پایگاهی)



- نمایش شماتیک مشی پایگاهی :

یادآوری تراکنش Transaction:

- دنباله ای از عملیات («قطعه برنامه») که معمولاً حد اقل یک عمل تغییردهنده (درج، حذف، به روزرسانی) در محیط ذخیره‌سازی داده‌ها انجام می‌دهد و باید یا به تمامی اجرا شود و یا اجرا نشده تلقی شود.
- دارای خواص ACID (Atomicity Consistency Isolation Durability)

مثال: شرط سازگاری پایگاه داده در این مثال : A+B ثابت باشد

```
BEGIN TRANS
        READ (A)
        A = A – 50
        UPDATE (A)
        READ (B)
        B = B + 50
        UPDATE (B)
END TRANS
```

عناصر اصلی محیط پایگاهی:

۱- سخت افزار ←
- ذخیره سازی
- پردازشگر
- ارتباطی (همرسانی) Data Communication

۲- نرم افزار

۳- کاربر

۴- داده

عناصر محیط پایگاهی – (۱) سخت‌افزار

- **سخت افزار ذخیره سازی:**
  - رسانه اصلی: دیسک ترجیحا با تکنولوژی RAID (Redundant Array of Inexpensive Disk)
  - رسانه فرعی: نوار مغناطیسی [از جمله برای تولید نسخه های پشتیبان]

- اغلب DBMSهای امروزی تکنیک های تولید Back up را دارا هستند.

: تکنیک های تولید نسخه پشتیبان؟

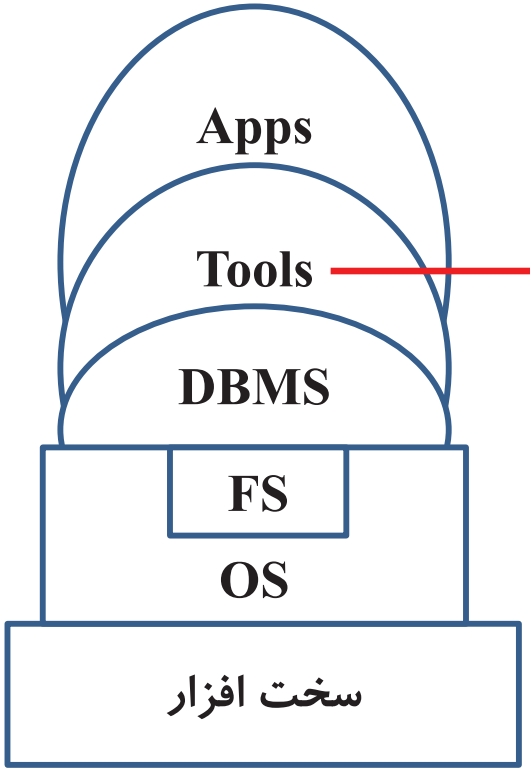
سطوح مختلف Back up

- **سخت افزار پردازشگر:**
  - کامپیوتر های معمولی از هر رده [PC, main,…]
  - اما ماشین‌های خاص DB هم داریم : DB Machines

- **سخت افزار ارتباطی (همرسانی):**
  - امکانات محلی: برای ارتباط دستگاه‌های جانبی با پردازنده
  - امکانات شبکه‌ای: برای ایجاد شبکه در سیستم پایگاهی نامتمرکز

عناصر محیط پایگاهی – (۲) نرم افزار

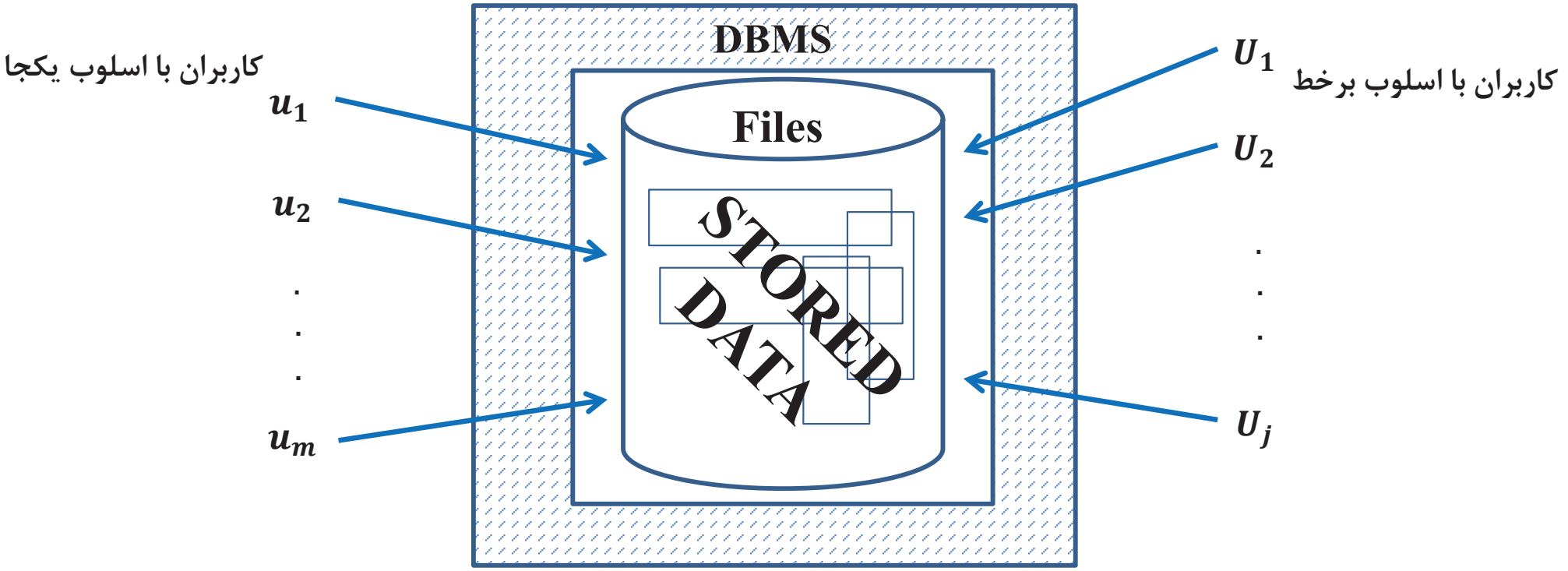
- **انواع نرم افزارهای مطرح در محیط پایگاهی:**
  - سیستم عامل و سیستم فایل (OS و FS)
  - سیستم مدیریت پایگاه داده‌ها (DBMS)
  - ابزارها (Tools)
  - برنامه‌های کاربردی (Apps)



تسهیلات نرم افزار ← یا با خود DBMS می فروشند، یا جداگانه خریداری می‌شود و به امکانات آن اضافه می‌شود.

عناصر محیط پایگاهی – (۳) کاربر

- در معنای عام هر استفاده‌کننده از سیستم پایگاهی را **کاربر** گوییم که انواع مختلفی دارد.



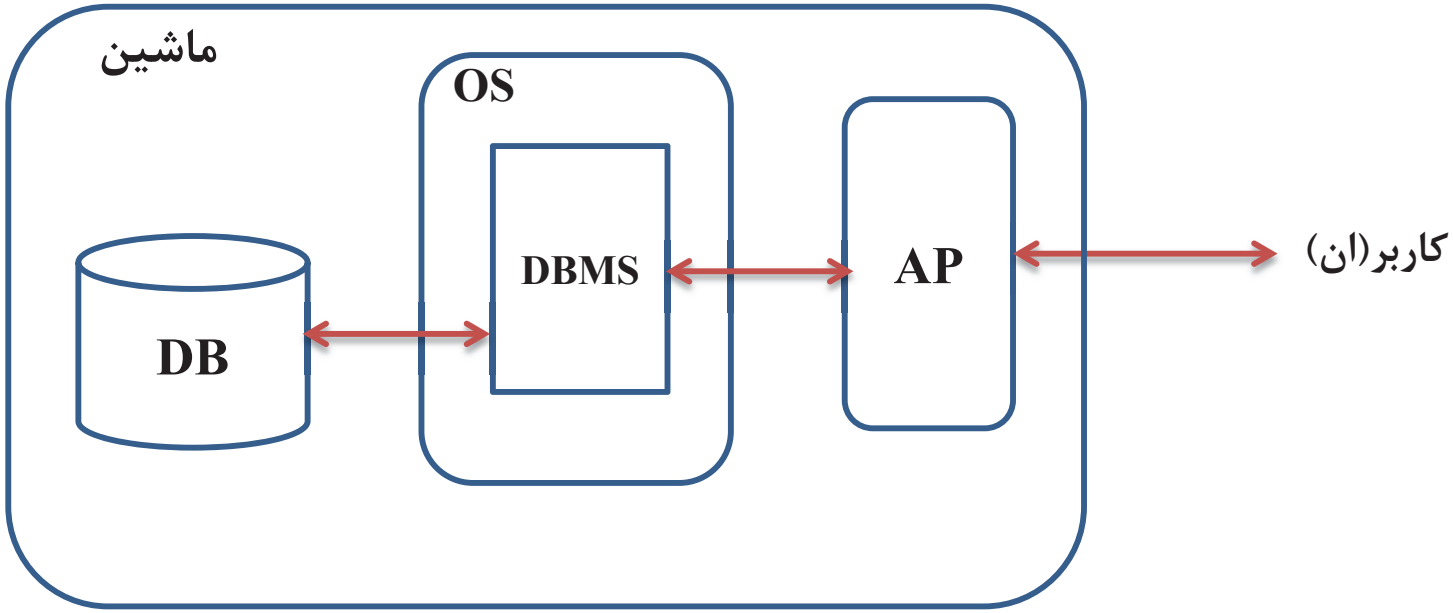
عناصر محیط پایگاهی – (۴) داده

- **داده‌های ذخیره شده** در یک سیستم پایگاهی عبارتند از:
  - داده‌های کاربران
  - داده‌های سیستمی
- مباحث مرتبط با داده در محیط پایگاهی در ادامه درس مطرح می‌گردد.

انواع معماری سیستم پایگاهی

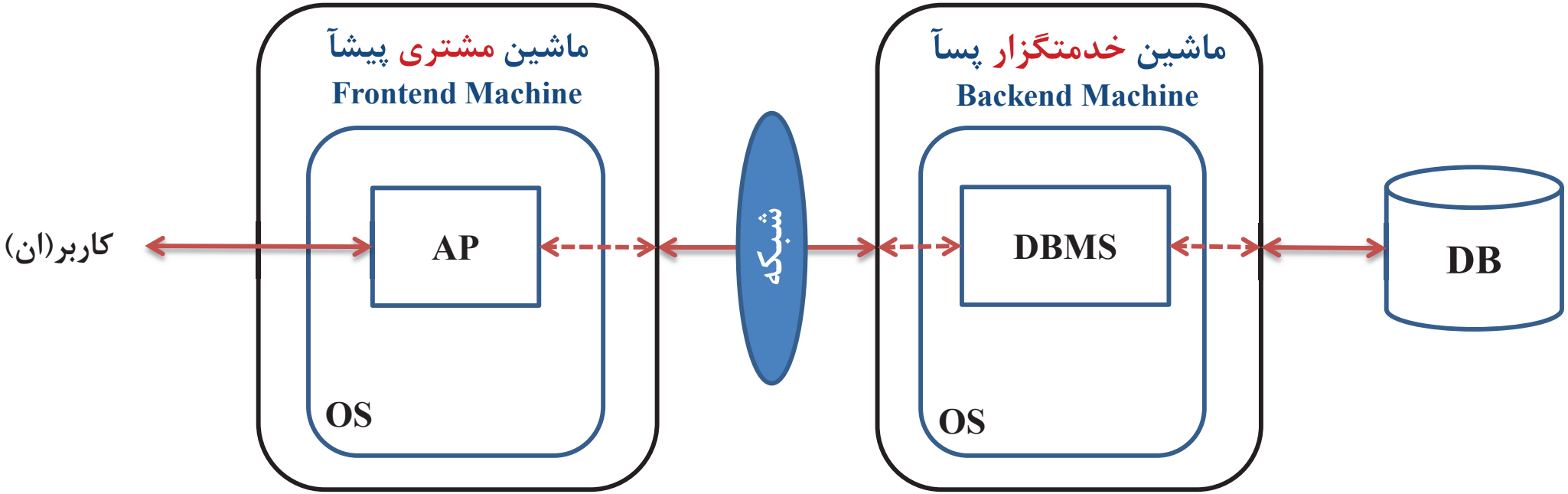
- **سوال:** می‌خواهیم یک سیستم کاربردی پایگاهی ایجاد کنیم.بر اساس کدام معماری ایجادکنیم؟
- **در توصیف معماری یک سیستم باید مشخص کنیم که**
  - از چه مولفه‌هایی، از هر مولفه چند عدد و با چه کیفیتی تشکیل شده است،
  - مولفه‌ها چگونه با هم ترکیب شده‌اند (جنبه ساختاری سیستم)،
  - مولفه‌ها چگونه با یکدیگر در تعامل هستند (جنبه رفتاری سیستم).
- **انواع معماری سیستم پایگاهی:**
  - معماری متمرکز
  - معماری نامتمرکز
    - معماری مشتری-خدمتگزار
    - معماری توزیع‌شده
    - معماری چندپایگاهی
    - معماری با پردازش موازی

- در این معماری یک پایگاه داده (متمرکز و مجتمع) روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر ایجاد می‌شود.
- معمولا به صورت تک کاربری و برای کاربردهای کوچک و با امکانات محدود از این معماری استفاده می‌شود.



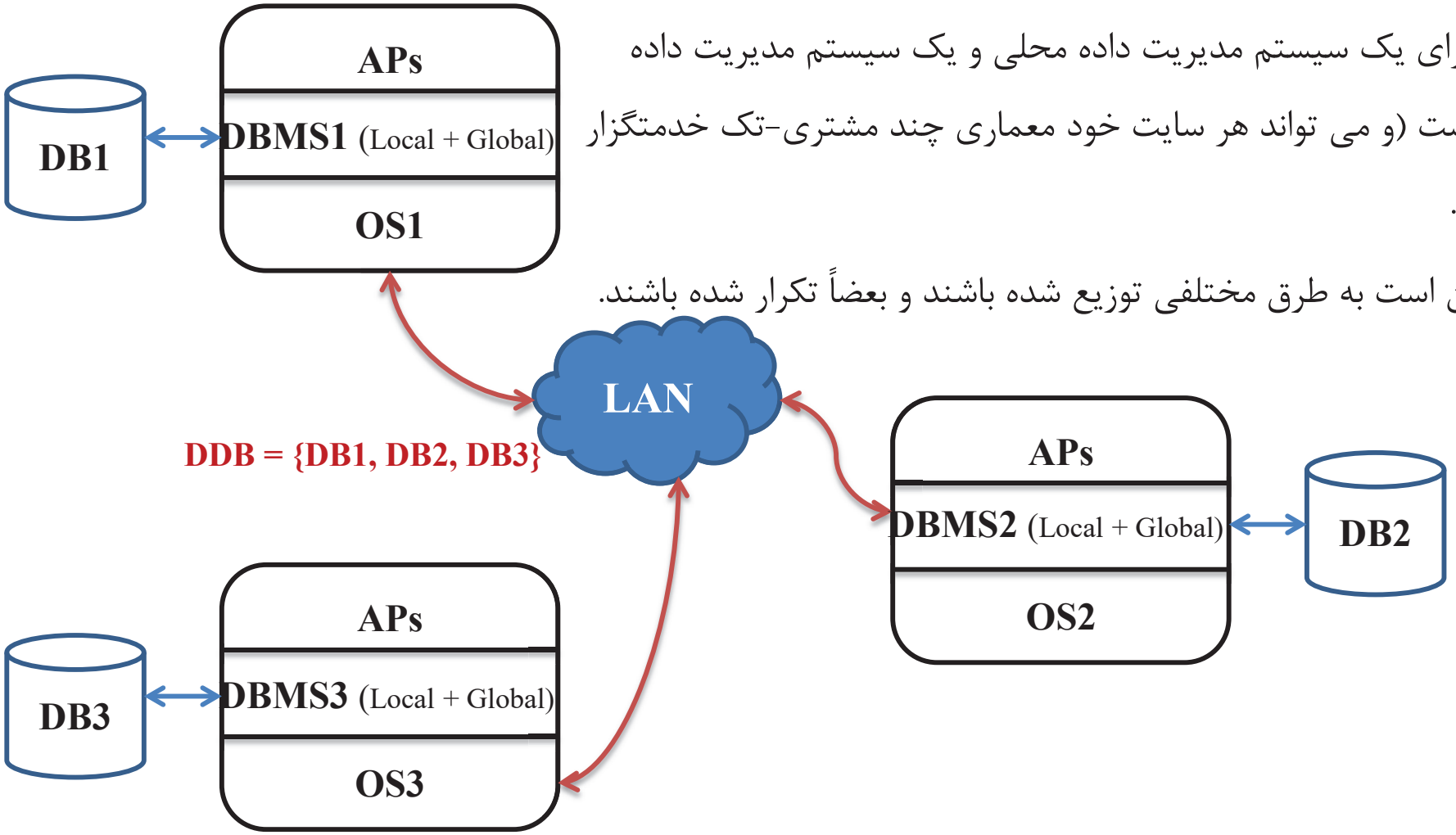
معماری مشتری –خدمتگزار دو لایه

- **معمولا شامل دو سایت:**
  - **سایت مشتری:** تمام برنامه‌های کاربردی در آن اجرا می‌شوند.
  - **سایت خدمتگزار:** تمام داده‌ها در آن ذخیره می‌شوند
- به این معماری، **معماری دولایه** (2-tier) نیز گویند.



معماری توزیع‌شده

- مجموعه‌ای است از چند پایگاه داده منطقاً یکپارچه (مجتمع)، ولی به طور فیزیکی توزیع شده روی یک شبکه کامپیوتری.
- توزیع شدگی از دید برنامه‌ها و کاربران پایگاه داده پنهان است.
- هر سایت دارای یک سیستم مدیریت داده محلی و یک سیستم مدیریت داده توزیع‌شده است (و می تواند هر سایت خود معماری چند مشتری-تک خدمتگزار داشته باشد).
- داده‌ها ممکن است به طرق مختلفی توزیع شده باشند و بعضاً تکرار شده باشند.



به نام آنکه جان را فکرت آموخت

بخش پنجم:
معماری پایگاه داده‌ها

مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

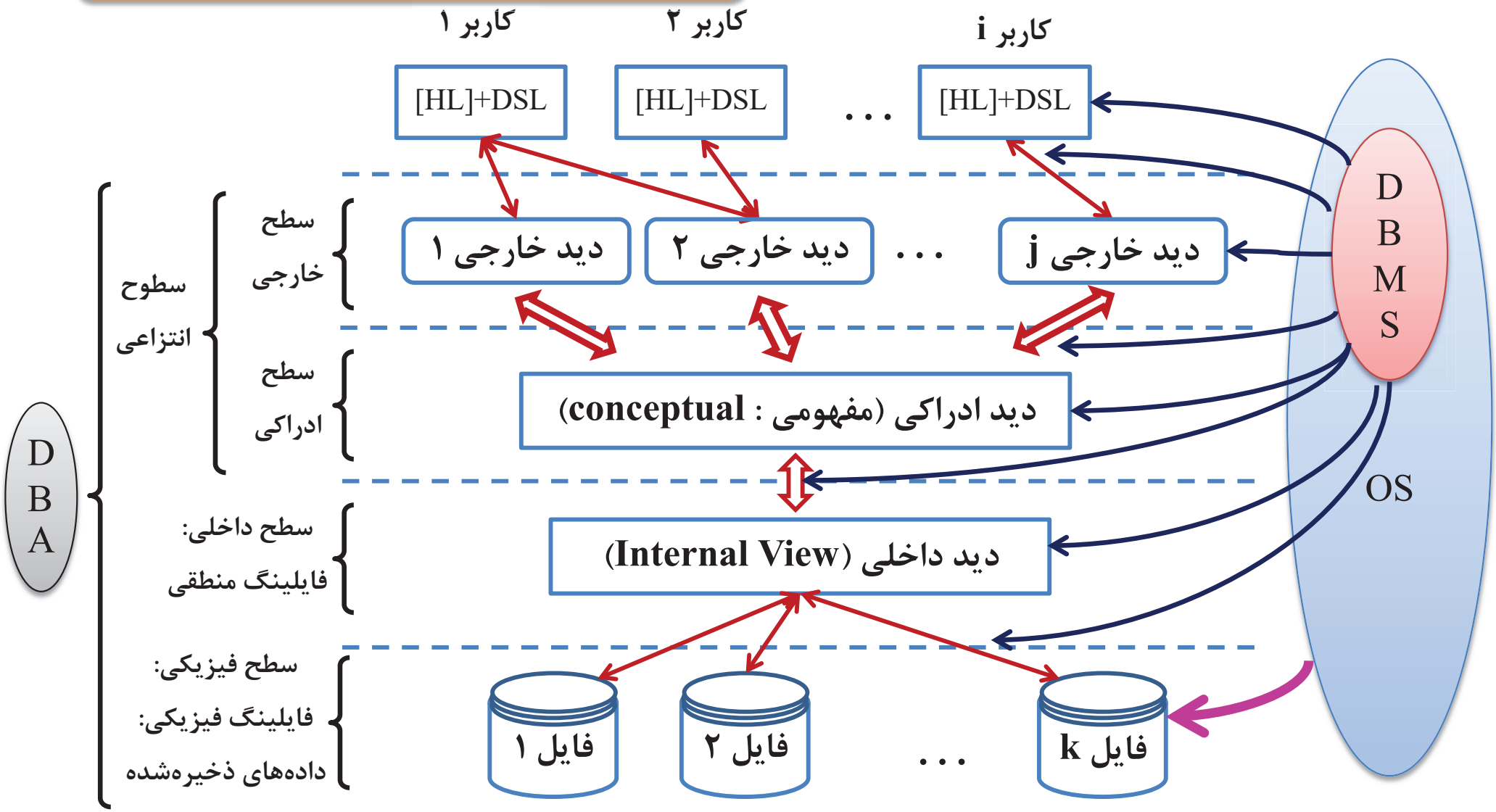
- نیاز به یک معماری واحد از دیدگاه **داده شناسانه** (و نه دیدگاه عملکردی یا دیدگاه مولفه-مبنا) که در آن داده‌ها به گونه‌ای قابل فهم (مستقل از پیچیدگی‌های سطح سمپاد) به کاربر نمایش داده شود.
- عدم وجود اتفاق نظر در چگونگی معماری پایگاه داده‌ها در سالهای آغازین ایجاد

- **پیشنهاد معماری سه سطحی از سوی ANSI / SPARC**
  - سه سطح معماری ANSI، در واقع سه سطح **تعریف و کنترل داده‌ها** است.
  - دو سطح در محیط انتزاعی و یک سطح در محیط فایلینگ منطقی.

معماری سه سطحی [پیشنهادی ANSI]

نشان دهنده نگاشت (تبدیل) بین سطوح ⟷

معماری سه سطحی



به نام آنکه جان را فکرت آموخت

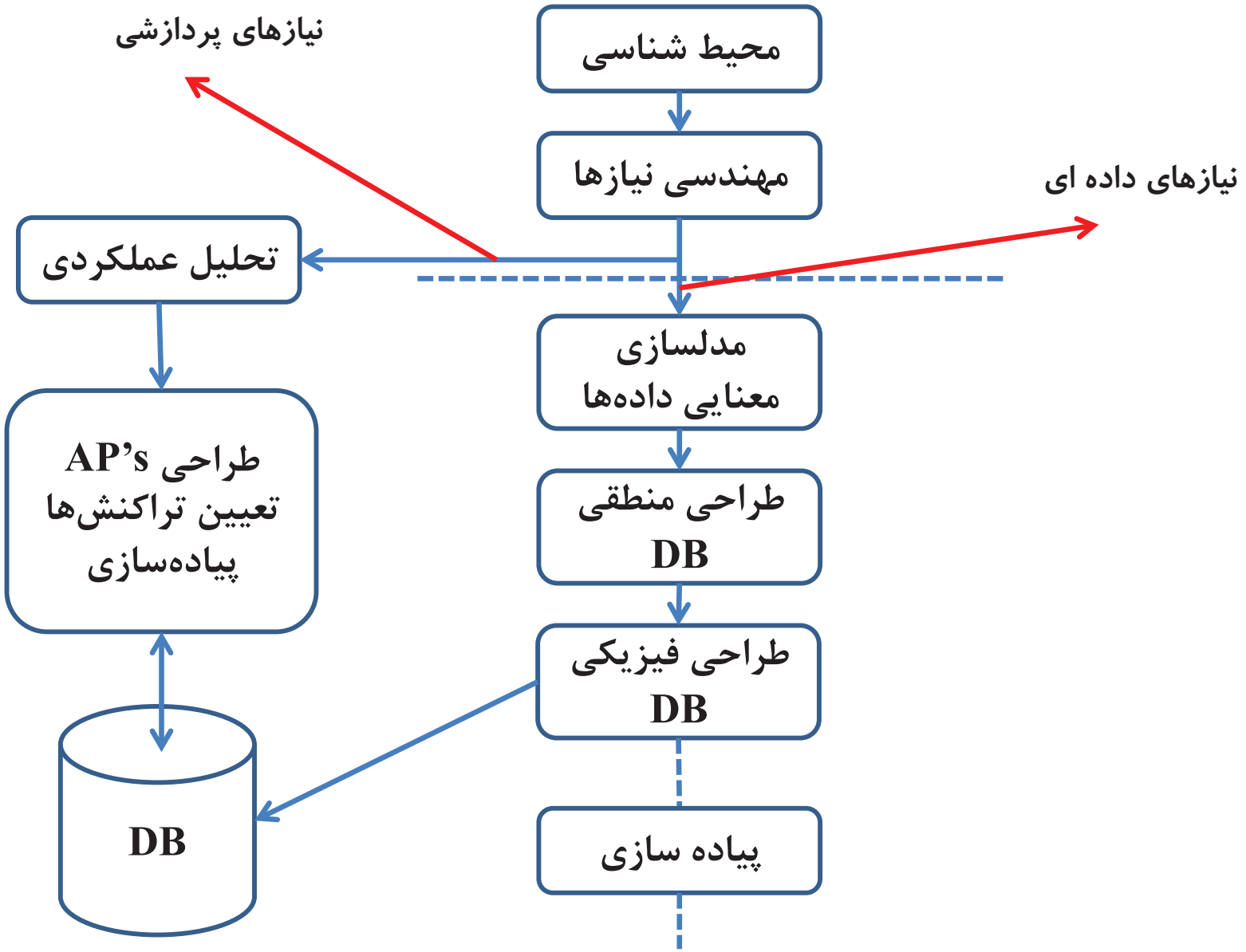
بخش دوم :
مدلسازی معنایی داده‌ها

مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

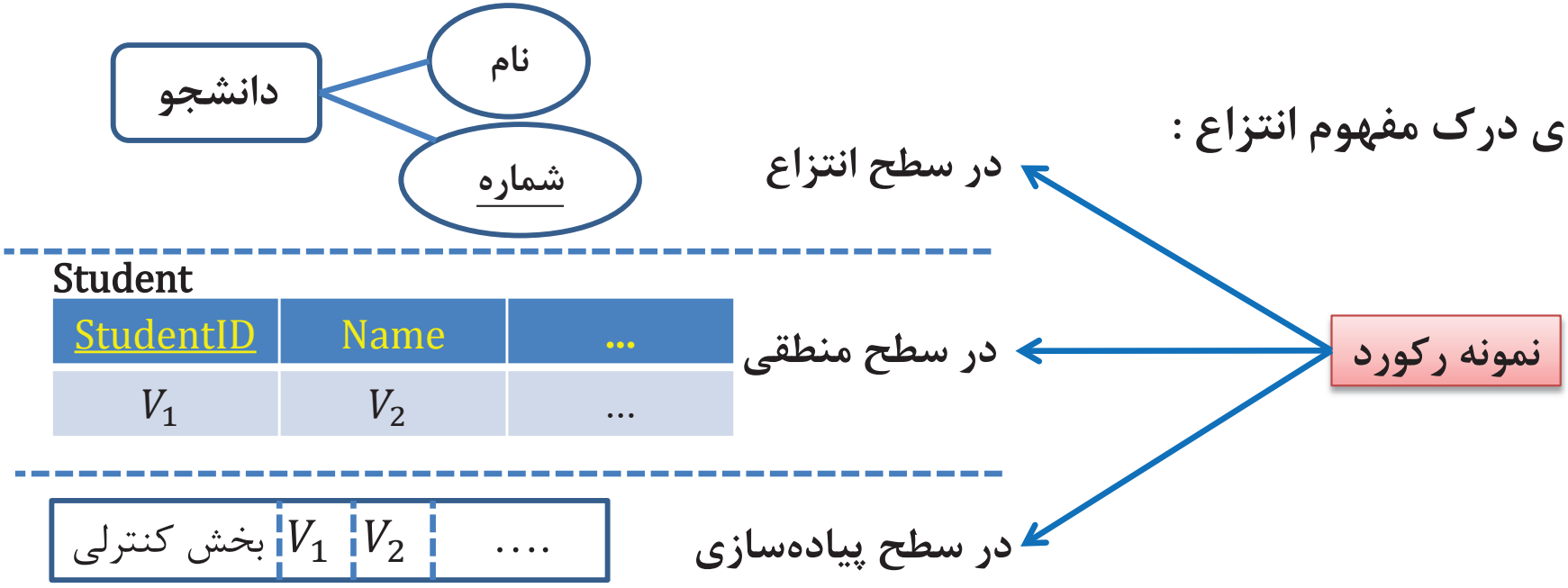
مراحل تولید سیستم اطلاعاتی



مدلسازی معنایی داده‌ها:

- ارائه یک مدل کلی (در بالاترین سطح انتزاع) از داده‌های محیط با استفاده از مفاهیم انتزاعی و براساس *معنایی* که کاربر برای داده‌ها قائل است.
- **مفهوم انتزاعی:** مفهومی است فراتر از سطح منطقی و طبعاً فراتر از سطح پیاده‌سازی

مثال برای درک مفهوم انتزاع :



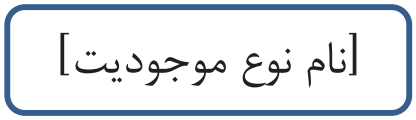
- **برای مدلسازی نیاز به روش داریم:**
  - روش رایج تر در دانش و تکنولوژی پایگاه داده
    - **روش ER (Entity Relationship):** ER مبنایی / ER گسترش یافته (Extended or Enhanced ER)
    - **روش UML (Unified Modeling Language):** خاصّ مدلسازی معنایی داده‌ها نیست بلکه برای مدلسازی و طراحی سیستم های نرم‌افزاری است. لذا با آن می‌توان پایگاه داده را مدل کرد.

- **سه مفهوم اساسی داریم:**
  - نوع موجودیت Entity Type
  - صفت (خصیصه – ویژگی) Attribute
  - نوع ارتباط Relationship Type

- **نمودار ER:**
  - نموداری است که سه مفهوم اساسی نوع موجودیت، صفت و نوع ارتباط در آن نمایش داده می‌شوند. در واقع این نمودار امکانی است برای نمایش مدلسازی و اولین طرح پایگاه داده‌ها در بالاترین سطح انتزاع.
  - برای رسم این نمودار به نمادهایی نیاز داریم. در این درس از نمادهای چِن استفاده می‌شود.

نمادهای نمودار ER مبنایی

- نوع موجودیت

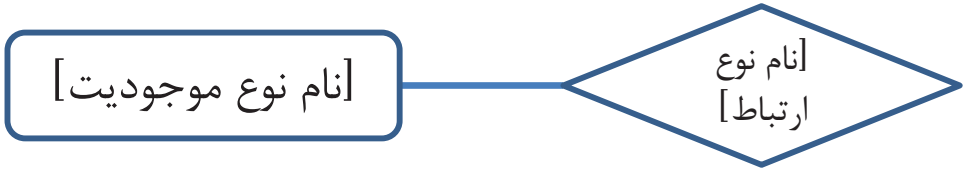


- نوع موجودیت ضعیف

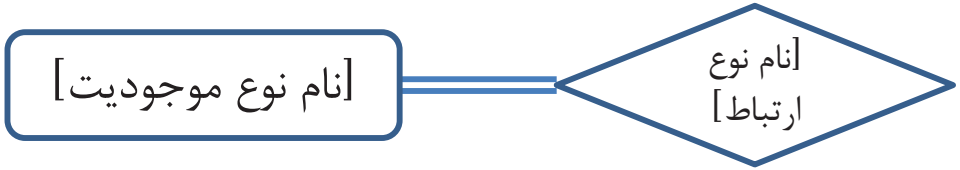
- نوع ارتباط

- نوع ارتباط موجودیت ضعیف با قوی

- مشارکت نوع موجودیت در نوع ارتباط



- مشارکت الزامی



نمادهای نمودار ER مبنایی (ادامه)

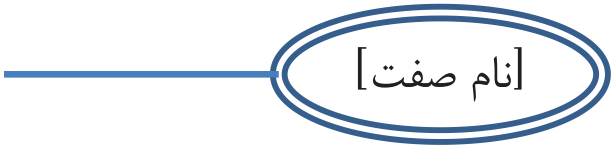
- صفت

- صفت شناسه اول

- صفت شناسه دوم (در صورت وجود)

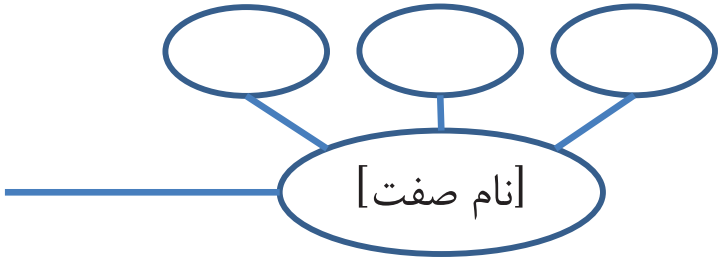
- صفت شناسه مرکب (مثلا دو صفتی)

- صفت چندمقداری



نمادهای نمودار ER مبنایی (ادامه)

- صفت مرکب



- صفت مشتق (مجازی یا محاسبه‌شدنی)

- چندی ارتباط

نوع موجودیت:

- مفهوم کلی شیئ، چیز، پدیده و به طور کلی آنچه از یک محیط که می‌خواهیم در موردش اطلاع داشته باشیم.

- خرد جهان واقع Micro Real World
- Mini World
- جهان مطرح Universe of Discourse(UOD)

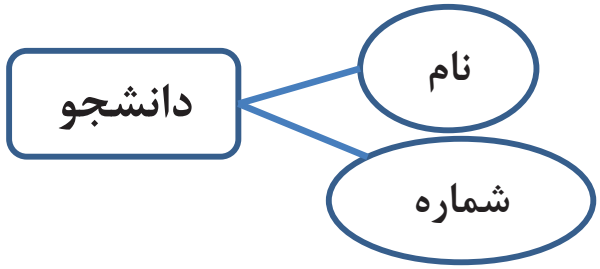
مثال: محیط عملیاتی : دانشگاه

- نوع موجودیت‌ها
  - ۱- دانشجو
  - ۲- درس
  - ۳- استاد
  - ۴- کارمند
  - ...

- تذکر: اولین قدم در مدلسازی معنایی تشخیص درست نوع موجودیت‌ها است.

کنجکاوی؟ در مثال فوق آیا دانشگاه یک نوع موجودیت در نظر گرفته می‌شود یا خیر؟

ER مبنایی – نوع موجودیت (ادامه)



- هر نوع موجودیت:
  - یک نام دارد.
  - یک معنا دارد.
  - مجموعه‌ای از صفات دارد (حداقل یکی).

کنجاوی؟ در چه حالتی بهتر است نوع موجودیت تک صفتی را نوع موجودیت بگیریم؟ در چه حالتی نگیریم؟

  - نمونه‌هایی دارد (حداقل یک نمونه).

کنجاوی؟ در چه حالتی نوع موجودیت تک نمونه‌ای را موجودیت در نظر می‌گیریم؟

  - ارتباط(هایی) با نوع موجودیت(های) دیگر دارد.

کنجاوی؟ آیا نوع موجودیت ایزوله داریم؟

  - نوع موجودیت دو گونه است. ←
    - قوی (مستقل) Strong
    - ضعیف (وابسته) Weak

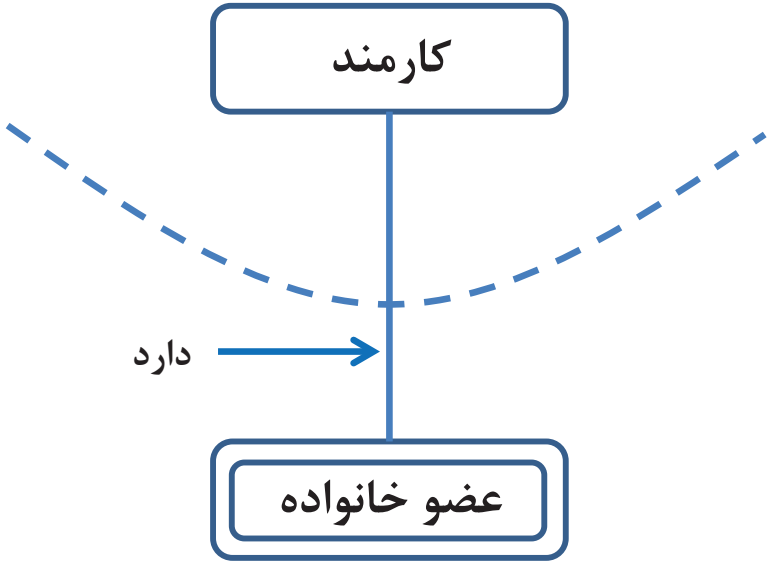
ER مبنایی – نوع موجودیت (ادامه)

تعریف موجودیت قوی:

- نوع موجودیت E را **قوی** گوییم هرگاه خود مستقلاً در محیط مطرح باشد.

تعریف موجودیت ضعیف:

- نوع موجودیت F را **ضعیفِ** نوع موجودیت E گوییم هرگاه به آن «وابستگی وجودی» داشته باشد. (اگر E مطرح نباشد F هم مطرح نیست) به عبارتی F در مدلسازی دیده می‌شود به اعتبار E.
- تذکر: قوی و ضعیف بودن نسبی است.



مثال: عضو *خانواده* وابسته به نوع موجودیت *کارمند* است.

صفت:

- خصیصه یا ویژگی نوع موجودیت و هر نوع موجودیت مجموعه‌ای از صفات دارد که حالت یا وضع آن را توصیف می‌کند.

- محیط عملیاتی: دانشگاه
- نوع موجودیت: درس
- صفات: شماره، نام، تعداد واحد، زمان برگزاری، تاریخ امتحان، نوع درس (پایه، تخصصی، اختیاری،...)، سطح درس (کارشناسی، کارشناسی ارشد، دکترا)، ماهیت درس (نظری، عملی، ترکیبی)

- **هر صفت:**
  - یک نام دارد.
  - یک معنا دارد (معنای مشخص در حیطه معنایی مشخص).
  - یک دامنه یا میدان (Domain) دارد.

- **محدودیتهای صفت:**
  - ۱- محدودیت میدانی: معنای، نوع، طیف مقادیر } صفت را مشخص می‌کند. و نه لزوماً نام صفت را.
  - ۲- محدودیت نمایشی. **مثال:** قالب تاریخ yyyy/mm/dd
  - ۳- محدودیت پردازشی ناشی از نوع صفت یا ناشی از قواعد محیط [غیر از آنچه ناشی از میدان است]
    - **مثال:** سن کاهش نمی‌یابد.
    - **مثال:** عدم جمع دو آدرس: محدودیت ناشی از میدان است.
  - ۴- محدودیت وابستگی به یک صفت دیگر. **مثال:** وابستگی شمول به صفت دیگر $B\{values\} \subseteq A\{values\}$
  - ۵- محدودیت یکتایی مقدار. **مثال:** شماره دانشجویی

آیا صفت محدودیت های دیگری هم دارد؟

ER مبنایی – صفت (ادامه)

مثال

نمره دانشجو GR ← از دامنه Grade

- نوع : Real
- طیف مقادیر : [0,..,20]

رده بندی صفت:

- صفت
  - ۱- **شناسه**
  - ۲- **ناشناسه**

Entity Identifier (EID) ←

۱- **یکتایی** مقدار Uniqueness

۲- مقادیرش همیشه **معلوم** باشد (هیچمقدارناپذیر) Not Null

← ویژگی ذاتی

۳- طول کد نمایش حتی‌الامکان کوتاه

۴- مقادیرش حتی‌الامکان تغییر ناپذیر

← بهتر است که داشته باشد.

- صفت
  - ۱- **ساده – تجزیه ناپذیر**: از نظر معنایی در یک محیط مشخص - اگر صفت را تجزیه کنیم، خودِ تکه ها مقداری از صفت در آن محیط نشود. مثال: عنوان درس
  - ۲- **مرکّب**: از چند صفت ساده (و می‌تواند ساختار سلسله مراتبی هم داشته باشد) مثال: آدرس (ترکیبی از استان، شهر، خیابان، ...)

- **توجه:** ساده یا مرکب بودن نسبی است و نه مطلق. بستگی به حیطه معنایی و کاربرد دارد. (مثال: آدرس از دید نشریه (ساده) یا از دید شهرداری (مرکب).

کنجکاوی؟ اینکه صفت مرکّب را در یک فیلد ذخیره کنیم یا اجزا را در فیلد های مجزا به چه عواملی بستگی دارد؟

- صفت
  - ۱- **تک مقداری:** به ازای یک نمونه از نوع موجودیت E، حداکثر یک مقدار می‌گیرد. مثال: نام درس
  - ۲- **چند مقداری:** حدّاقل برای یک نمونه از نوع موجودیت E، بیش از یک مقدار . مثال: شماره تلفن استاد
- توجه
  - ساده – تک مقداری
  - مرکب - تک مقداری
  - ساده - چند مقداری
  - مرکب - چند مقداری
- صفت
  - ۱- **هیچمقدار پذیر** (Nullable یا Nullvalue): مقدار صفت می تواند ناشناخته، ناموجود، تعریف نشده یا غیر قابل اعمال باشد. مثال: شماره تلفن دانشجو
  - ۲- **هیچمقدارناپذیر** (Not nullabe): حتما مقدار صفت برای هر نمونه موجودیت باید معلوم باشد. مثال: شماره درس

کنجکاوی؟ مشکلات هیچمقدار؟ package ها با آن چه برخوردی دارند؟

- صفت
  - ۱- **واقعی (Real)**: مقدار ذخیره شده در DB دارد. مثال: نمره درس
  - ۲- **مجازی – مشتق (Virtual)**: مقدار ذخیره شده در DB ندارد، سیستم با پردازشی معمولاً **محاسبه** و مقدارش را در اختیار کاربر قرار می دهد. مثال: میانگین نمرات درس

- **تذکر:** اگر صفتی ماهیت **محاسبه شوندگی** داشته باشد لزوماً مجازی نیست و ممکن است برای افزایش سرعت و در صورتی که بسامد (فرکانس) ارجاع زیاد باشد مقدار ذخیره شده داشته باشد.

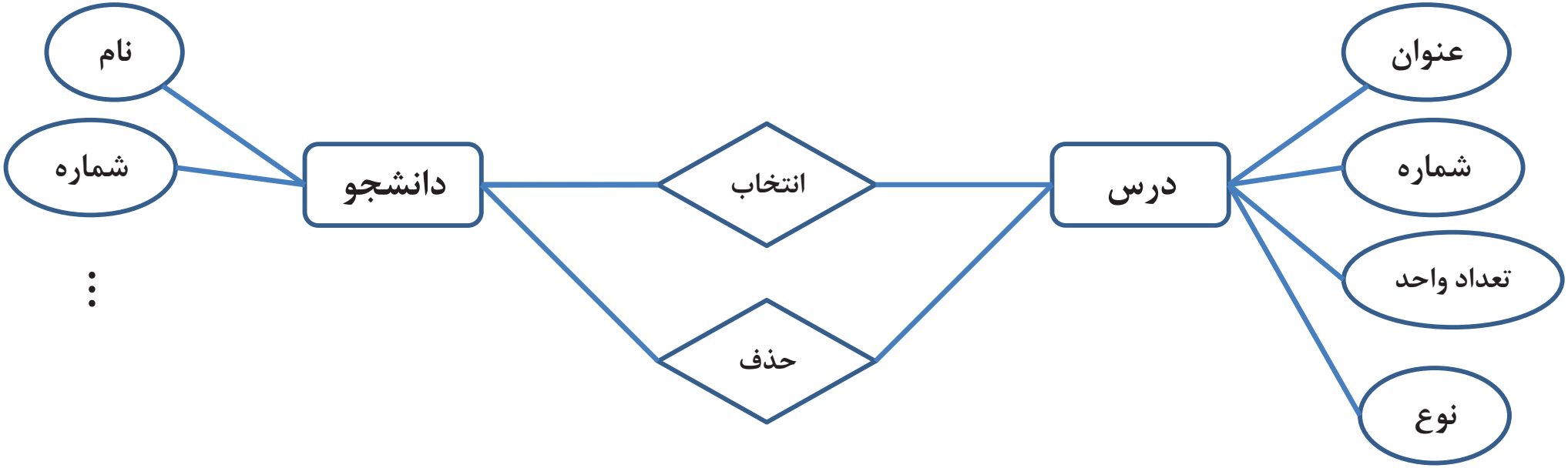
نوع ارتباط Relationship Type:

- رابطه، اندرکنش و یا تعامل بین نوع موجودیت $N \geq 1$ ← ارتباط با خود - بازگشتی (self-relationship) $N = 1$

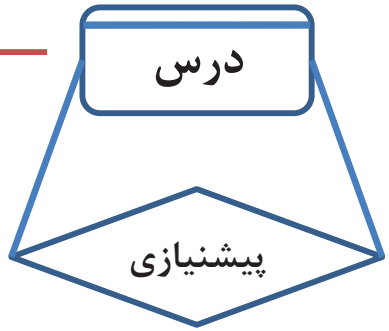
ارتباط نوع موجودیت‌های دانشجو و درس

- دانشجو درس را انتخاب می‌کند.
- دانشجو درس را حذف می‌کند.

ER مبنایی – نوع ارتباط (ادامه)



مثال ارتباط موجودیت با خود :



طرز نمایش نوع موجودیت زمانی که یکبار دیگر در نمودار ER آمده باشد. (به خاطر اجتناب از شلوغ شدن نمودار)

فکاوی؟ مفهوم پیشنیازی درس را به چند روش دیگر می‌توان مدل کرد؟

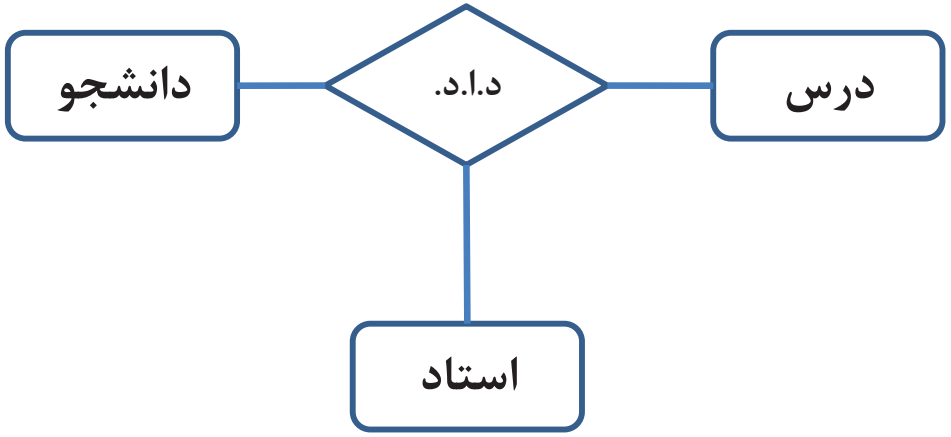
ER مبنایی – نوع ارتباط (ادامه)

- **نوع ارتباط:**
  - یک نام دارد.
  - یک معنا دارد.
  - شرکت کنندگانی (participants) دارد ($N \geq 1$).
  - به تعداد شرکت کنندگان **درجه** (arity or degree) ارتباط گویند.

N	اصطلاح
۱	ارتباط یگانی
۲	ارتباط دوگانی
۳	ارتباط سه‌گانی
n	ارتباط n-گانی (n-ary)

مثال درجه یک و دو: مثال‌های پیش دیده

مثال درجه سه: ارتباط درس، استاد، دانشجو



- تذکر: در عمل به ندرت $N \geq 4$ پیش می‌آید.

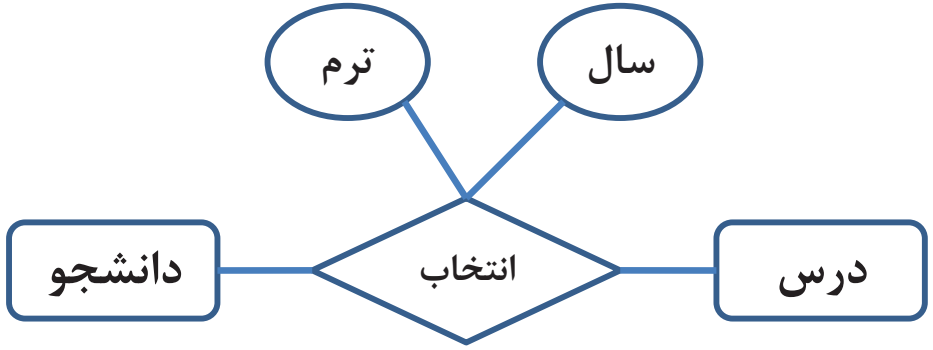
- مشارکت نوع موجودیت E در نوع ارتباط R
  - **الزامی** (کامل): هر نمونه از موجودیت E لزوماً در یک نمونه ارتباط R مشارکت دارد.
  - **غیر الزامی** (ناقص): حداقل یک نمونه موجودیت E وجود دارد که در هیچ نمونه ارتباط R مشارکت ندارد.
- الزامی بودن مشارکت از محدودیت‌های معنایی محیط، ناظر به نوع ارتباط است.

مثال: هر دانشجو لزوماً درسی را انتخاب می‌کند ولی همه دروس لزوماً توسط دانشجویان انتخاب نمی‌شوند.

ER مبنایی – نوع ارتباط (ادامه)

- **هر نوع ارتباط:**
  - می‌تواند صفت(هایی)، موسوم به صفت(های) توصیفی داشته باشد.

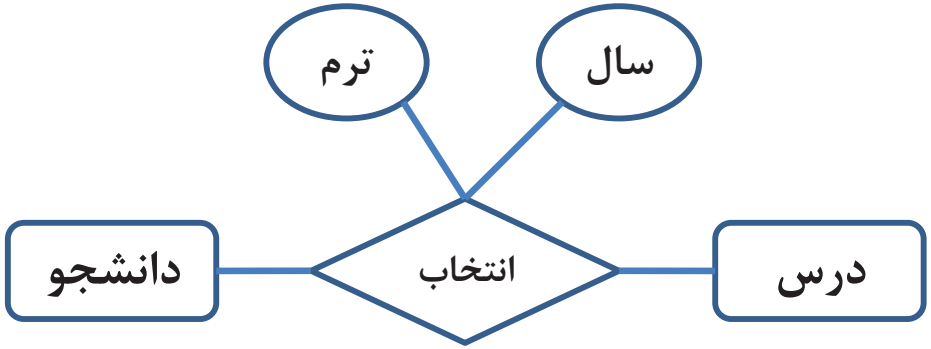
مثال دانشجوی x درس y را در چه ترم و سالی انتخاب می‌کند؟



- **نکته مهم:** هر نمونه ارتباط باید توسط نمونه موجودیت‌های شرکت کننده در آن ارتباط به طور یکتا قابل شناسایی باشد [Silb2010].

ER مبنایی – نوع ارتباط (ادامه)

- در مواردی که به ظاهر نتوانیم با نمونه موجودیت‌های شرکت کننده، یکتایی نمونه‌های یک ارتباط را تامین نماییم، می‌توانیم از صفت چندمقداری (برای رعایت نکته بیان شده) استفاده کنیم.



دانشجو	درس	سال	ترم
۹۲۱۰۱۲۳۵	۴۰۳۸۴	۹۴-۹۳	۲
۹۲۱۰۱۲۳۵	۴۰۱۳۲	۹۵-۹۴	۱

قابل درج نیست. چون ترکیب دانشجو و درس تکرار می‌شود و دیگر شناسه رابطه محسوب نمی‌شود.

دانشجو	درس	سال	ترم
۹۲۱۰۱۲۳۵	۴۰۱۳۲	۹۵-۹۴	۱
۹۲۱۰۱۲۳۵	۴۰۳۸۴	۹۴-۹۳	۲
		۹۵-۹۴	۱

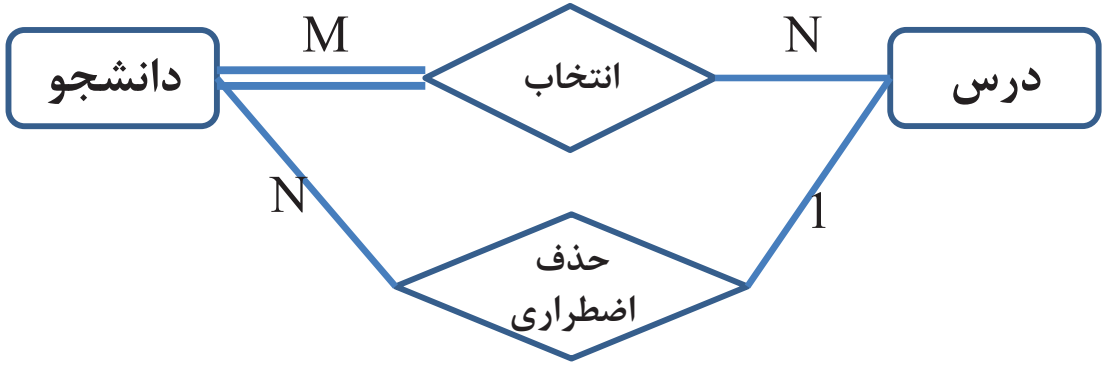
قابل درج است؛ به عنوان مقادیر دیگر یک صفت مرکب چند مقداری.

چندی ارتباط Multiplicity یا Cardinality Ratio:

تناظر
1:1
1:N
M:N

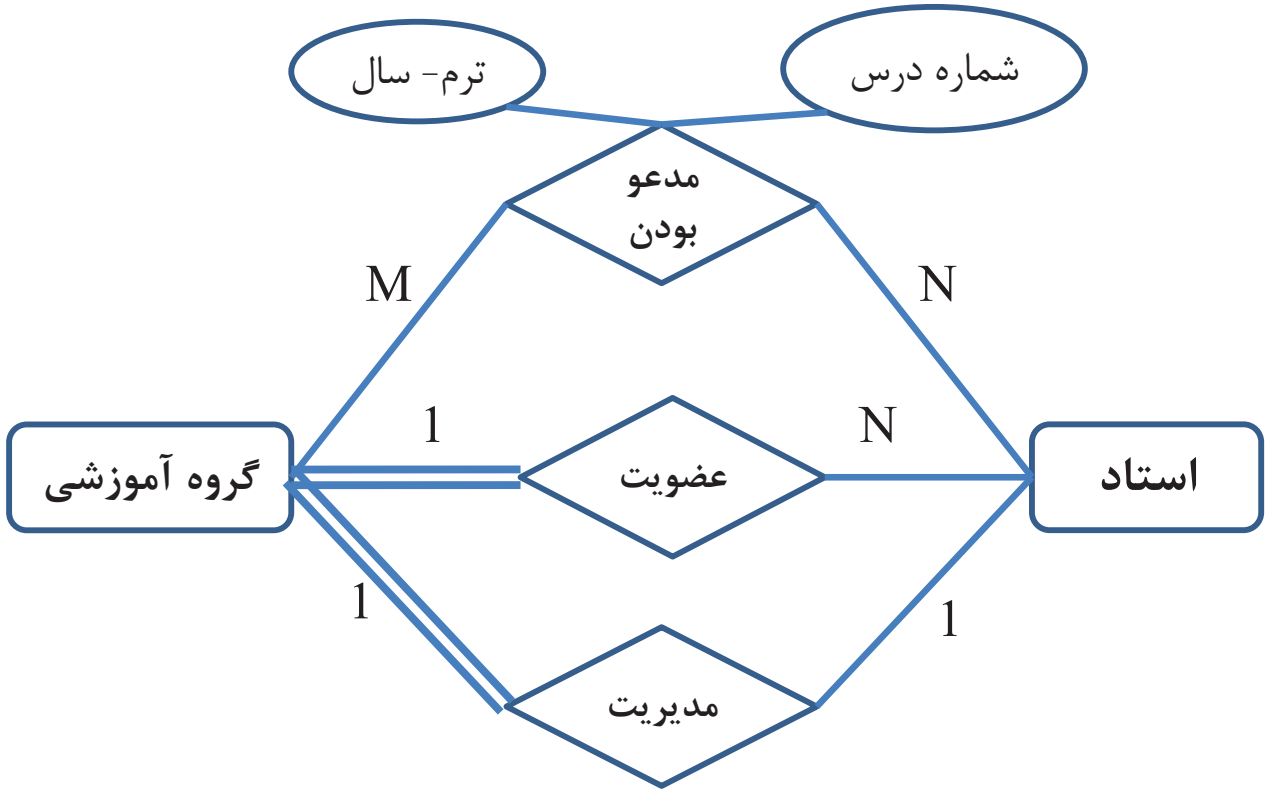
- چندی ارتباط بین دو نوع موجودیت E و F عبارت است از چگونگی تناظر بین عناصر مجموعه نمونه‌های موجودیت E و عناصر مجموعه نمونه‌های موجودیت F.
- اگر دو نوع موجودیت E و F را در نظر بگیریم:
  - در ارتباط یک به یک، یک نمونه از E حداکثر با یک نمونه از F ارتباط دارد و برعکس.
  - در ارتباط یک به چند (از E به F)، یک نمونه از E با n نمونه از F ($n\geq 1$) و در صورت مشارکت غیرالزامی، ($n \geq 0$) ارتباط دارد، ولی یک نمونه از F حداکثر با یک نمونه از E ارتباط دارد.
  - در ارتباط چند به چند، یک نمونه از E با n نمونه از F ($n\geq 1$) ارتباط دارد و برعکس.
- **نکته:** چندی نوع ارتباط چندگانی ($m>2$) عبارت است از تعداد نمونه‌های یک نوع موجودیت شرکت کننده در آن نوع ارتباط، وقتی که تعداد نمونه‌های m-1 نوع موجودیت دیگر شرکت کننده در نوع ارتباط را ثابت فرض کنیم.

مثال با فرض اینکه هر دانشجو چند درس می‌تواند انتخاب کند ولی فقط یک درس را می‌تواند حذف اضطراری کند، چندی ارتباطات به صورت زیر خواهد بود.

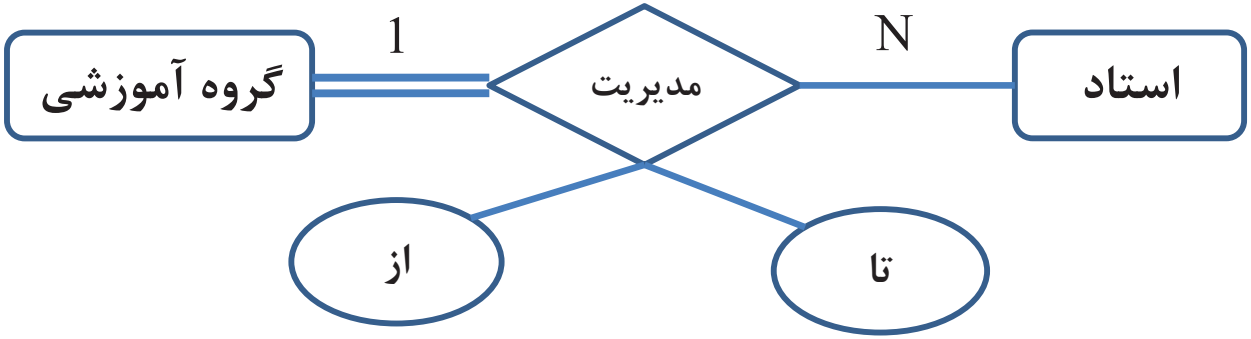


ER مبنایی – نوع ارتباط (ادامه)

مثال مثالی دیگر از چندی ارتباط

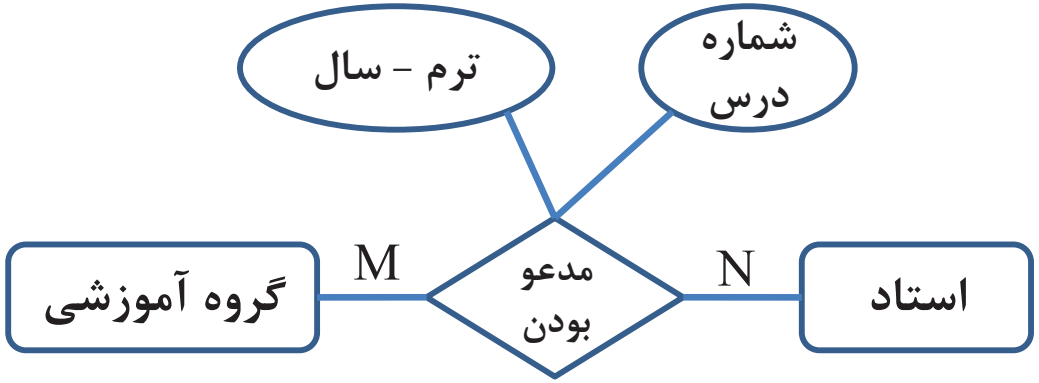


- **تذکر:** اگر به ارتباط صفت هایی از جنس زمان بدهیم، چندی ارتباط می‌تواند بسته به قواعد معنایی محیط تغییر کند.

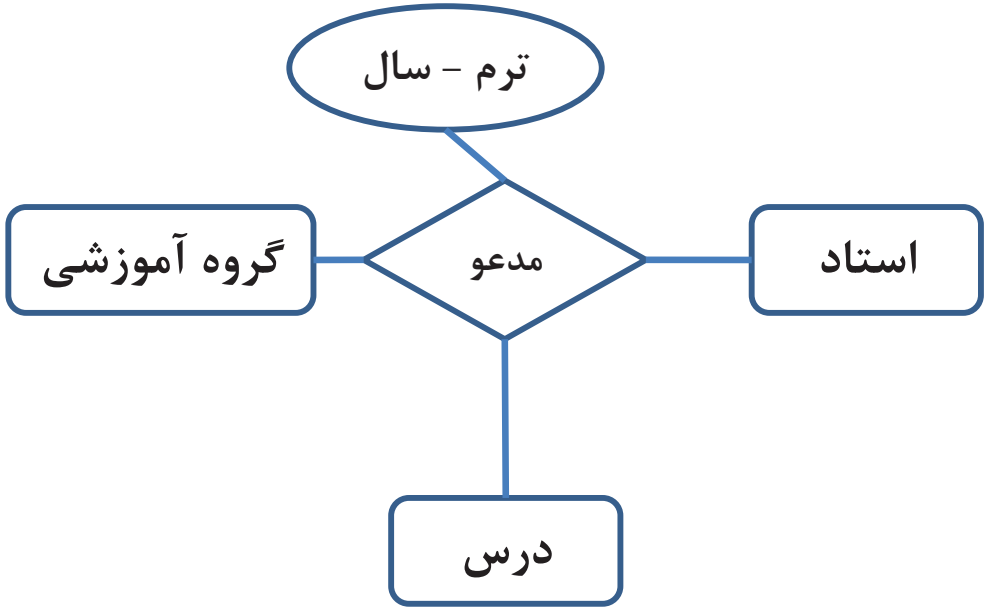


ER مبنایی – نوع ارتباط (ادامه)

نگاهی گونه‌های دیگر مدل کردن نوع ارتباط مدعو بودن چیست؟

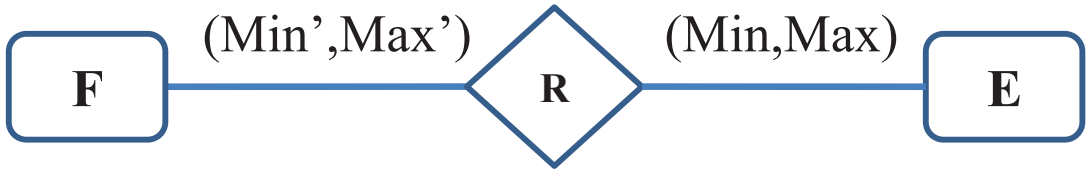


- با استفاده از نوع ارتباط سه گانی:



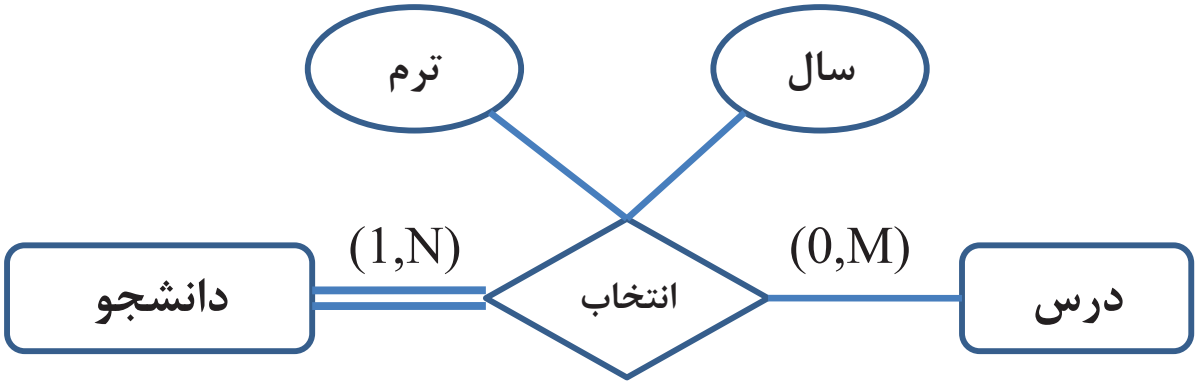
ER مبنایی – نوع ارتباط (ادامه)

- **تذکر: طرز دیگر نمایش چندی ارتباط**



- هر نمونه e از نوع موجودیت E باید حداقل در Min و حداکثر در Max نمونه از ارتباط R شرکت داشته باشد.

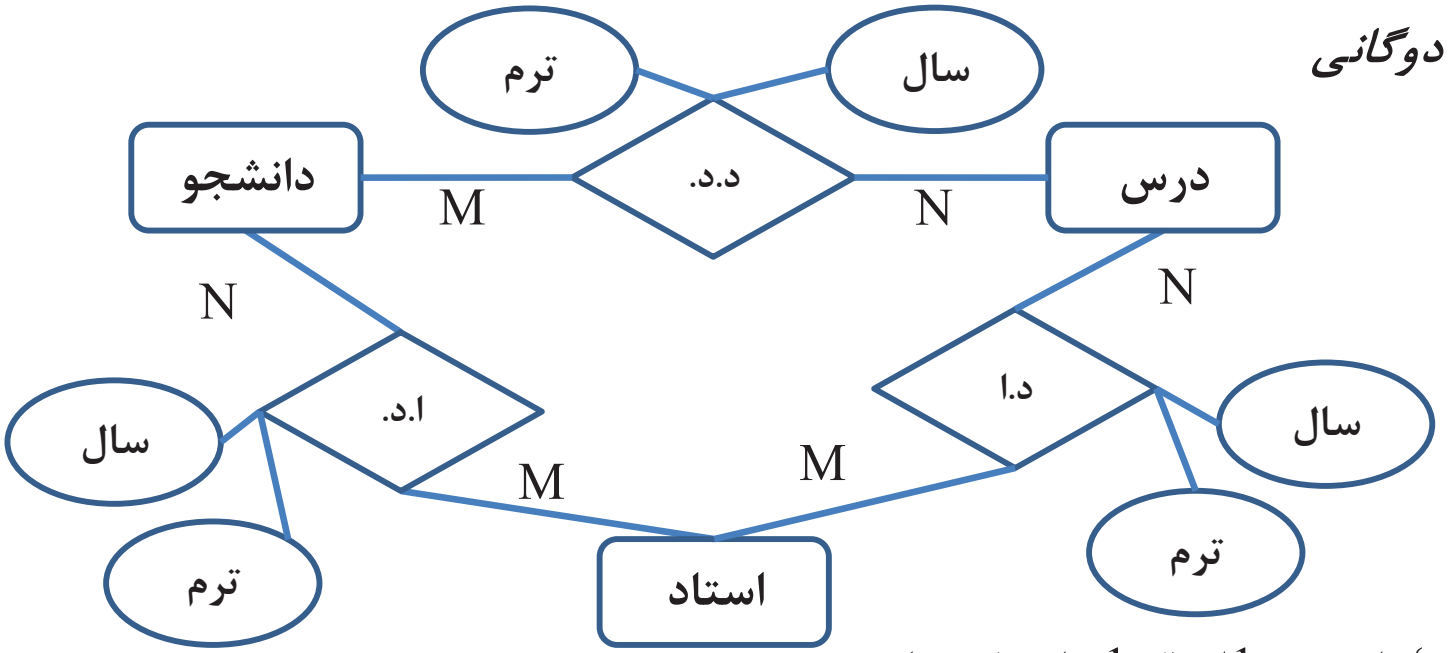
مثال رابطه انتخاب درس توسط دانشجو



کنجکاوی؟ مزایای این روش نمایش چندی؟

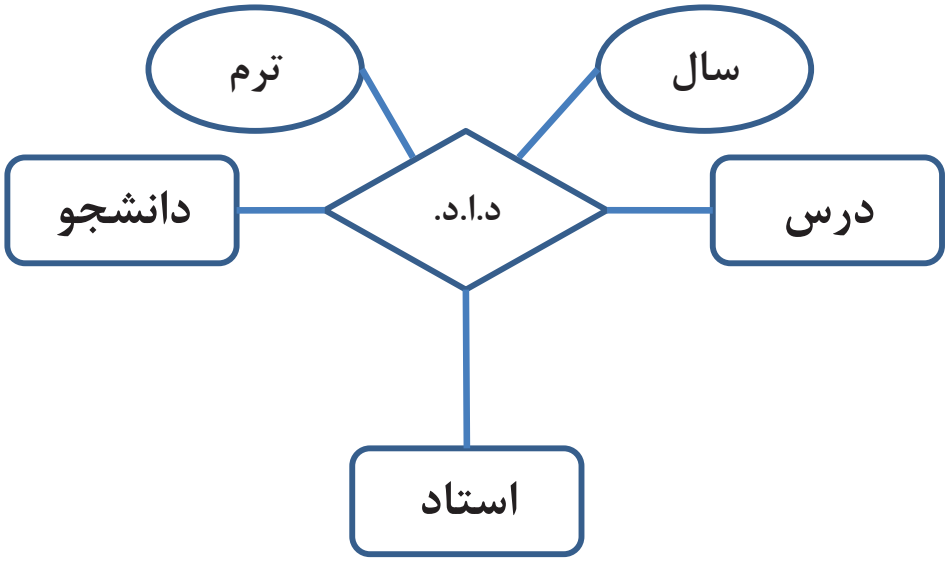
ER مبنایی – نوع ارتباط (ادامه)

- **نکته مهم در مورد ارتباط بین سه نوع موجودیت:**
  - ***مدل یک: سه ارتباط دوگانی***



- سه فقره اطلاع:
  - دانشجو ’s‘ درس ’c‘ را در ترم t1 سال y1 اخذ کرده است.
  - استاد ’p‘ درس ’c‘ را در ترم t1 سال y1 ارایه کرده است.
  - دانشجو ’s‘ دانشجوی استاد ’p‘ است.
- از این سه فقره اطلاع لزوماً همیشه **نمی‌توان** نتیجه گرفت که دانشجو ’s‘ درس ’c‘ را با استاد ’p‘ گذرانده است.

- *مدل دوم : ارتباط سه گانی*

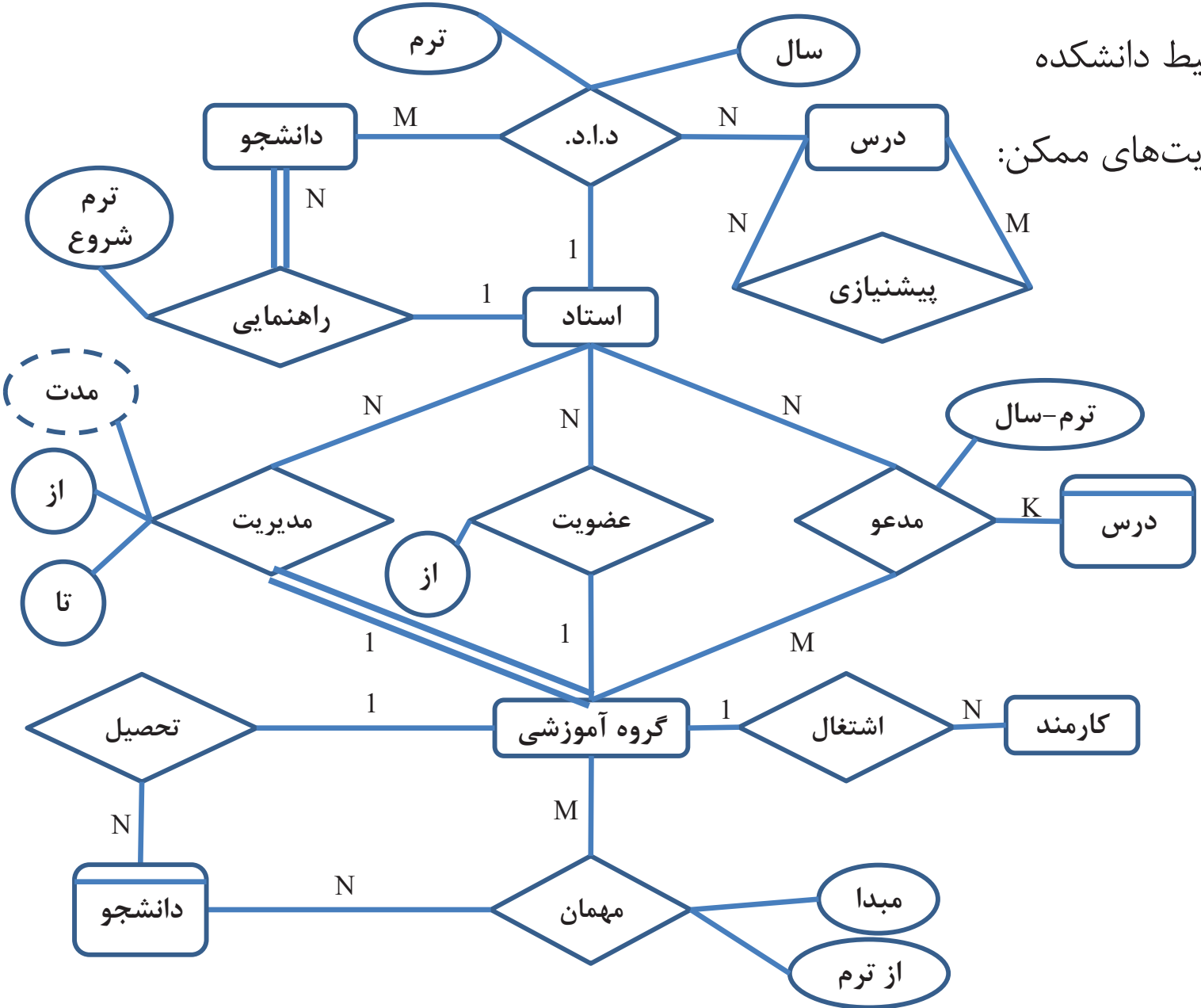


- در حالت سه ارتباط دوگانی اگر از فقره اطلاع‌های دوگانی، فقره اطلاع سه گانی را استنتاج کنیم در شرایطی که از لحاظ معنایی این استنتاج درست نباشد می گوییم دچار **دام پیوندی حلقه‌ای** شده‌ایم.

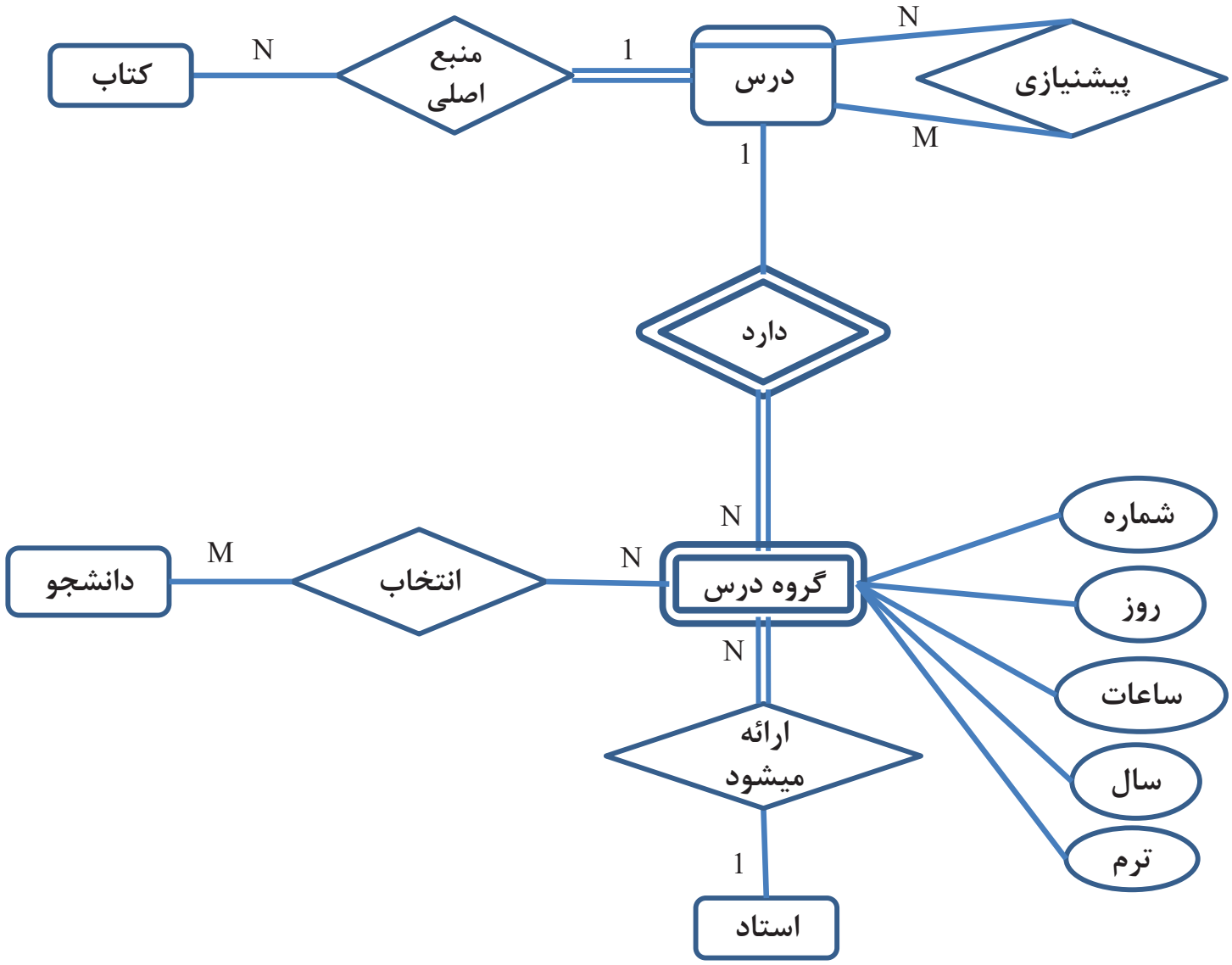
کنجاوی؟ انواع دیگر دام چیست؟ (دام چندشاخه (چتری)، دام گسل (شکافت)، ...)

مثال: محیط دانشکده

- مثال: فعالیت هایی از محیط دانشکده
- بعضی از نوع موجودیت‌های ممکن:
  - دانشجو
  - استاد
  - درس
  - کارمند
  - گروه آموزشی
  - کتاب
  - ...

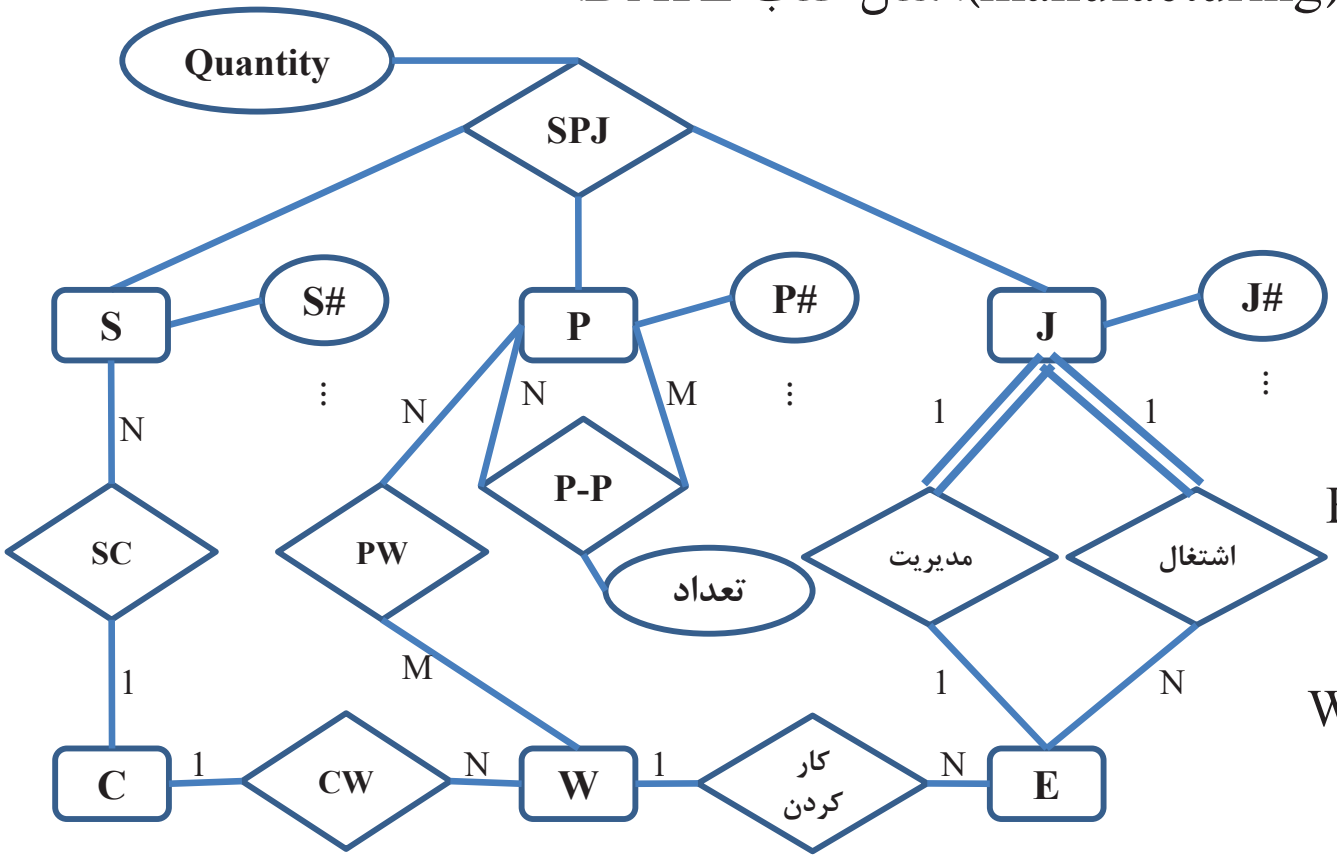


مثال: محیط دانشکده (ادامه)



مثال: محیط تولید

- مثال: محیط تولیدی-کارگاهی (manufacturing). مثال کتاب DATE.



- نوع موجودیت ها:
  - تولید کننده :S Supplier
  - نوع قطعه :P Part
  - پروژه :J Project
  - :E Employee
  - :C City
  - :W Warehouse

گسترش داده شود.

نکات راهنمای تدوین نمودار ER

- مشکل تصمیم‌گیری در مورد اینکه یک مفهوم، نوع موجودیت در نظر گرفته شود یا صفت یا نوع ارتباط باید در یک فرآیند تدریجی در مدلسازی معنایی داده‌ها اصلاح شود.
  - اگر یک مفهوم، صفت به نظر آید، آنرا **صفت** می‌گیریم، اما اگر به نوع موجودیت دیگری ارجاع داشته باشد، آن را یک **نوع ارتباط** در نظر می‌گیریم.
  - اگر یک (چند) صفت به هم مرتبط (از لحاظ معنایی) در چند نوع موجودیت، مشترک باشند، آنها را به عنوان **صفات یک نوع موجودیت مستقل** منظور می‌کنیم.
  - اگر یک **نوع موجودیت**، تنها یک صفت داشته باشد و تنها با یک نوع موجودیت دیگر مرتبط باشد، آن را **صفت** در نظر می‌گیریم.
  - اگر مجموعه‌ای از صفات مستقلا قابل شناسایی نباشند، آن را به صورت **نوع موجودیت ضعیف** در نظر می‌گیریم.

❑ **EER**: Extended ER یا Enhanced ER

❑ ER مبنایی کمداشت‌هایی دارد در نمایش بعضی نوع ارتباطها (که بعدا در حیطه شیئ‌گرایی مطرح شد)

ER

+

بعض نوع ارتباطهای دیگر

۱- ارتباط "IS A" ← «گونه ایست» / «هست یک»
- تکنیک تخصیص- ویژه نمایی Specialization
- تکنیک تعمیم Generalization

۲- ارتباط "IS A PART OF" ←
- تکنیک تجزیه
- تکنیک ترکیب

۳- ارتباط با ارتباط ← تکنیک تجمیع Aggregation

ارتباط "IS A"

- **ارتباط IS A:** ارتباط بین یک نوع موجودیت عام است با نوع موجودیت (های) خاص آن که بر

زبرنوع (Supertype)

زیرنوع (SubType)

اساس یک ضابطه مشخص بازشناسی می‌شود.

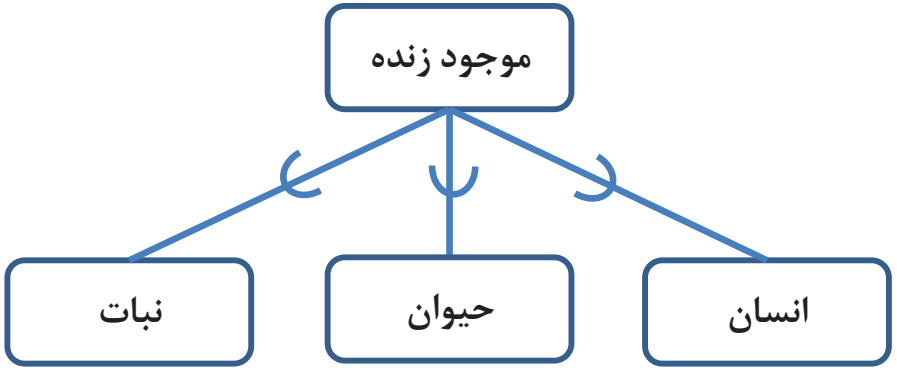
صفت معرف

Defining Attribute

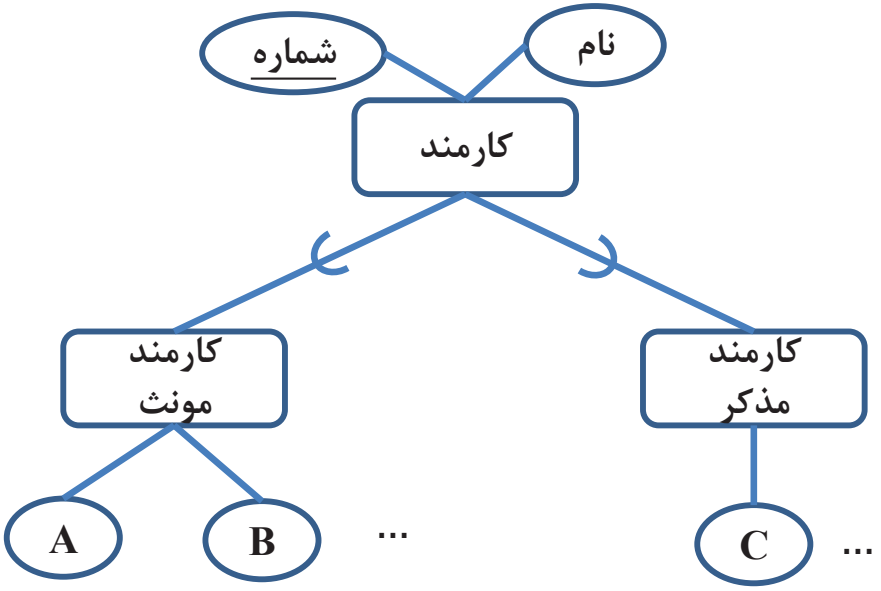
- طرز نوشتن: "F IS-A E"
- وقتی نوع های خاصِ یک نوع عام را بازشناسی می‌کنیم به آن تکنیک ویژه نمایی-تخصیص یا Specialization گوییم.
- عکس این تکنیک را تعمیم یا Generalization گوییم.

ارتباط "IS A" (ادامه)

مثال انواع موجودات زنده



مثال انواع کارمندان



ارتباط "IS A" (ادامه)

نکات:

- **زیرنوع** مجموعه صفاتی دارد مشترک در تمام **زیرنوع‌ها**
  - درنتیجه **زیرنوع** تمام صفات **زیرنوع** را به ارث می‌برد (وراثت صفات از نوع ساختاری).
  - مفهوم ارث‌بری با تکنیک ارتباط IS-A مدلسازی می‌شود.
  - 

وراثت ممکن است ساختاری باشد یا رفتاری. در اینجا وراثت صفات، وراثتی ساختاری است.
- **زیرنوع** مجموعه صفات خاص خود را هم دارد [حداقل یک صفت]
- اگر m تعداد شاخه های تخصیص منشعب از یک **زیرنوع** باشد داریم: $\boldsymbol{m \geq 1}$

ارتباط "IS A" – تخصیص

تخصیص

۱- **کامل:** تمام زیرنوع‌های (ممکن) زبرنوع در مدلسازی در نظر گرفته می‌شوند. بدین ترتیب هر نمونه از زبرنوع، جزء نمونه‌های حداقل یکی از زیرنوع‌ها است.

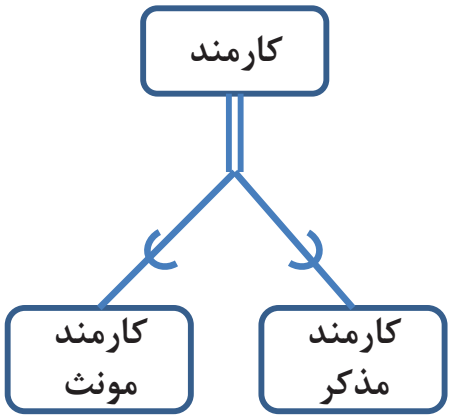
۲- **ناقص:** تمام زیرنوع‌های (ممکن) زبرنوع در مدلسازی در نظر گرفته نمی‌شوند. هر نمونه از زبرنوع لزوما جزء نمونه‌های یکی از زیرنوع‌ها نیست.

مثال

تخصیص ناقص: براساس مهارت کارمند فقط برنامه‌سازان را جدا کرده‌ایم. ممکن است کارمندی باشد که برنامه‌ساز نباشد.

مثال

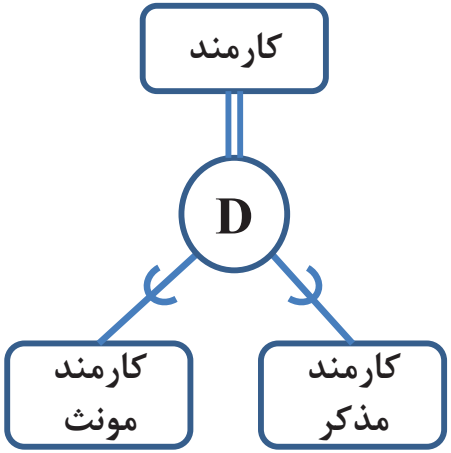
تخصیص کامل: هر نمونه کارمند یا مونث است یا مذکر.



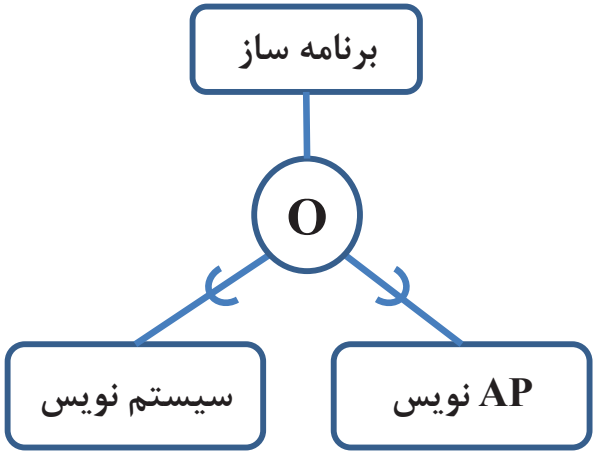
ارتباط "IS A" – تخصیص (ادامه)

- تخصیص
  - ۱- **مجزا:** یک نمونه از **زیرنوع** جزء مجموعه نمونه‌های حداکثر یک **زیرنوع** است.
  - ۲- **همپوشا:** یک نمونه از **زیرنوع** جزء مجموعه نمونه‌های حداقل دو **زیرنوع** است.

مثال تخصیص مجزا

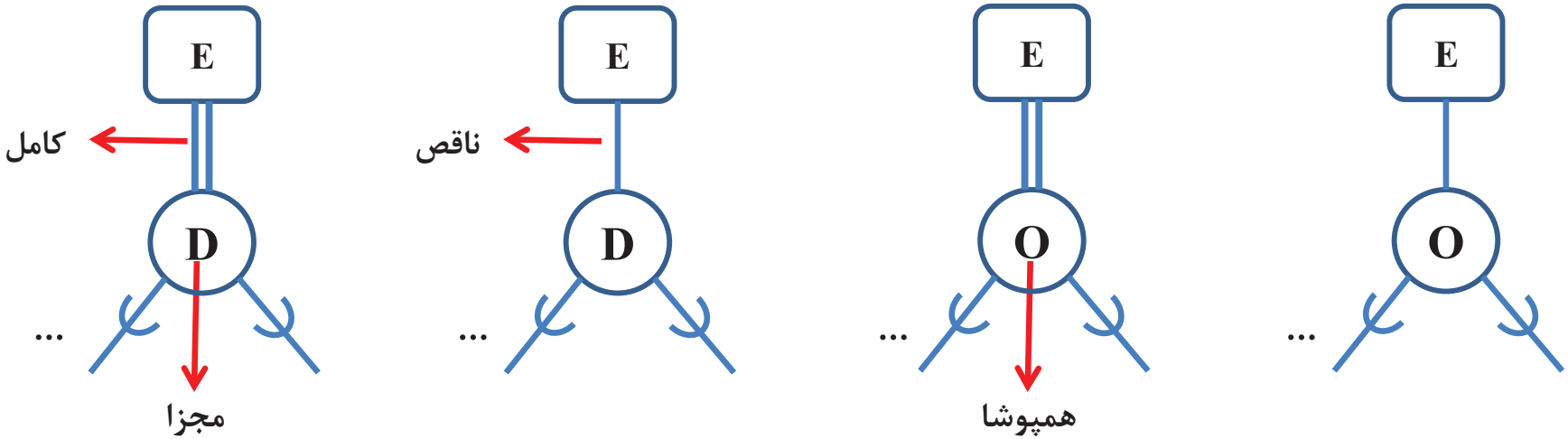


مثال تخصیص همپوشا



ارتباط "IS A" – تخصیص (ادامه)

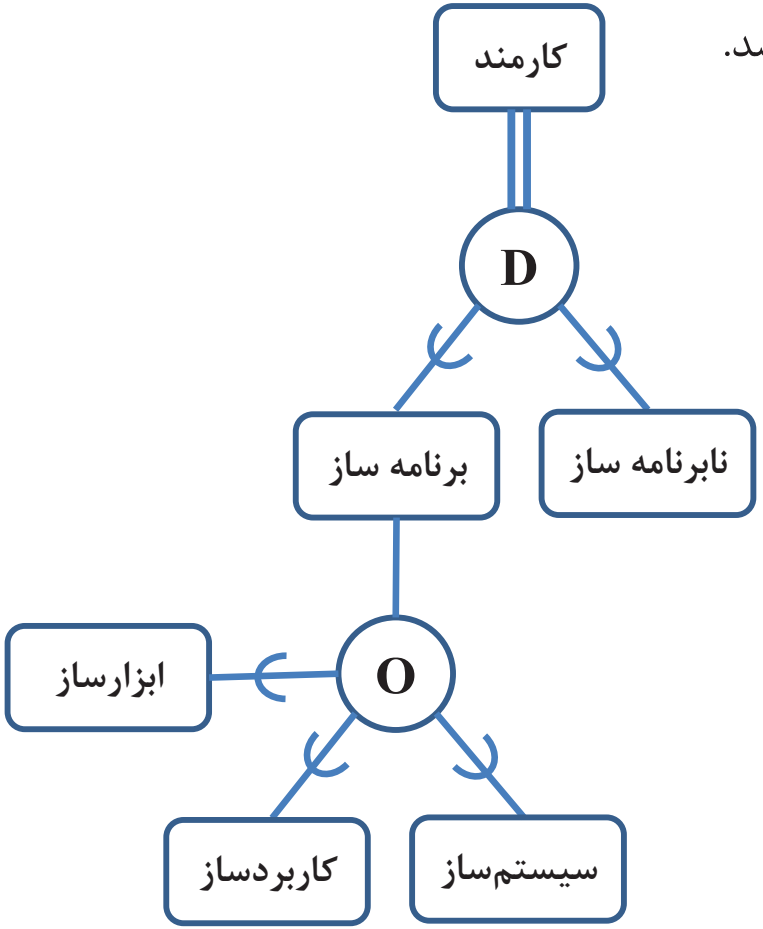
- براساس این دو ویژگی چهارگونه تخصیص داریم:



ارتباط "IS A" (ادامه)

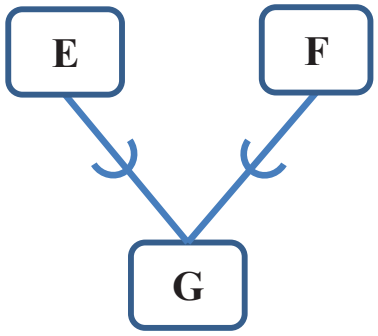
ادامه نکات:

- **زیرنوع** می‌تواند خود **زیرنوع‌هایی** داشته باشد.
  - یعنی ژرفای (عمق) درخت تخصیص می‌تواند بیش از یک باشد.



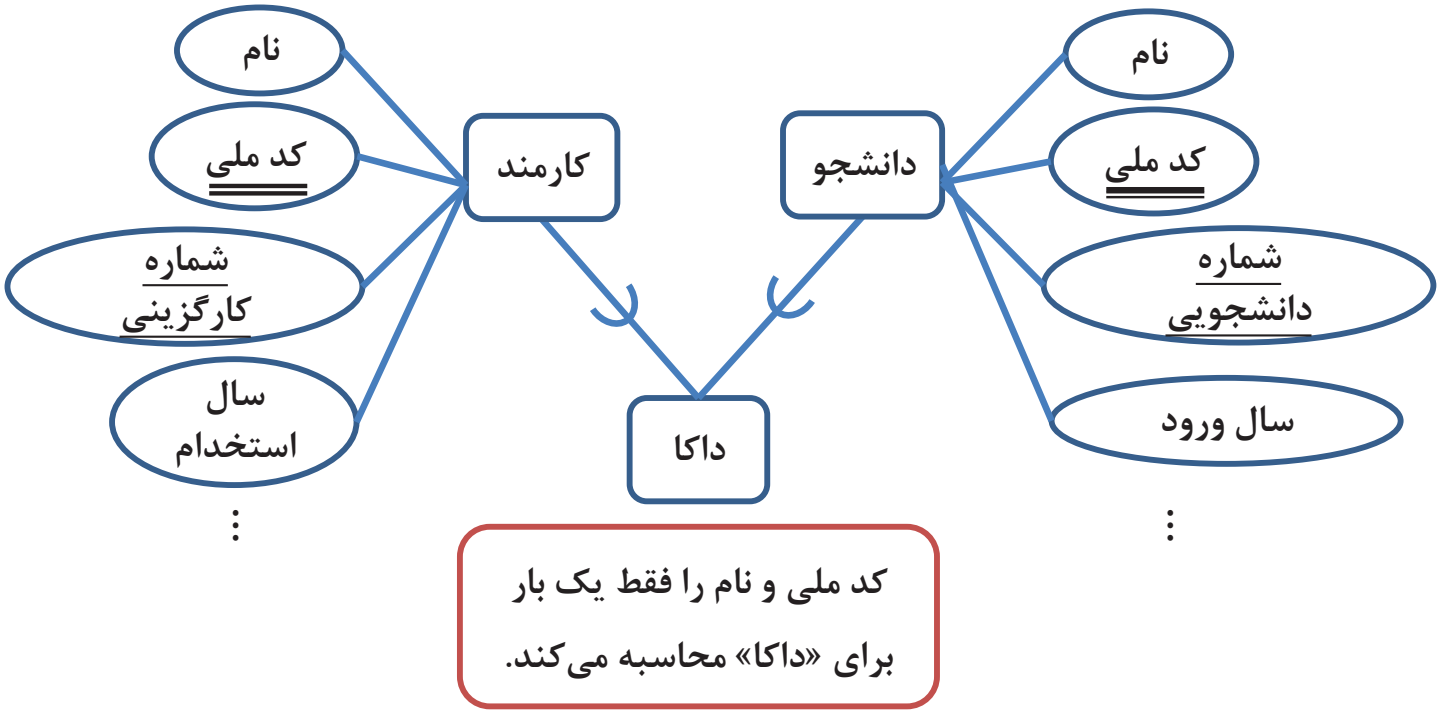
ارتباط "IS A" (ادامه)

- **زیرنوع** می‌تواند بیش از یک **زبرنوع** داشته باشد.
- G صفات را هم از E و هم صفات F را به ارث می‌برد
- **وراثت چندگانه (Multiple Inheritance)** را می‌توان اینگونه مدل کرد.



آیا G می‌تواند از خود نیز صفاتی داشته باشد؟

مثال ارث‌بری چندگانه



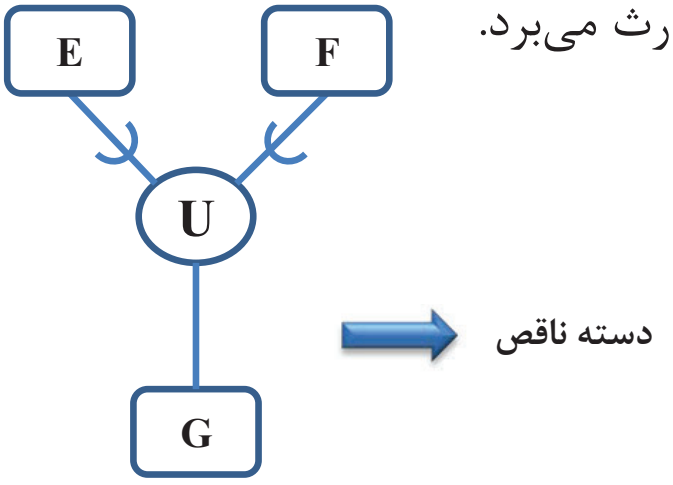
زیرنوع اجتماع (U-Type)

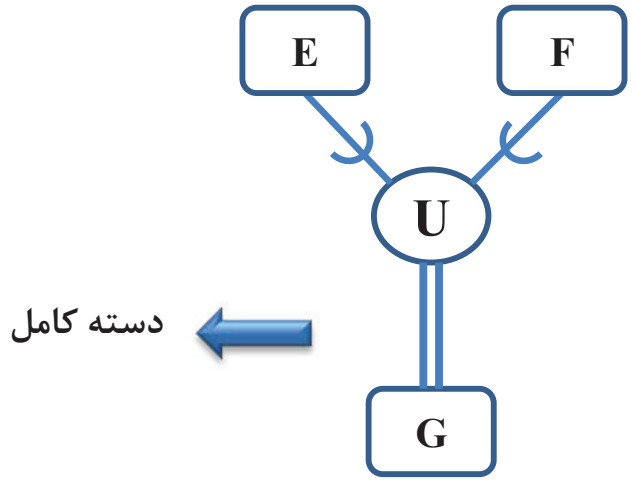
- **زیرنوع اجتماع (U-Type)** یا Category «دسته»
- **زیرنوع** موجودیت G را زیرنوع U-Type **زیرنوع‌های** E,F,... گوییم هرگاه در مجموعه نمونه‌های G نمونه‌هایی از E,F,... وجود داشته باشد. در واقع نمایانگر اجتماعی از نمونه‌ها از انواع مختلف است.

نمونه‌هایی

- اگر همه نمونه‌ها ← دسته کامل
- اگر بعض نمونه‌ها ← دسته ناقص

- یک نمونه از زیرنوع اجتماع (دسته)، بسته به اینکه از نوع کدام زبرنوع باشد، صفات همان زبرنوع را به ارث می‌برد.

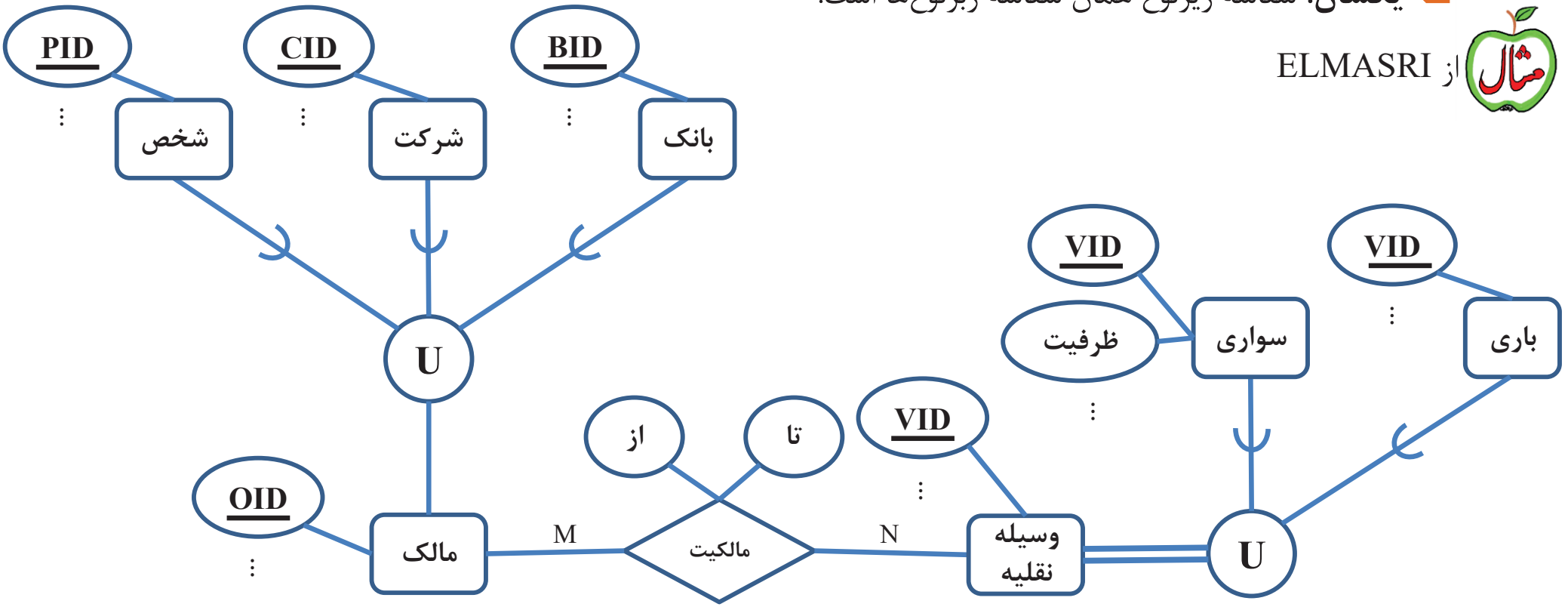




زیرنوع اجتماع (ادامه)

- شناسه های زبرنوع ها می‌تواند از دامنه های متفاوت باشد.
- **متفاوت:** شناسه زیرنوع شناسه ای است که خود باید در نظر بگیریم.
- **یکسان:** شناسه زیرنوع همان شناسه زبرنوع‌ها است.

مثال از ELMASRI



کنجاوی در چه صورت مدلسازی با U-Type را می‌توان با تکنیک تخصیص (ویژه‌نمایی) معمولی مدل کرد؟ در چه شرایطی کدام یک بهتر است؟

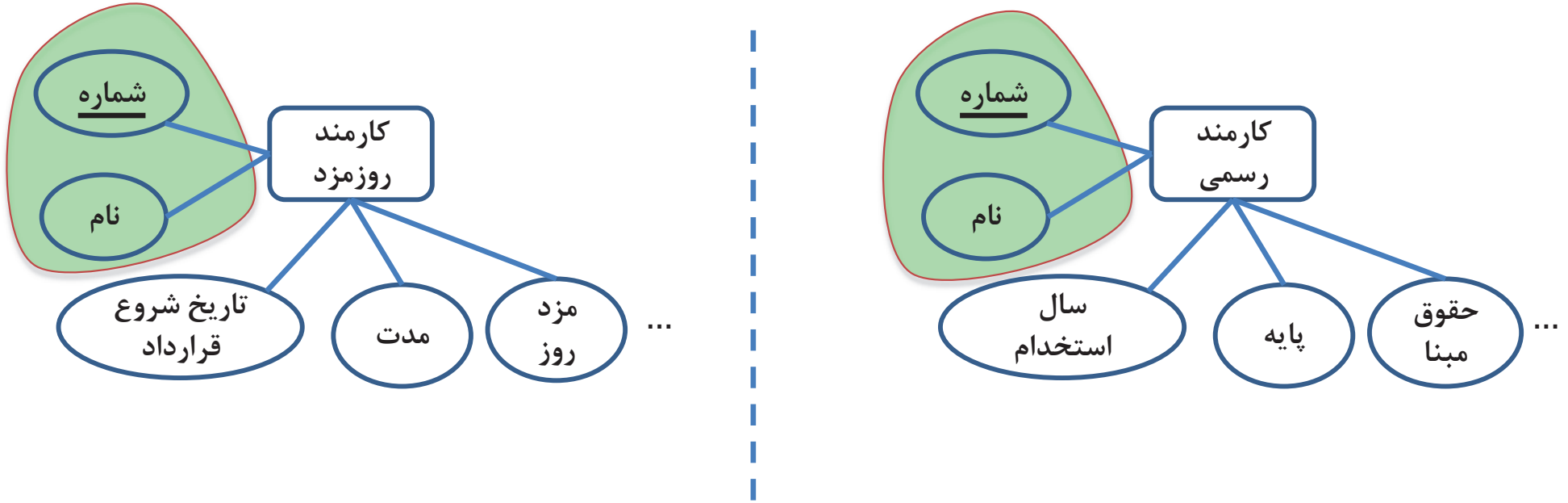
زیرنوع اجتماع (ادامه)

- تمرین : برای محیط با مفاهیم زیر، هم با U-Type و هم بدون U-Type یک مدلسازی ارایه دهید:
  - بانک - دانشگاه
  - شخص (دانشجو – استاد – کارمند و متفرقه)
  - حساب بانکی (کوتاه مدت – بلند مدت – قرض الحسنه و...)
  - عملیات واریز – برداشت - انتقال وجه

تعمیم (Generalization)

- **تعمیم** عبارت است از تشخیص یک نوع موجودیت جدید (در سطح انتزاع بالاتر) از روی [با داشتن] $n \geq 2$ نوع موجودیت از پیش دیده که ماهیتا از یک نوع باشند. (احیانا به منظور ادغام ERDهای جدا)

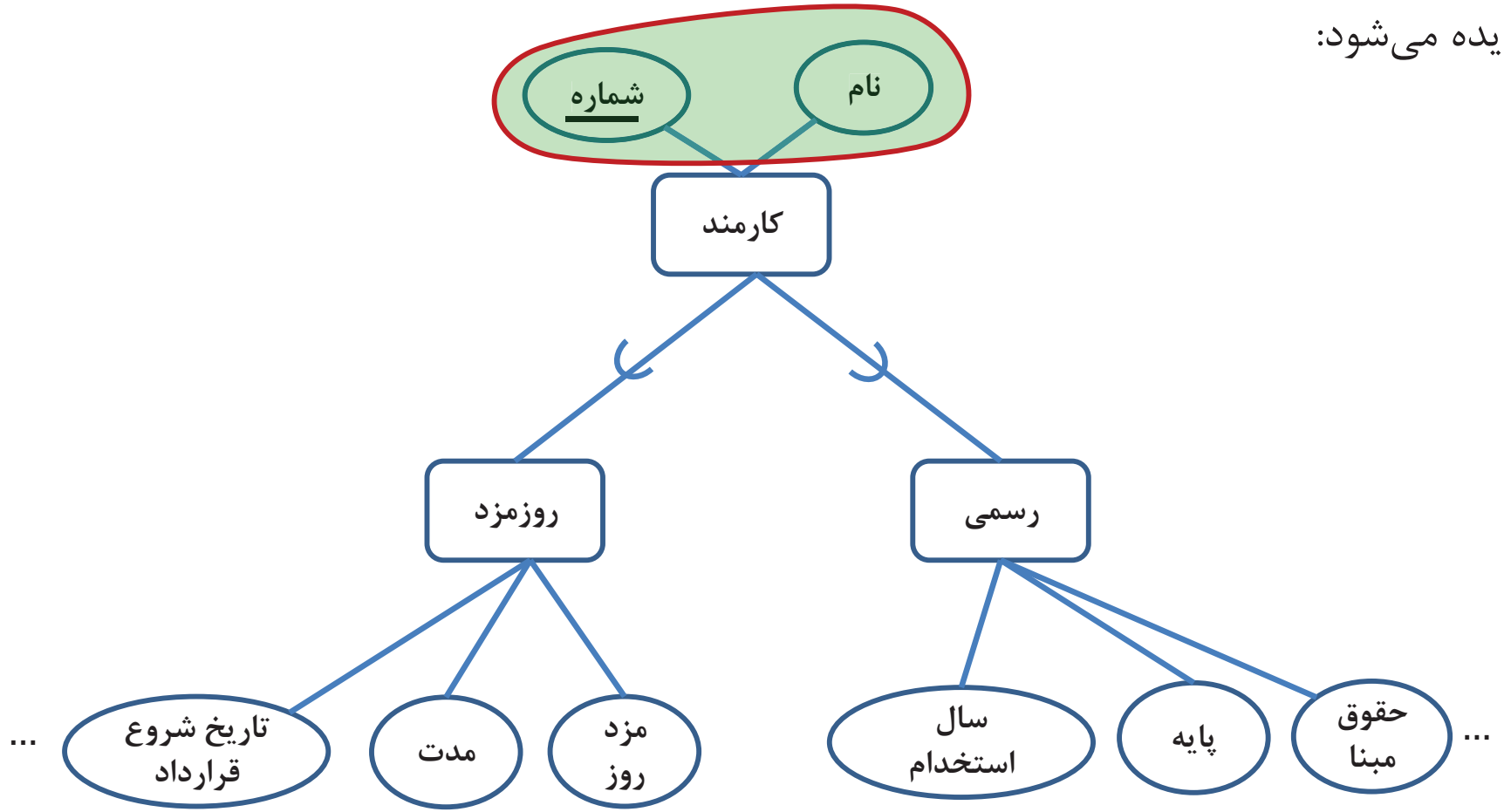
مثال: فرض: در یک مدلسازی یا در دو مدلسازی جدا برای دو زیر محیط:



مثال ادامه:

- یک نوع موجودیت (کارمند) در سطح انتزاعی

بالاتر دیده می‌شود:



شرایط تعمیم:

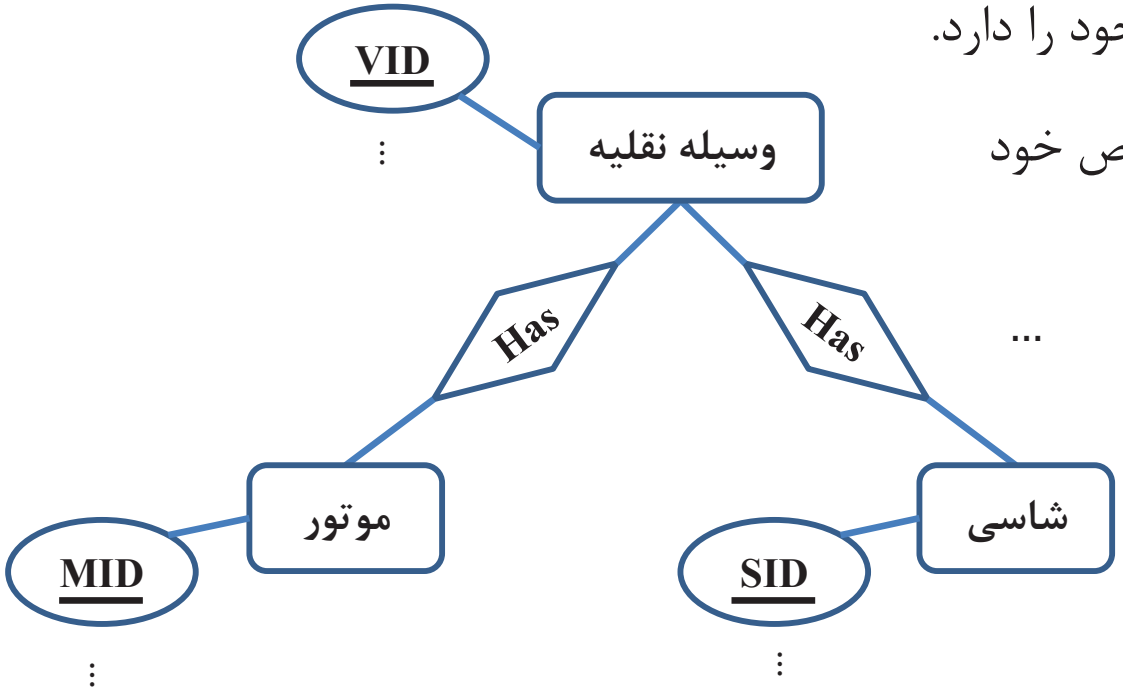
- داشتن شناسه مشترک [یعنی از یک دامنه]
- حداقل وجود دو نوع زیرنوع
- هرچه صفات مشترک بیشتر، تعمیم توجیه پذیرتر است [شرطِ لازم نیست ولی شرطِ ارجحیت است].

ارتباط‌ها؟

ارتباط "IS-A-PART Of" یا "Has-A" یا "Contains"

- **تعریف:** ارتباط بین نوع موجودیت کل است با نوع موجودیت‌های جزء آن (تشکیل دهنده آن)
- **F** is a part of **E**
- E شامل F است.
- E دارد F.
- نکته: نوع کل مجموعه صفات خاص خود را دارد.
- نکته: نوع جزء هم مجموعه صفات خاص خود را دارد [از جمله شناسه].

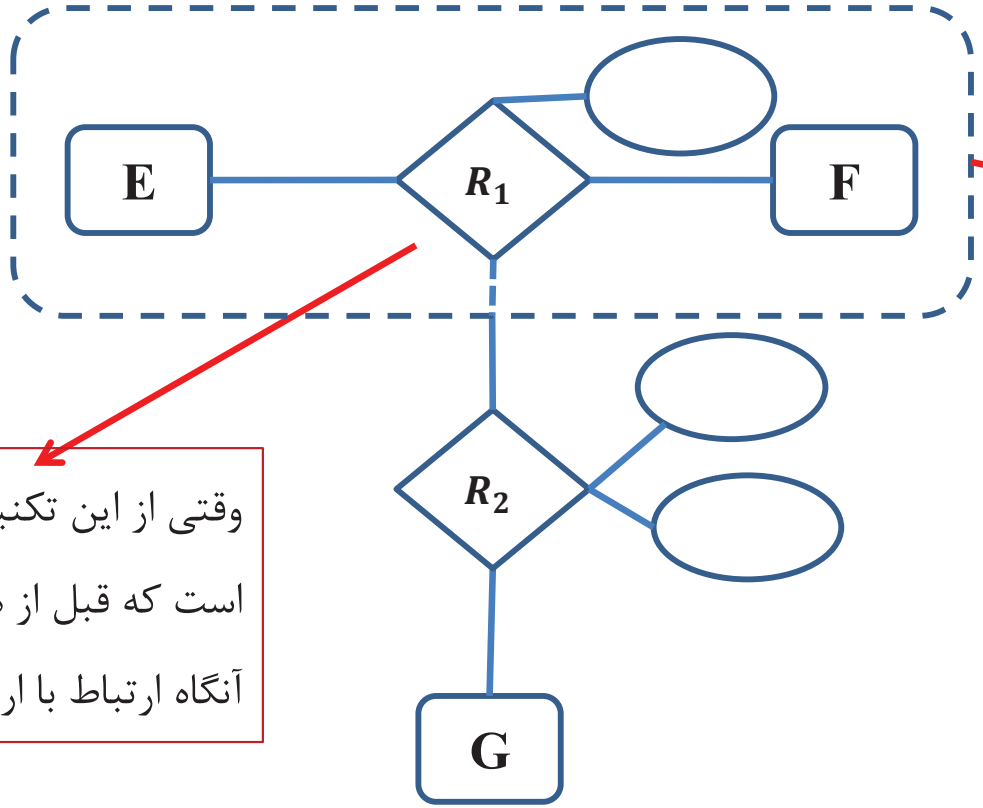
مثال ارتباط شاسی و موتور با وسیله نقلیه



ارتباط "IS-A-PART Of" (ادامه)

- **تفاوت های نوع ضعیف با نوع جزء:**
  - نوع جزء از خود شناسه دارد ولی نوع ضعیف نه.
  - با حذف نوع کل لزوما نوع جزء حذف نمی‌شود (به عبارتی وابستگی وجودی لزوما نداریم.)
  - ...؟
- در ارتباط "IS-A-PART Of" ←
  - **تکنیک تجزیه:** دیدن نوع موجودیت‌های جزء از روی نوع موجودیت کل
  - **تکنیک ترکیب:** دیدن نوع موجودیت کل از روی اجزاء

- **تکنیک تجمیع (Aggregation):** دیدن $N \geq 1$ نوع موجودیت شرکت کننده در ارتباط R، به صورت یک نوع موجودیت **انتزاعی**: به منظور مدلسازی ارتباط با ارتباط (به ویژه زمانی که نوع ارتباط R صفاتی هم داشته باشد).
- ارتباط با ارتباط حیطه معنایی خاص خود را دارد.

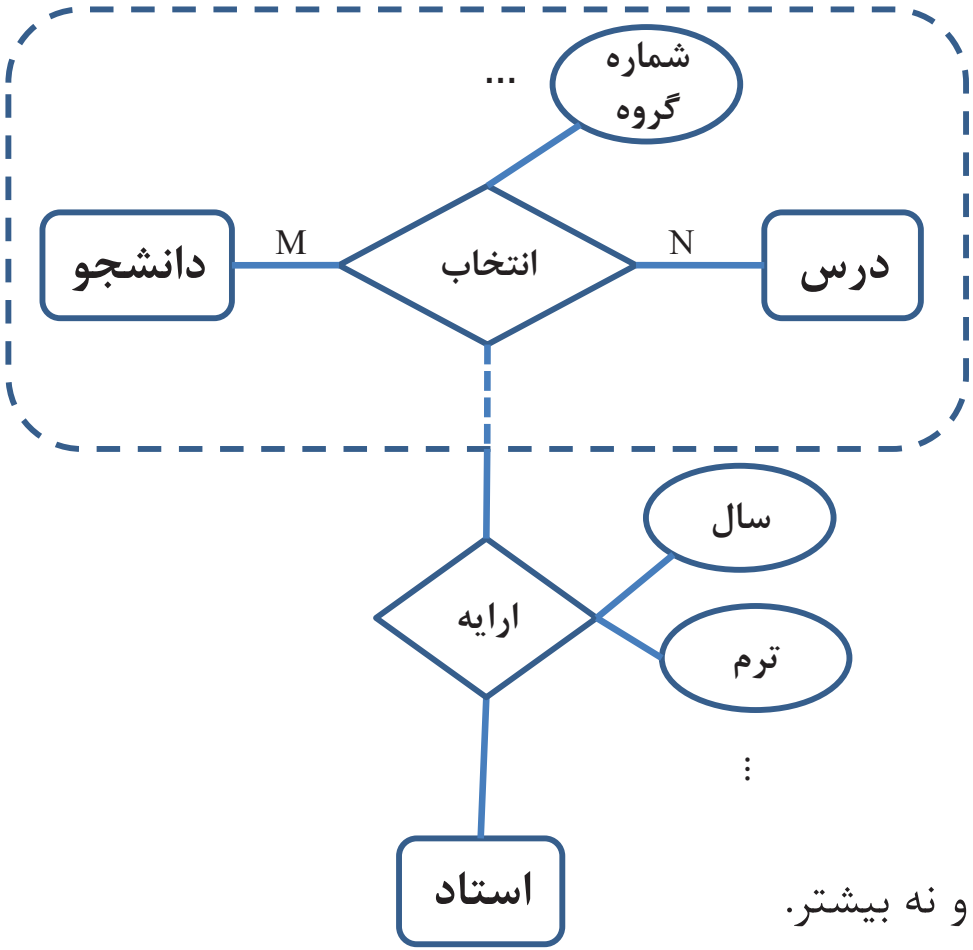


دیدن موجودیتهای دخیل در ارتباط R_1 به صورت یک موجودیت انتزاعی

وقتی از این تکنیک استفاده می‌شود، معنایش این است که قبل از هر چیز به ارتباط R_1 نیاز است. آنگاه ارتباط با ارتباط مطرح شده است.

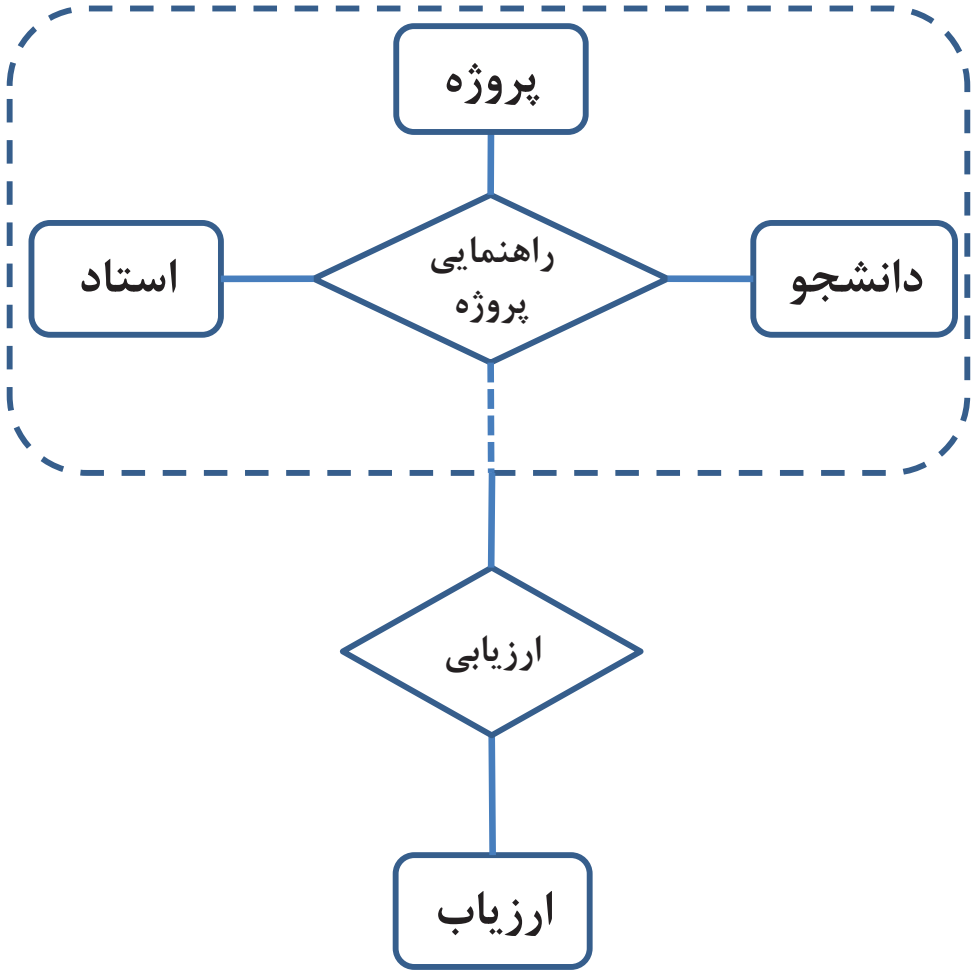
- معمولا از این تکنیک زمانی استفاده می‌شود که چندی ارتباط M:N باشد.  چرا ؟


طرز دیگر مدلسازی برای برای محیط دانشجو – درس – استاد:



- **نکته:** هر Aggregation برای یک ارتباط است و نه بیشتر.

مثال ارزیابی راهنمایی پروژه پژوهشی دانشجو توسط استاد



به نام آنکه جان را فکرت آموخت

بخش سوم :
طراحی منطقی

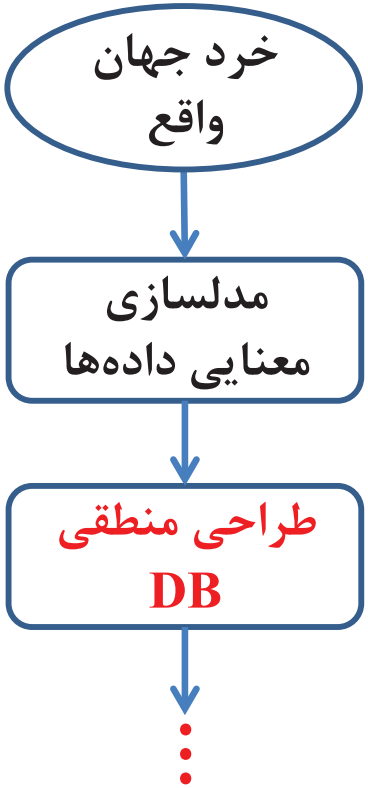
مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

طراحی منطقی DB

- مدلسازی داده‌ها می‌تواند در سطوح انتزاعی مختلفی صورت پذیرد.
- سطح پایین‌تر از سطح مدلسازی معنایی داده‌ها، سطح طراحی منطقی است.



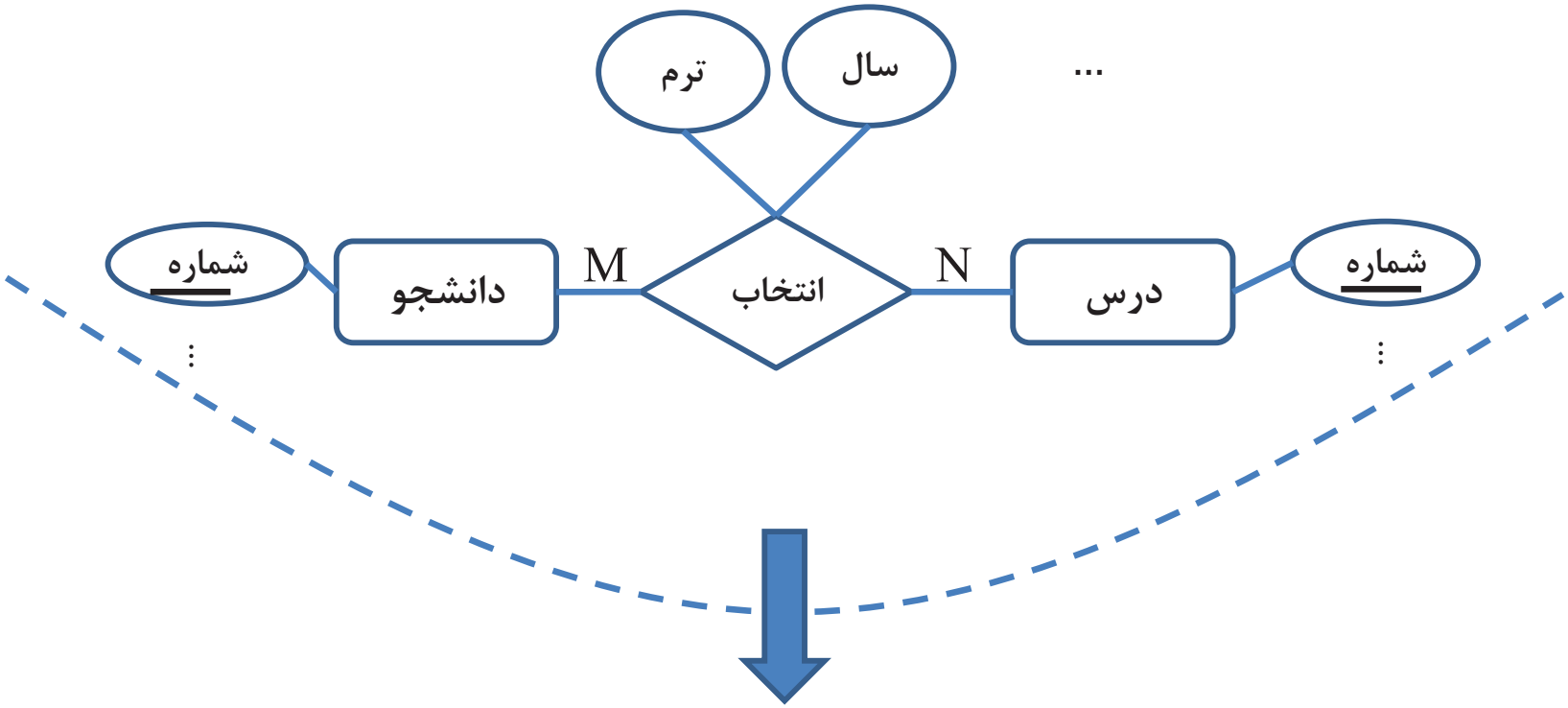
- **سطح طراحی منطقی:** برای نمایش پایگاه داده‌ها در این سطح از مفاهیمی استفاده می‌شود که مستقل از مفاهیم محیط فایلینگ پایگاه داده‌ها است.

طراحی منطقی DB (ادامه)

- بحث مقدماتی: دیدگاه کاربردی [و نه تئوریک]
- برای طراحی منطقی پایگاه داده‌ها (و همچنین عملیات در DB و کنترل DB) هم امکان خاصی لازم است: یک **مدل داده (DM)**، که شامل یک **ساختار داده (DS)** است.
- مفاهیم مطرح در طراحی منطقی پایگاه داده‌ها
  - ساختار داده جدولی : TDS
  - پایگاه داده جدولی : TDB
  - زبان پایگاهی جدولی : TDBL

طراحی منطقی با TDS – رابطه چند به چند

مثال چندی M:N



مساله: تبدیل به TDB [با TDS]

- سه نوع جدول لازم داریم:
  - برای هر نوع موجودیت یک نوع جدول
  - برای نوع ارتباط M:N یک نوع جدول

طراحی منطقی با TDS – رابطه چند به چند (ادامه)

خط زیرین نمایانگر کلید

STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	ms	phys	d11
⋮	⋮	⋮	⋮	⋮

COT

COID	COTITLE	CREDIT	COTYPE	CEDEID
⋮	⋮	⋮	⋮	⋮
co3	programming	4	t (تئوری)	d13
⋮	⋮	⋮	⋮	⋮

طراحی منطقی با TDS – رابطه چند به چند (ادامه)

طبق قواعد معنایی محیط ممکن است سال و ترم هم جزو کلید باشند.

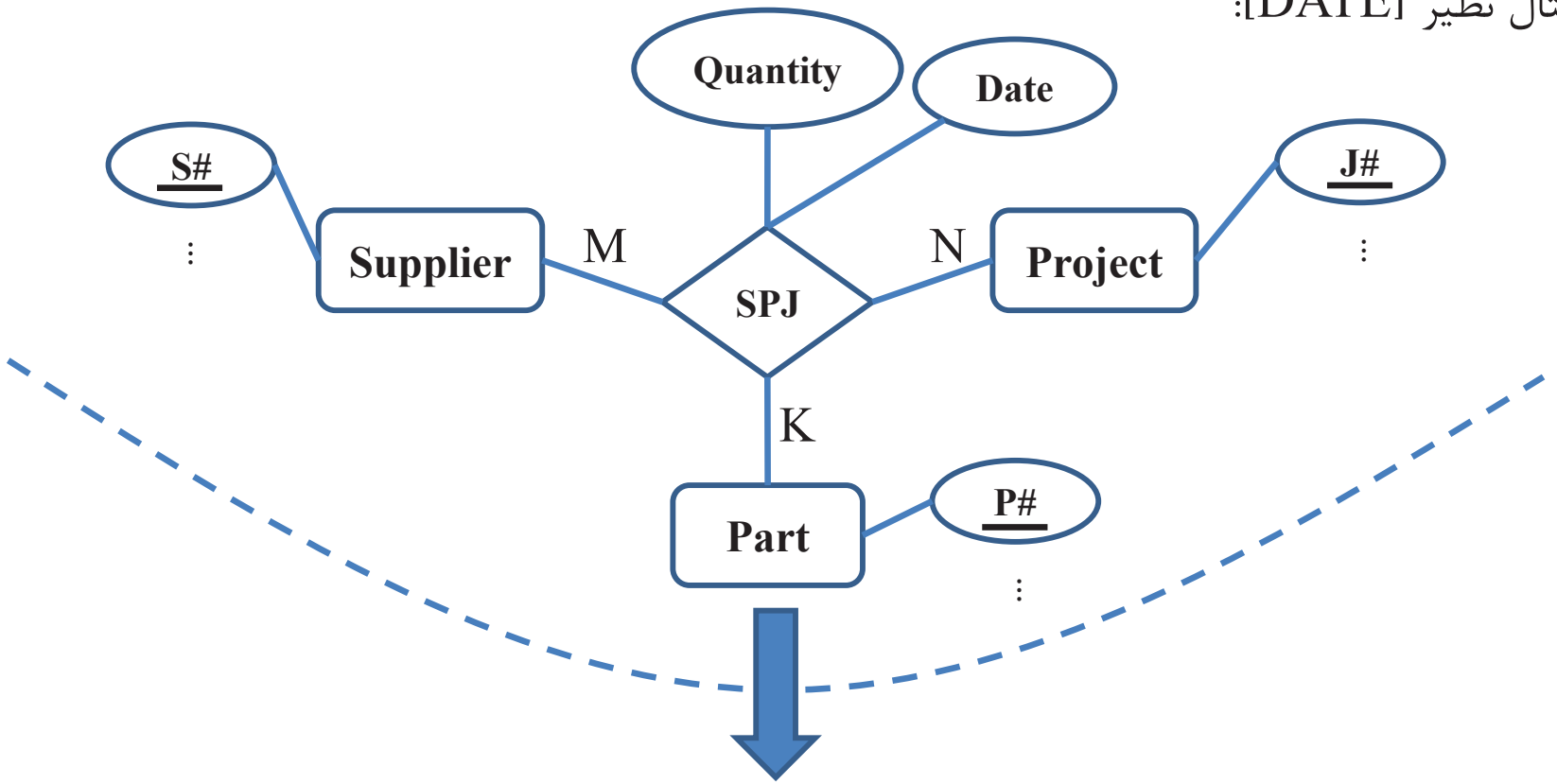
(در واقع اگر صفت چند مقداری مرکب برای رابطه باشند، جزو کلید محسوب می‌شوند.)

STCOT

STID	COID	TR	YR
⋮	⋮	⋮	⋮
888	co2	1	87
888	co3	1	87
444	co2	1	87

طراحی منطقی با TDS – رابطه چند به چند (ادامه)

مثال نظیر [DATE]:



مساله: تبدیل به TDB [با TDS]

- چهار نوع جدول داریم: ← برای هر نوع موجودیت یک نوع جدول / برای نوع ارتباط یک نوع جدول

طراحی منطقی با TDS – رابطه چند به چند (ادامه)

طبق قواعد معنایی محیط ممکن است تاریخ هم جزو کلید بشود.

(در واقع اگر صفت چند مقداری باشد، جزو کلید محسوب می‌شود.)

Supplier

S#	SNAME	CITY	…
s1	…	c1	…
s2	…	c1	…
⋮	⋮	⋮	⋮

Part

P#	PNAME	CITY	…
p1	…	c1	…
p2	…	c2	…
⋮	⋮	⋮	⋮

Project

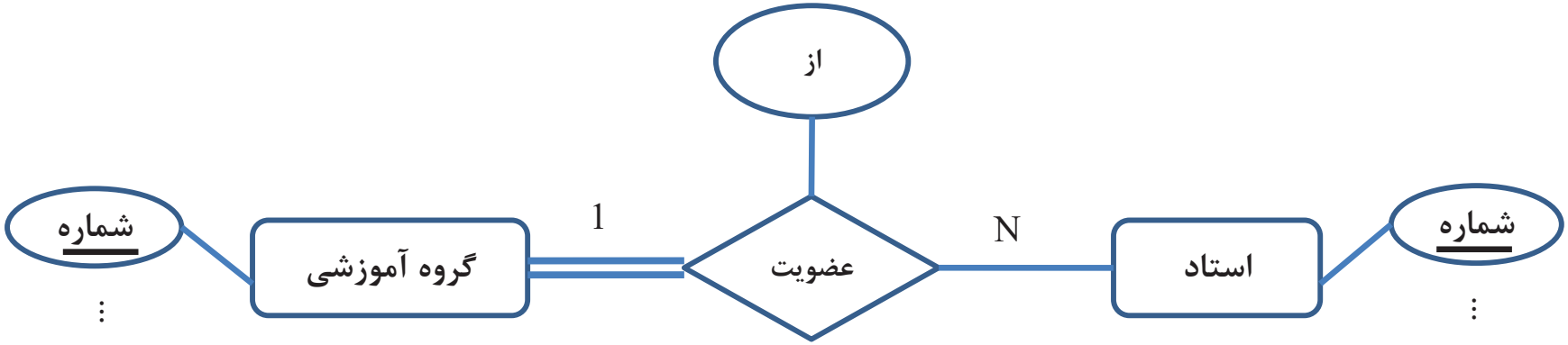
J#	JNAME	CITY	…
j1	…	c2	…
j2	…	c1	…
⋮	⋮	⋮	⋮

SPJ

S#	P#	J#	Date	QTY
s1	p1	j1	d1	100
s1	p1	j1	d2	50
⋮	⋮	⋮	⋮	⋮

طراحی منطقی با TDS – رابطه یک به چند

مثال چندی 1:N



- دو نوع جدول داریم:
  - یکی برای نوع موجودیت سمت 1
  - یکی برای نوع موجودیت سمت N و نیز خود ارتباط

طراحی منطقی با TDS – رابطه یک به چند (ادامه)

DEPT

DEID	DETITLE	…	DEPHONE
D11	Phys	…	…
D12	Math	…	…
⋮	⋮	⋮	⋮

PROF

PRID	PRNAME	RANK	…	FROM	DEID
Pr100	…	استاد	…	d1	D13
Pr200	…	استادیار	…	d2	D11
Pr300	…	دانشیار	…	?	?

* ستون DEID در جدول PROF **کلید خارجی** است و با خط‌چین مشخص می‌شود.

تعریف **کلید خارجی**[کاربردی]: ستون c از جدول T1 در جدول T2 کلید خارجی است هرگاه در جدول T1 کلید اصلی باشد.

کنکاوی؟ در چه حالاتی استفاده از سه نوع جدول قابل توجیه است؟

طراحی منطقی با TDS – رابطه یک به یک

مثال چندی 1:1

- دو نوع جدول داریم:
  - یکی برای نوع موجودیت سمت 1 غیرالزامی
  - یکی برای نوع موجودیت سمت 1 الزامی و نیز خود ارتباط

طراحی منطقی با TDS – رابطه یک به یک (ادامه)

- یک طرز طراحی ممکن:

DEPT

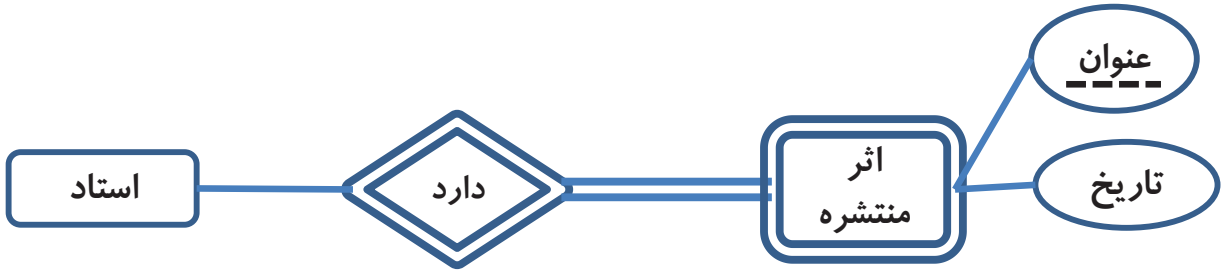
DEID	DETITLE	…	DEPHONE	PRID
D11	Phys	…	…	…
D12	Math	…	…	…
⋮	⋮	⋮	⋮	⋮

PROF

PRID	PRNAME	RANK	…
Pr100	…	استاد	…
Pr200	…	استادیار	…
Pr300	…	دانشیار	…
⋮	⋮	⋮	⋮

طرزهای دیگر طراحی؟

مثال رابطه شناسا (رابطه موجودیت ضعیف)



- دو نوع جدول داریم:
  - یکی برای نوع موجودیت قوی
  - یکی برای نوع موجودیت ضعیف و رابطه (حاوی شناسه موجودیت قوی)

طراحی منطقی با TDS – رابطه شناسا (ادامه)

PROF

PRID	PRNAME	RANK	...
Pr100	...	استاد	...
Pr200	...	استادیار	...
Pr300	...	دانشیار	...
⋮	⋮	⋮	⋮

PUB

PRID	PTITLE	...	PDATE
Pr100	Data Encryption…	...	...
Pr100	Semantic Analysis of …	...	...
⋮	⋮	⋮	⋮

* دو صفت PRID (کلید خارجی از جدول PROF) و TITLE، کلید اصلی جدول انتشارات را تشکیل می‌دهند.

حذف و بروزرسانی در جدول PROF چه تاثیری بر PUB باید داشته باشد.

مثال رابطه IS-A

- دو نوع جدول داریم:
  - یکی برای زبرنوع موجودیت (حاوی صفات عام یا مشترک)
  - یکی برای نوع زیرنوع موجودیت (حاوی صفات خاص زیرنوع و شناسه زبرنوع)

طراحی منطقی با TDS – رابطه IS-A (ادامه)

EMP

EID	ENAME	EBDATE	…	EPHONE
E100	…	…	…	…
E101	…	…	…	…
E102	…	…	…	…
⋮	⋮	⋮	⋮	⋮

PROG

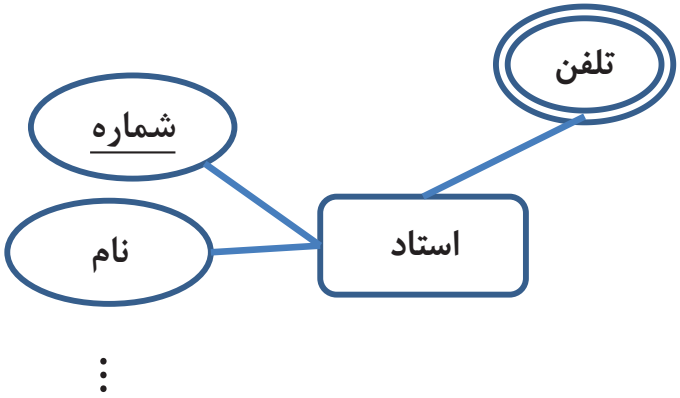
EID	LANG	…	LEVEL
E100	C++	…	…
E102	Java	…	…
⋮	⋮	⋮	⋮

* EID (کلید خارجی از جدول EMP) کلید اصلی جدول PROG نیز هست.

حذف و بروزرسانی در جدول EMP چه تاثیری بر PROG باید داشته باشد (و بالعکس)؟

طراحی منطقی با TDS – صفت چندمقداری

مثال صفت چندمقداری



- دو نوع جدول داریم:
  - یکی برای نوع موجودیت (حاوی صفات تک‌مقداری)
  - یکی برای صفت (ساده یا مرکب) چندمقداری

طراحی منطقی با TDS – صفت چندمقداری (ادامه)

PROF

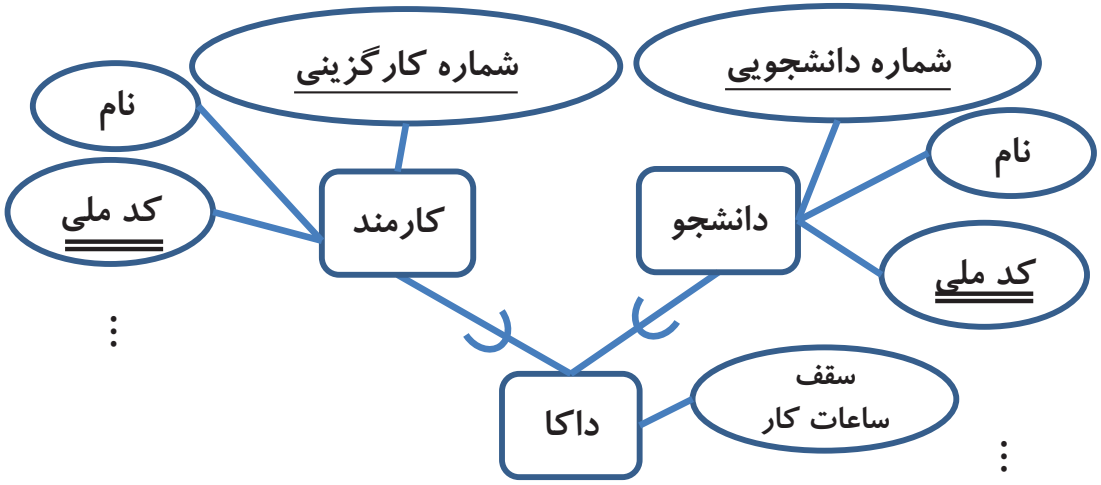
PRID	PNAME	RANK	…	
Pr100	…	…	…	…
Pr101	…	…	…	…
Pr102	…	…	…	…
⋮	⋮	⋮	⋮	⋮

PROFTEL

PRID	TEL
Pr100	09121234567
Pr100	02177889911
Pr101	09352348762
⋮	⋮

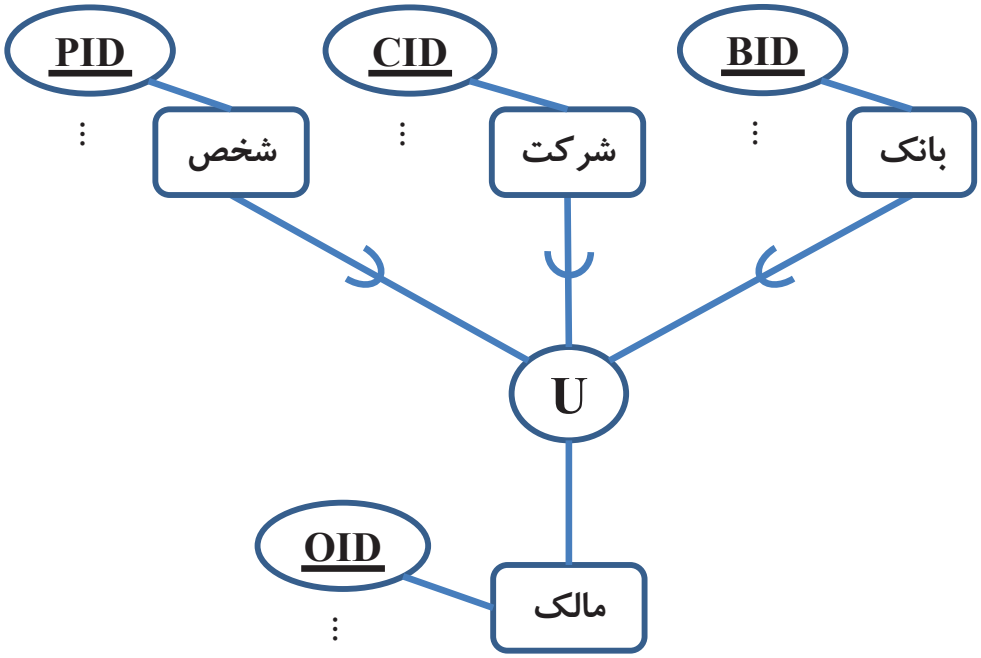
طراحی منطقی با TDS (ادامه)

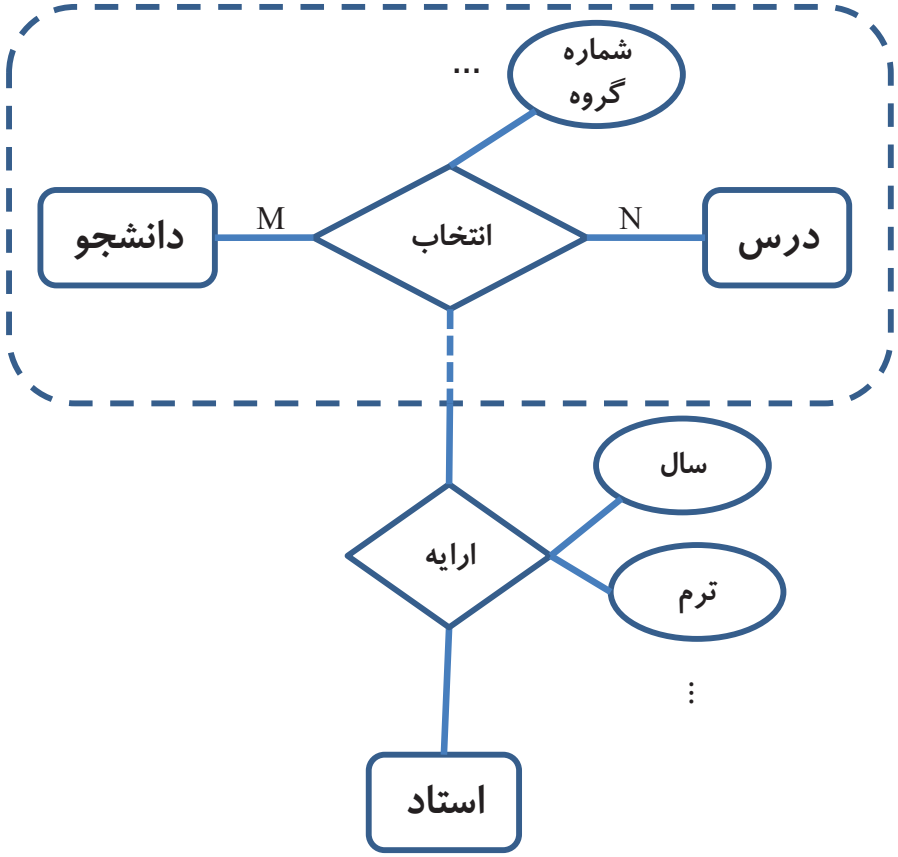
- **تمرین:** TDB را برای مدلسازی‌های زیر طراحی کنید.



طراحی منطقی با TDS (ادامه)

- **تمرین:** TDB را برای مدلسازی‌های زیر طراحی کنید.





به نام آنکه جان را فکرت آموخت

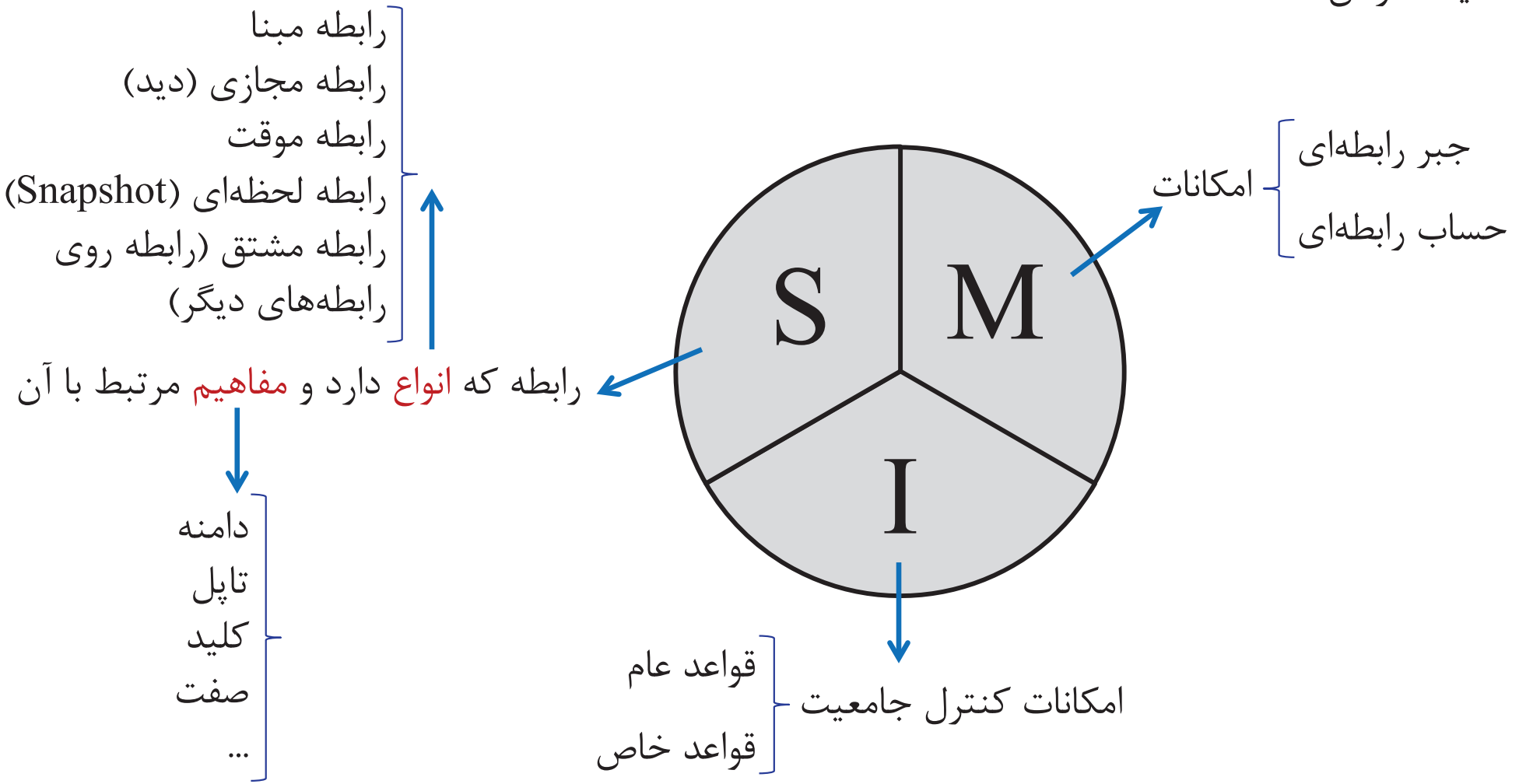
بخش ششم:
مفاهیم اساسی مدل داده رابطه‌ای

مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

✓ **مدل داده** مجموعه‌ای است از امکانات برای طراحی منطقی و تعریف پایگاه داده‌ها، کنترل آن و نیز انجام عملیات در آن.



رابطه

تعریف در ریاضی: هر زیر مجموعه از ضرب کارتزین چند مجموعه

تعریف (۱) با فرض وجود m مجموعه از مقادیر موسوم به دامنه [میدان] D_1، ...، D_m:

رابطه R با صفات A_1، ...، A_m تعریف شده روی این m دامنه

مجموعه‌ای است از عناصر، هر یک به صورت $\langle d_{1i}, d_{2i}, \ldots, d_{mi} \rangle$ موسوم به m-تاپل (m-tuple)

به نحوی که $d_{1i} \in D_1$، ...، $d_{ji} \in D_j$

مثال

STUD (STID, STNAME, STJ, STL, STD)

777	st7	bs	phys	d11
⋮	⋮	⋮	⋮	⋮
444	st4	bs	comp	d14

یک تاپل ۵-تایی

رابطه (ادامه)

تعریف (۲) [Date] با فرض وجود m مجموعه از مقادیر موسوم به دامنه [میدان] D_1، ...، D_m نه لزوماً متمایز، رابطه R تعریف شده روی این m دامنه:

دومجموعه:
- عنوان [سرآیند] (Heading): مجموعه‌ای است نامدار از اسامی صفات یعنی $\{A_1,\ldots,A_m\}$ که با $R(A_1,\ldots,A_m)$ نمایش داده می‌شود.
- بدنه [پیکر] (Body): مجموعه ای است از تاپل‌ها [همان مجموعه در تعریف اول].

مثال رابطه دانشجو

STUD (STID, STNAME, STJ, STL, STD)

- **درجه رابطه:** کاردینالیتی عنوان یا تعداد صفات رابطه

m	اصطلاح
۱	رابطه یگانی
۲	رابطه دوگانی
n	رابطه nگانی

رابطه (ادامه)

- **کاردینالیتی رابطه:** همان کاردینالیتی بدنه؛ تعداد تاپل‌ها (بزرگتر مساوی صفر؛ صفر در بدو تعریف)
  - بدنه رابطه، متغیر در زمان است.
  - به یک مقدار بدنه در یک لحظه مشخص instance گویند.
  - به بدنه رابطه، Extension (**بسط** یا گسترده) یا حالت رابطه گویند.

تناظر بین مفاهیم رابطه‌ای و اصطلاحات جدولی

مفهوم رابطه‌ای	اصطلاح
رابطه	جدول (صرفاً امکانی است برای نمایش مفهوم رابطه‌ای و تفاوت‌های متعددی با رابطه دارد.)
تاپل	سطر
صفت	ستون
دامنه	مقادیر مجاز ستون
درجه	تعداد ستون‌ها
کاردینالیتی	تعداد سطرها
کلید	؟ (به معنایی که در مدل رابطه‌ای داریم، در بحث‌های جدولی مطرح نیست.)

کلید در مدل رابطه‌ای

- اصطلاح **کلید**، یک اصطلاح عام است و گونه‌هایی دارد:

۱- سوپرکلید (اَبَر کلید): SK

۲- کلید کاندید (کلید نامزد): CK

۳- کلید اصلی: PK

۴- کلید بدیل: AK

۵- کلید خارجی: FK

کلید در مدل رابطه‌ای – سوپرکلید

- رابطه $R(\underbrace{A_1, A_2, \ldots, A_m}_{H_R})$ را در نظر می‌گیریم.

- **سوپرکلید (Super Key)**

تعریف: هر زیر مجموعه $S \subseteq H_R$ که یکتایی مقدار داشته باشد.

- اگر t_i و t_j دو تاپل دلخواه و متمایز از R باشند و $t_i(S) \neq t_j(S)$، آنگاه S یک سوپرکلید است.
- اگر N تعداد SKهای رابطه R باشد، $N \geq 1$ است، زیرا در بدترین حالت خود H سوپرکلید می‌شود.

چون بدنه، مجموعه است و تاپل تکراری نداریم.

$$1 \leq N \leq 2^m - 1$$

- **کاربرد سوپرکلید:**
  - در عمل، فاقد کاربرد مستقیم، در تئوری در بحث طراحی.
  - در SQL: با UNIQUE محدودیت یکتایی مقدار را اعمال می‌کنیم.

کلید در مدل رابطه‌ای – کلید کاندید

کلید کاندید (Candidate Key)

تعریف: هر زیرمجموعه $K \subseteq H_R$ که دو ویژگی داشته باشد:

۱- یکتایی مقدار

۲- کاهش‌ناپذیری (Irreducibility) یا کمینگی (Minimality)

- $K \subseteq H_R$ کاهش‌ناپذیر است هرگاه هر زیرمجموعه محض از K، خود یکتایی مقدار نداشته باشد.
- هر زیرمجموعه از H_R به نحوی که یک صفت را از آن حذف کنیم دیگر یکتایی مقدار نداشته باشد.

مثال:

رابطه	کلید کاندید
STT	STID
COT	COID
STCOT	(STID, COID)
S	S#
P	P#
SP	(S#, P#)

کلید در مدل رابطه‌ای – کلید کاندید (ادامه)

- CKها بر اساس قواعد معنایی محیط به دست می‌آیند.

دو حالت مختلف:

- هر کارمند در بیش از یک پروژه می‌تواند شرکت داشته باشد.

شماره‌ملی شماره‌پروژه شماره‌کارمند

EMPROJ (E#, J#, ENC, …)

CK: E#, J#

CK: ENC

- هر کارمند در حداکثر یک پروژه می‌تواند شرکت داشته باشد.

EMPROJ (E#, J#, ENC, …)

CK: E#

CK: ENC

کلید در مدل رابطه‌ای – کلید کاندید (ادامه)

خصوصیات کلید کاندید:

- هر CK، SK هم هست ولی عکس این مطلب صادق نیست.
- هر رابطه حداقل یک CK دارد، زیرا در بدترین حالت، خود H_R می‌شود CK.
- رابطه می‌تواند بیش از یک CK داشته باشد.

- رابطه R حداکثر چند CK دارد؟

- بیشترین تعداد CK زمانی است که به اندازه نصف تعداد صفات رابطه در CK شرکت کنند.
- CKهای رابطه می‌توانند همپوشا باشند، یعنی حداقل در یک صفت مشترک باشند.
- بنابراین اگر رابطه از درجه m باشد، بیشترین تعداد CK: $C_n^m = \frac{m!}{n!(m-n)!}$ به نحوی که $n = \left\lceil \frac{m}{2} \right\rceil$.

کلید در مدل رابطه‌ای – کلید کاندید (ادامه)

- **نقش کلید کاندید:** تضمین‌کننده عملیات تاپلی (و نه مجموعه‌ای) یا امکان ارجاع به تک تاپل در رابطه را فراهم می‌نماید.
- هر **زیر**مجموعه از CK، یک SK است (تفاوتشان در این است که CK با کمترین تعداد صفات یکتایی مقدار را می‌دهد).
- CK(های) رابطه باید به سیستم معرفی شوند.

مثال

```
CREATE RELATEION EMPROJ
        (E#   …  NOT NULL,
         J# …  NOT NULL,
         ENC …  NOT NULL)

CANDIDATE KEY  (E#, J#)
CANDIDATE KEY (J#, ENC)
```

- تئوری این را می‌گوید ولی در عمل، پکیج‌ها نمی‌پذیرند.

کلید در مدل رابطه‌ای – کلید اصلی

کلید اصلی (Primary Key)

کلید اصلی (PK) یکی از CKها است به انتخاب طراح.

در عمل با عبارت PRIMARY KEY تعریف می‌شود.

ضوابط انتخاب کلید اصلی:

۱- شناسه رایج در محیط باشد.

۲- مقادیرش همیشه معلوم باشد (نه هر CK، آنکه به عنوان PK انتخاب می‌شود)

۳- کوتاه‌تر بودن طول

۴- حتی‌الامکان مقادیرش تغییر نکند.

کلید در مدل رابطه‌ای – کلید بدیل

کلید بدیل (Alternate Key)

به هر کلید کاندید (CK) غیر از کلید اصلی (PK)، کلید بدیل (AK) گویند.

- در عمل متناظر ندارد.
- اگر N تعداد AKهای رابطه R باشد، داریم $N >= 0$.

ممکن است فقط یک CK داشته باشیم که آن هم می‌شود PK و دیگر AK نداریم.

کلید در مدل رابطه‌ای – کلید خارجی

کلید خارجی (Foreign Key)

- *در عمل:* $T_2.C$ در T_2، کلید خارجی است هرگاه در T_1، کلید اصلی باشد.
- *در تئوری:* صفت (ساده یا مرکب) $R_2.A_i$ در R_2 کلید خارجی است، هرگاه در R_1، نه لزوماً متمایز از R_2، کلید کاندید (CK) باشد.
- صفت (صفات) کلید خارجی باید هم‌میدان با صفت (صفات) کلید کاندید باشد و معمولا هم‌نام با کلید کاندید است، ولی گاه لازم می‌شود که نام دیگری داشته باشد.

مثال

رابطه	کلید خارجی	دلیل: CK در
STCOT	STID	STT
STCOT	COID	COT
SPJ	S#	S
SPJ	P#	P
SPJ	J#	J

کلید در مدل رابطه‌ای – کلید خارجی (ادامه)

- اگر N تعداد FKهای رابطه R باشد، داریم $N \geq 0$.
- معرفی کلید خارجی با عبارت FOREIGN KEY انجام می‌شود.
- **نقش کلید خارجی:** برای نمایش ارتباطهای صریح بین نوع موجودیت‌ها (و در نتیجه بین نمونه‌های آنها) به کار می‌رود. منظور از ارتباط صریح، ارتباطی است که در مدل ER با لوزی مشخص شده است.

مثال

S (S#, …) — CK

P (P#, …) — CK

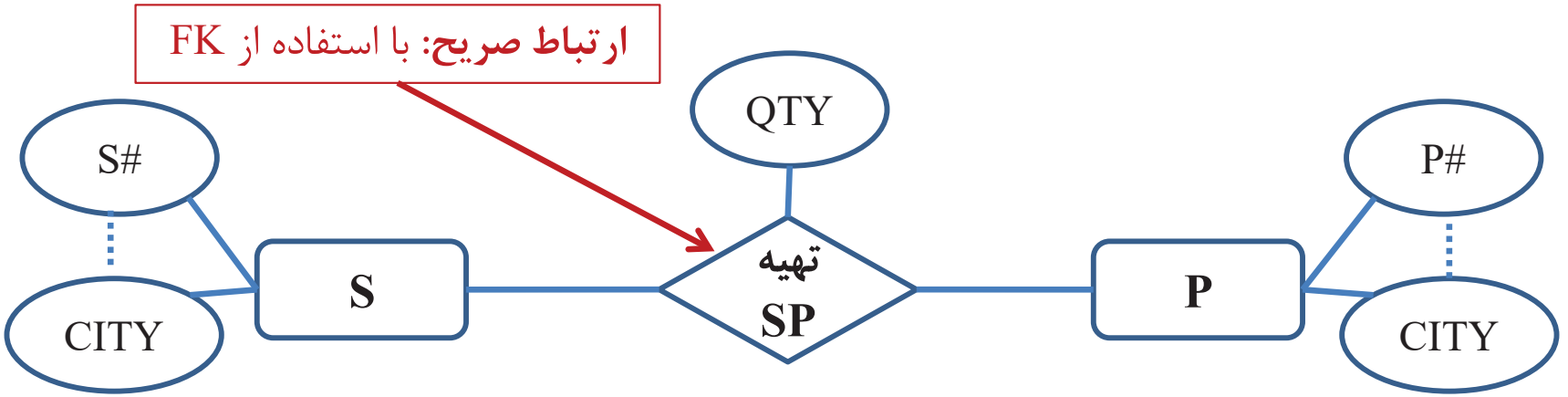
SP (S#, P#, …) — FK (S#), FK (P#), CK

SCOT (STID, COID, …) — FK (STID), FK (COID), CK

کلید در مدل رابطه‌ای – کلید خارجی (ادامه)

- آیا FK تنها امکان نمایش ارتباط است یا امکان دیگری هم وجود دارد؟
- FK تنها امکان نیست.
- وجود هر صفت مشترک [هم دامنه و در عمل، هم‌نام (نه لزوماً)]، در عنوان مثلاً دو رابطه، نمایشگر نوعی ارتباط است بین دو نوع موجودیت که با آن دو رابطه نمایش داده‌ایم.

مثال



جامعیت در مدل رابطه‌ای

جامعیت پایگاه داده‌ها (DB Integrity)

تعریف: صحت، سازگاری [، دقت و اعتبار] داده‌های ذخیره شده در پایگاه داده‌ها

جنبه‌های کیفی داده (Data Quality Features)

- مسئولیت کنترل جامعیت DB با RDBMS است.
- بر اساس اطلاعاتی که کاربر [تیم طراح – پیاده‌ساز] به سیستم می‌دهد.

← **قواعد یا محدودیت‌های جامعیتی (Integrity Rules/Constraints)**

- IRها [ICها] با استفاده از دستورات زبان پایگاهی به سیستم داده می‌شوند.

← **اعلانی:** قواعد به نحوی اعلان می‌شوند.

← **اجرایی:** قواعد در یک رویه به سیستم داده می‌شوند.

قواعد جامعیت در مدل رابطه‌ای

IRها [ICها] در مدل رابطه‌ای

۱- **قواعد [محدودیت‌های] عام:** ناوابسته به داده‌های محیط: فراقواعد (MetaRules)

۲- **قواعد [محدودیت‌های] خاص:** وابسته به داده‌های محیط: قواعد کاربری (User Defined)

یا قواعد فعالیت‌های محیط (Business Rules)

قواعد عام در مدل رابطه‌ای

- قاعده C1: جامعیت موجودیتی
- قاعده C2: جامعیت ارجاعی

قواعد عام در مدل رابطه‌ای – قاعده جامعیت موجودیتی C1

قاعده (محدودیت) C1 – قاعده جامعیت موجودیتی (Entity IR)

- ناظر است به PK.
- هیچ جزء تشکیل‌دهنده PK نباید هیچ‌مقدار (Null) داشته باشد.
- دلیل:
  - ✓ PK عامل تمییز تاپل‌ها است.
  - ✓ تاپل در مدل رابطه‌ای نمایشگر نمونه موجودیت است.
  - ✓ PK عامل تمییز نمونه موجودیت‌ها است.

  } عامل تمییز خود نمی‌تواند ناشناخته باشد.

- مکانیزم اِعمال C1: اعلان PK به سیستم ← کنترل می‌کند
  - ۱- محدودیت یکتایی مقدار (با UNIQUE
  - فقط این محدودیت کنترل می‌شود)
  - ۲- محدودیت هیچ‌مقدارناپذیری

قواعد عام در مدل رابطه‌ای – قاعده جامعیت ارجاعی C2

قاعده (محدودیت) C2 – قاعده جامعیت ارجاعی (Referential IR)

- ناظر است به FK.
- اگر $R_2.A_i$ در R_2، کلید خارجی باشد، مقدار A_i در هر تاپل از R_2 باید در R_1 مقدار قابل انطباق (Matchable Value) داشته باشد.
- به عبارت دیگر باید هر مقدار معلوم A_i در R_2، در R_1 نیز وجود داشته باشد. یعنی در عمل می‌تواند در R_2 مقدار آن Null باشد (البته اگر جزء تشکیل‌دهنده کلید R_2 نباشد).
- دلیل:
  - FK عامل ارجاع است؛ ارجاع به نمونه موجودیت (ارجاع مقداری و نه ارجاع از طریق اشاره‌گر).
  - در واقعیت نمی‌توان به نمونه موجودیت ناموجود ارجاع داد.

قواعد عام در مدل رابطه‌ای – قاعده جامعیت ارجاعی C2 (ادامه)

مثال

STT (**STID**, …)

777
888
444

STCOT (**STID**, **COID**, …)

777	CO1
…	…
444	CO4

INSERT INTO STCOT
VALUES (‘999’, ‘CO9’, …)

- چون برای 999 مقدار قابل انطباق در STT وجود ندارد، پس این درخواست رد می‌شود.

قواعد خاص در مدل رابطه‌ای

- **قواعد خاص در مدل رابطه‌ای:**
  - محدودیت دامنه‌ای (میدانی)
  - محدودیت صفتی
  - محدودیت رابطه‌ای
  - محدودیت پایگاهی

قواعد خاص در مدل رابطه‌ای (ادامه)

محدودیت دامنه‌ای (میدانی)

- این محدودیت ناظر است به دامنه، مشخص‌کننده نوع و طیف مقادیر دامنه
- در همان دستور CREATE DOMAIN اعلان می‌شود.

CREATE DOMAIN GRADE DEC(2, 2) **DEFAULT** '?...?' دستور ایجاد دامنه

CONSTRAINT GRADECONST نام محدودیت (اختیاری)

CHECK VALUE BETWEEN (0, 20)

DROP DOMAIN GRADE دستور حذف دامنه

قواعد خاص در مدل رابطه‌ای (ادامه)

محدودیت صفتی [استونی]

- این محدودیت ناشی می‌شود از محدودیت دامنه‌اش
- صفت می‌تواند محدودیت‌های دیگری هم داشته باشد، به شرطی که ناقض محدودیت دامنه‌ای‌اش نباشد.

مثال محدودیت‌های ناظر به صفت:

۱- صفت نمره باید بین ۰ تا ۲۰ باشد.

۲- صفت سن کاهش نمی‌یابد (محدودیت پردازشی).

محدودیت ۱، یک **محدودیت وضعیتی** است ولی محدودیت ۲، یک **محدودیت گذاری** است.

قواعد خاص در مدل رابطه‌ای (ادامه)

- محدودیت صفتی را چگونه می‌توان به سیستم اعلان کرد؟

۱- با تعریف دامنه‌اش اعلان می‌شود.

۲- در همان دستور CREATE TABLE با عبارت CHECK اعلان می‌شود.

جدول انتخاب درس

```
CREATE TABLE STCOT
        (STID …
         COID …
         TR …
         GR …)
        CHECK (0 <= GR <= 20)
```

۳- با ASSERTION اعلان می‌شود.

۴- با TRIGGER به سیستم داده می‌شود (اجرایی).

قواعد خاص در مدل رابطه‌ای (ادامه)

محدودیت رابطه‌ای

- ناظر است به تاپل‌های یک رابطه (درون رابطه‌ای Intra-relational).
- حیطه اِعمالش یک رابطه است و مقادیر مجاز یک متغیر رابطه‌ای را مشخص می‌کند.
- باید در هر عملی که بر روی رابطه انجام می‌شود (که منجر به تغییر در متغیر رابطه‌ای می‌گردد) کنترل شود.

مثال تعداد واحد درس‌های عملی قابل اخذ برای هر فرد در هر ترم، حداکثر ۲ واحد است.

مثال تهیه‌کنندگان ساکن شهر C2 نمی‌توانند مقدار وضعیت بیش از ۱۵ داشته باشند.

قواعد خاص در مدل رابطه‌ای (ادامه)

محدودیت پایگاهی

- ناظر است به تاپل‌های بیش از یک رابطه که به نحوی با هم ارتباط معنایی [منطقی] دارند.

مثال رابطه بین جداول STT و STCOT

یا رابطه بین جداول S و SP

مثال دانشجوی رشته کامپیوتر نمی‌تواند درس آمار و احتمال را از گروه آموزشی D13 (دانشکده ریاضی) انتخاب کند. رابطه‌های دخیل: STT، COT و STCOT

مثال تهیه‌کننده ساکن شهر C7 با وضعیت کمتر از ۱۵، نمی‌تواند قطعه آبی رنگ با وزن بیش از ۱۰ گرم به تعداد بیش از ۱۰۰ عدد تهیه کند.

- محدودیت‌های رابطه‌ای و پایگاهی چگونه اِعمال می‌شوند؟
  - با ASSERTION (اِعلانی)
  - با TRIGGER (اجرایی)

امکانات بیان محدودیت‌ها – اِظهار

اِظهار – ASSERTION

- امکانی است اِعلانی برای بیان محدودیت‌های رابطه‌ای و پایگاهی [و صفتی]

CREATE ASSERTION *name*
[BEFORE|AFTER *action*
ON *tablename* **]**
CHECK *condition(s)*

- در قسمت *condition(s)* می‌توان یک شرط ساده، یک عبارت بولی شامل چند شرط و نیز یک عبارت SELECT معتبر نوشت (همانطور که بعد از عبارت WHERE نوشته می‌شود).

- دستور حذف اِظهار

DROP ASSERTION ***name***

امکانات بیان محدودیت‌ها – اِظهار (ادامه)

مثال با این اظهار، محدودیت یکتایی مقادیر صفت کد ملی STNATID اعلان می‌شود.

```
CREATE ASSERTION  UNC-CHECK
        CHECK (UNIQUE(SELECT  STNATID  FROM  STT))
```

مثال با این اظهار این محدودیت که «جمع واحدهای انتخابی دانشجو در هر ترم-سال نباید بیش از ۲۰ واحد باشد»، اعلان می‌شود.

```
CREATE ASSERTION  TOTCRED-CHECK
        CHECK (NOT EXISTS (SELECT STID
                           FROM COT  JOIN  STCOT
                           GROUP BY  (STID, TR, YR)
                           HAVING SUM(CREDIT) > 20) )
```

امکانات بیان محدودیت‌ها – اِظهار (ادامه)

همه دانشجویان دانشکده مهندسی کامپیوتر (CE) باید درس مبانی برنامه‌سازی (با کد ۴۰۱۱۱) را اخذ کرده باشند.

```
CREATE ASSERTION   ELEM-CHECK
      CHECK (NOT EXISTS
                  ( SELECT  *  FROM  STT
                        WHERE  DEPT='CE'  AND
                              NOT EXISTS
                              ( SELECT * FROM  STCOT
                                WHERE  STCOT.STID = STT.STID
                                      AND  STCOT.COID='40111'  ) )
```

رهانا [راه‌انداز] – TRIGGER

- امکانی است اجرایی برای اِعمال محدودیت‌های [صفتی،] رابطه‌ای و پایگاهی قبل یا بعد از بروز یک رویداد و یا به جای یک رویداد (معمولا تغییر دهنده داده‌ها).

```
CREATE TRIGGER name
        {BEFORE | AFTER | INSTEAD OF}
        {INSERT | DELETE | UPDATE OF columnlist
        ON tablename
        [REFERENCING { OLD ROW | NEW ROW | OLD TABLE | NEW TABLE} AS name ]
        [FOR EACH {ROW | STATEMENT}]
        {(WHEN condition(s)
                SQL 2003 Procedure
        )}
```

- مفهوم نظری TRIGGER: مفهوم قاعده فعال [مفهوم محوری است در ADBMSها]

ساختار (قاعده ECA): if **E**vent on **C**ondition, then **A**ction

- Insert
- Delete
- Update

امکانات بیان محدودیت‌ها – رهانا (ادامه)

- با FOR EACH ROW بعد از بروز رویداد در هر سطر عبارت رهانا اجرا شود.
- با FOR EACH STATEMENT فقط یک بار پس از بروز رویداد (با هر تعداد سطر متاثر از آن)، عبارت رهانا اجرا شود.

این رهانا این محدودیت را که «حقوق کارمند هیچگاه کاهش نمی‌یابد» اِعمال می‌کند.

```
CREATE TRIGGER EMP-PAY-TRIG
        BEFORE UPDATE OF EMPSAL
        ON EMPL
        REFERENCING OLD AS OEMPL,  NEW AS  NEMPL
        FOR EACH ROW
        (WHEN OEMPL.EMPSAL > NEMPL.EMPSAL
                SIGNAL.SQL State ‘7005’ (‘salary cannot be decreased’)
        )
```

امکانات بیان محدودیت‌ها – رهانا (ادامه)

این رهانا باعث حفظ سازگاری در جدول PROF می‌شود تا همواره صفت SALAUG حاوی آخرین میزان افزایش حقوق استاد باشد.

```
CREATE TRIGGER EMP-PAY-TRIG
        AFTER UPDATE OF  PSALARY
        ON  PROF
        REFERENCING OLD AS OPROF,  NEW AS  NPROF
        FOR EACH ROW
        (UPADATE   PROF
                SET   SALAUG=NPROF.PSALARY – OPROF.PSALARY
                WHERE   PROF.PID=OPROF.PID
        )
```

اگر بیش از یک عبارت باشد، آنها را داخل BEGIN و END قرار می‌دهیم.

امکانات بیان محدودیت‌ها – رهانا (ادامه)

مثال از کاربردهای رهانا، استفاده از آن در انجام عملیات ذخیره‌سازی از دید خارجی است (به خصوص در سمپادهایی که از عملیات در دید خارجی پشتیبانی نمی‌کنند).

STT1 (STID, NAME, MAJOR, LEVEL)
STT2 (STID, DEPT, BDATE, NATID)

```
CREATE VIEW CE-STT
        AS SELECT STID, NAME, MAJOR
        FROM STT1 JOIN STT2
        WHERE DEPT='CE' AND LEVEL ='BS'

CREATE TRIGGER INS-VIEW-TRIG
        INSTEAD OF INSERT ON CE-STT
        REFERENCING NEW AS NST
        FOR EACH ROW
        BEGIN
            INSERT INTO STT1 VALUES ( NST.STID, NST.NAME, NST.MAJOR, 'BS')
            INSERT INTO STT1 VALUES ( NST.STID, 'CE', NULL, NULL)
        END
```

امکانات بیان محدودیت‌ها – رهانا (ادامه)

مثال این رهانا باعث اعمال قاعده C2 در عمل حذف می‌شود.

```
CREATE TRIGGER  DEL-TRIG
        BEFORE   DELETE
        ON  COT
        REFERENCING OLD AS OCOT
        FOR EACH ROW
        (DELETE   FROM   STCOT
                WHERE   STCOT.COID=OCOT.COID  )
```

- **مطالعه مثالهای بیشتر از اظهار و رهانا در یادداشت‌های تکمیلی**

به نام آنکه جان را فکرت آموخت

بخش هفتم:
عملیات در پایگاه داده رابطه‌ای

مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

- عملگرها
  - عملگرهای متعارف
    - عملگرهای دو عملوندی $R_1\ op\ R_2$ $op \in \{\cup,\cap,,-,\times\}$
      - اجتماع - UNION
      - اشتراک - INTERSECT
      - تفاضل - MINUS
      - ضرب کارتزین - TIMES
  - عملگرهای خاص
    - گزینش یا تحدید - RESTRICT
    - پرتو یا تصویر - PROJECT
    - پیوند یا الصاق - JOIN

عملگرهای متعارف جبر رابطه‌ای

- **خاصیت بسته بودن:** حاصل ارزیابی هر عبارت جبر رابطه‌ای معتبر، باز هم یک رابطه است (که تاپل تکراری ندارد).
- برای **سه عملگر** $\cup$ ، $\cap$ **و** $-$ ، باید عملوندها نوع-سازگار (Type Compatible) باشند:
  - **پیش شرط:** $H_{R_1} = H_{R_2}$
  - $R_3 = R_1 \; op \; R_2 \longrightarrow H_{R_3} = H_{R_1} = H_{R_2} \qquad op \in \{\cup, \cap, , -\}$
    - بدنه نتیجه، حاصل انجام هر یک از اَعمال اجتماع، اشتراک و یا تفاضل دو مجموعه بدنه است.
- **در عملگر ضرب کارتزین (TIMES):**
  - **شرط:** در عنوان دو رابطه نباید صفت هم‌نام وجود داشته باشد. $H_{R_2} \cap H_{R_1} = \varnothing$
  - عنوان رابطه نتیجه برابر است با $H_{R_2} \cup H_{R_1}$ و بدنه نتیجه برابر ضرب کارتزین دو مجموعه بدنه است.
  - TIMES در SQL چگونه شبیه‌سازی می‌شود؟

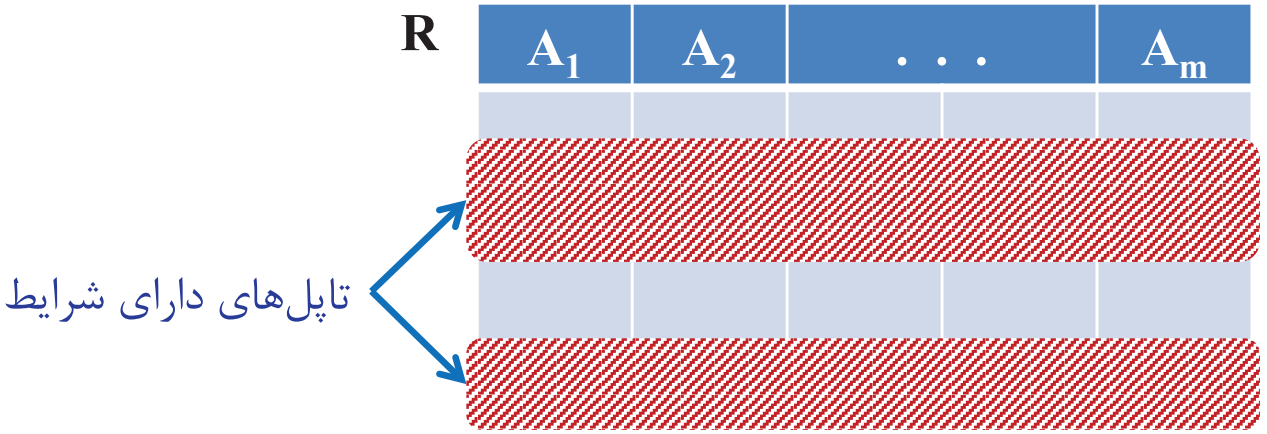
عملگر گزینش

عملگر گزینش یا تحدید – RESTRICT

- نماد ریاضی: σ_c ← شرط یا شرایط گزینش

یک عبارت بولی تشکیل شده از شرطهای ساده به صورت (A_i *theta* A_j) یا (A_i *theta* literal) که در آن *theta* یکی از عملگرهای $=$، $\neq$، $<$، $>$، $\leq$ و $\geq$ است و literal یک مقدار ثابت است.

- شکل کلی: $\sigma_c(R)$ **یا** R **WHERE** c **یا** **RESTRICT** R **WHERE** c
- تک عملوندی: Monadic
- **عملکرد** (در نمایش جدولی رابطه): زیرمجموعه‌ای افقی می‌دهد. ← عملگر تاپل(ها)یاب



عملگر گزینش (ادامه)

مشخصات کامل دانشجویان رشته فیزیک دوره کارشناسی را بدهید.

$$\sigma_{STJ='phys' \wedge STL='bs'}(STT)$$

```
SELECT STT.*
        FROM  STT
        WHERE  STJ='phys'  AND  STL='bs'
```

وقتی در شرط C (یا کلاز WHERE) بخشی از کلید را با شرط تساوی داده باشیم.

اگر $R' = \sigma_c(R)$ باشد آنگاه $CK_{R'} \subseteq CK_R$.

عملگر گزینش (ادامه)

- عملگر گزینش جابجایی‌پذیر است، یعنی:

$$\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R)) = \sigma_{c1 \wedge c2}(R)$$

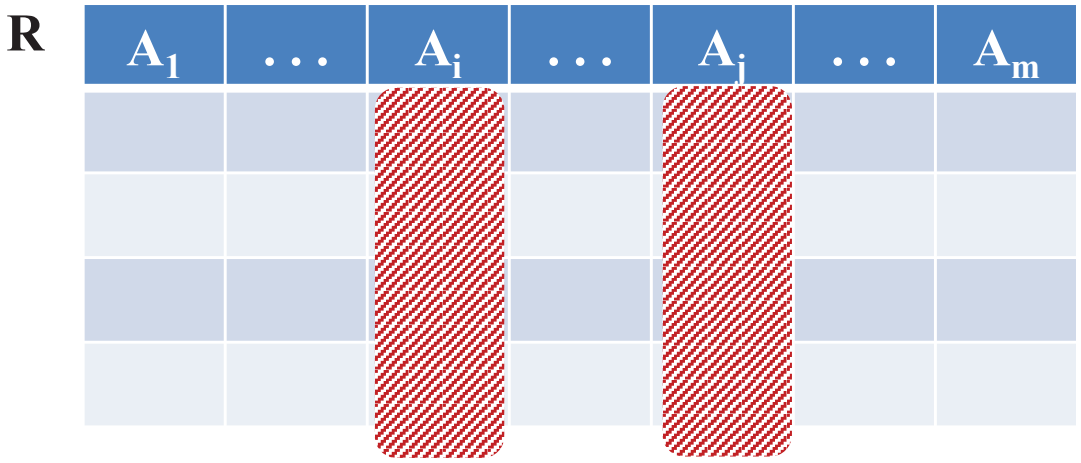
- **عبارتهای جبری معادل:**

- R **WHERE** (C_1 **AND** C_2) ≡ (R **WHERE** C_1) **INTERSECT** (R **WHERE** C_2)
- R **WHERE** (C_1 **OR** C_2) ≡ (R **WHERE** C_1) **UNION** (R **WHERE** C_2)
- R **WHERE NOT** C ≡ R **MINUS** (R **WHERE** C)

عملگر پرتو

عملگر پرتو – PROJECT

- نماد ریاضی: **Π**
- شکل کلی: $\Pi_{\langle L \rangle}(R)$ یا $(R)[L]$ یا **PROJECT R OVER (L)**
  - $\langle L \rangle$ ← لیست صفات پرتو
- تک عملوندی: Monodic
- **عملکرد** (در نمایش جدولی رابطه): زیرمجموعه عمودی می‌دهد. ← عملگر ستون(ها)یاب



عملگر پرتو (ادامه)

- عملگر پرتو تکراری‌ها را حذف می‌کند. ← چون جواب رابطه است، پس یک مجموعه است و عضو تکراری ندارد.

مثال شماره و رشته تمام دانشجویان را بدهید.

$$\Pi_{\langle STID,STJ\rangle}(STT)$$

SELECT STID, STJ **FROM** STT

مثال شماره دانشجویانی که درسی انتخاب نکرده‌اند.

$$R := \Pi_{\langle STID\rangle}(STT) - \Pi_{\langle STID\rangle}(STCOT)$$

مثال شماره و مقطع تحصیلی دانشجویان رشته IT را بدهید.

$$\Pi_{\langle STID,STL\rangle}\left(\sigma_{STJ='IT'}(STT)\right)$$

عملگر پرتو (ادامه)

- اگر $R' = \Pi_{\langle L \rangle}(R)$ باشد آنگاه:
  - اگر $CK_R \subseteq L$ آنگاه $CK_{R'} = CK_R$،
  - اگر نه در حالت کلی $CK_{R'} = L$.

اگر $R' = R_1 \; op \; R_2$ و $op \in \{\cup, \cap, , -, \times\}$، آنگاه $CK_{R'} = ?$.

- SELECT در SQL استاندارد، در حالت کلی ترکیبی از دو عملگر RESTRICT و PROJECT است.

عملگر تغییر نام

عملگر تغییر نام – RENAME

- نماد ریاضی: ρ
- شکل کلی: $\rho_R(E)$

  نام رابطه حاصل از عبارت جبر رابطه‌ای E
- این عملگر برای نامیدن رابطه حاصل از یک عبارت جبر رابطه‌ای به کار می‌رود.
- **عملکرد:** $\rho_R(E)$ رابطه حاصل از عبارت جبر رابطه‌ای E را با نام R برمی‌گرداند.
- از عملگر RENAME برای دگرنامی صفت هم می‌توان استفاده کرد (مشابه آنچه در مثال اسلاید قبل آمد). مثلاً با دستور R **RENAME** A_i **AS** B_j، به صفت A_i از R، نام دیگر B_j داده می‌شود.

عملگر پیوند

عملگر پیوند JOIN (مدل ریاضی عمومی)

- **نام عمومی:** Theta Join
- **نماد ریاضی:** $\bowtie_{Cond(s)}$ ← شرط پیوند
- **فرض:** دو رابطه R_1 و R_2 نام صفت مشترک ندارند.
  - $R_1 (A_1, A_2, \ldots, A_n)$
  - $R_2 (B_1, B_2, \ldots, B_m)$
- **شکل کلی:** $\mathbf{R_1 \bowtie_C R_2}$ یا $\mathbf{R_1\ \theta\text{-}JOIN_C\ R_2}$ یا فقط $\mathbf{R_1\ JOIN_C\ R_2}$
- **Theta**
  - EQUI-JOIN $=$
  - NOT EQUI-JOIN $\neq$
  - LESS THAN-JOIN $<$
  - LESS EQUI-JOIN $\leq$
  - GREATER THAN-JOIN $>$
  - GREATER EQUI-JOIN $\geq$

عملگر پیوند (ادامه)

- **شرط پیوند (c):** $R_1.A_i$ **theta** $R_2.B_j$

صفات پیوند

که باید **هم‌دامنه** و **ناهم‌نام** باشند.

چون نتیجه JOIN رابطه است و در headingاش صفت تکراری نباید وجود داشته باشد.

- **نکته:** اگر صفات پیوند هم‌نام باشند، حداقل یکی را باید دگرنامی کرد (به دلیل وجود این راه حل، حساسیتی در عدم وجود صفت مشترک نداریم).
- در حالت کلی شرط پیوند می‌تواند به صورت زیر باشد که در آن c_1، ...، c_n قالب بالا (قالب شرط پیوند) را دارند.

<c_1> AND <c_2> AND ... AND <c_n>

مثال

<R1.A1 = R2.B1> **AND** <R1.A2 = R2.B2>

عملگر پیوند (ادامه)

مثال: مشخصات کامل جفت تهیه‌کننده-قطعه از یک شهر را بدهید.

$$R_1 := S \bowtie_{S.CITY=P.PCITY} (P \text{ RENAME } CITY \text{ AS } PCITY)$$

S (S#, SNAME, STATUS, CITY)

S1		C1
S2		C2
S3		C3
S4		C4
S5		C5
S6		C6

P (P#, . . . , W, CITY)

P1	5	C1
P2	6	C2
P3	4	C1
P4	7	C4
P5	10	C5

R_1 (S#, . . . , CITY, P#, . . . , W, PCITY)

S1	C1	P1	5	C1
S1	C1	P3	4	C1
S2	C2	P2	6	C2
~~S3~~ تاپل پیوندشدنی ندارد.				
S4	C4	P4	7	C4
S5	C5	P5	10	C5
~~S6~~ تاپل پیوندشدنی ندارد.				

عملگر پیوند (ادامه)

❑ **عملکرد:**

$$R_3 = R_1 \bowtie_C R_2$$

$$H_{R_3} = H_{R_1} \cup H_{R_2}$$

- در بدنه R_3 تاپل‌های پیوندشدنی از دو رابطه قرار دارند.

❑ **خصوصیات:**

- $R_1 \bowtie_C R_2 = R_2 \bowtie_C R_1$ چون صفات در heading رابطه نظم مکانی ندارند.
- $R_1 \bowtie_C R_2 = \sigma_C(R_1 \times R_2)$ حاصل Theta-Join در حالت عمومی، زیرمجموعه‌ای افقی از ضرب کارتزین است که در آن تاپل‌هایی از حاصلضرب که حائز شرط پیوند هستند حضور دارند.

وقتی در شرط پیوند، تساوی بخشی از کلید هر دو رابطه را داده باشیم.

اگر $R' = R_1 \bowtie_C R_2$ باشد، آنگاه $CK_{R'} \subseteq CK_{R_1} \cup CK_{R_2}$

گونه‌های خاص عملگر پیوند – پیوند طبیعی

پیوند طبیعی (Natural Join)

- گونه‌ای از پیوند است که دو ویژگی دارد:

۱- Theta: =

۲- صفات پیوند یک بار در جواب می‌آیند. (صفت یا صفات پیوند باید هم‌نام هم باشند.)

$$R_2 := S \bowtie_{S.CITY=P.CITY} P$$

R_2 (S#, . . . , CITY, P#, . . . , W)

S1	C1	P1	5
S1	C1	P3	4
S2	C2	P2	6
S4	C4	P4	7
S5	C5	P5	10

گونه‌های خاص عملگر پیوند – پیوند طبیعی (ادامه)

- اگر صفت مشترک [هم‌نام و هم‌دامنه] یک صفت باشد، نیازی به قیدکردن نیست.

  اما اگر بیش از یک صفت باشد، باید صفت یا صفات پیوند را قید کنیم.

  اگر قید نکنیم، پیوند روی تساوی مقادیر تمام صفات مشترک انجام می‌شود.

$$R_1: (A, B, C)$$

$$R_2: (A, F, C)$$

$$R' = R_1 \bowtie R_2$$

$$R': (A, B, C, F)$$

- اگر $H_{R_1} \cap H_{R_2} = \emptyset$، آنگاه $R_1 \bowtie R_2 = R_1 \times R_2$.
- اگر $H_{R_1} = H_{R_2}$، آنگاه $R_1 \bowtie R_2 = R_1 \cap R_2$.

گونه‌های خاص عملگر پیوند – نیم‌پیوند

نیم‌پیوند (Semijoin)

- در شکل عمومی با هر Theta نوشته می‌شود.
- **نماد:** $\ltimes_C$ (در چپ تعریف شده)
- **مدل ریاضی:** $R_3 := R_1 \ltimes_C R_2 = \Pi_{\langle H_{R_1} \rangle}(R_1 \bowtie_C R_2)$
- **عملکرد:**
  - $H_{R_3} = H_{R_1}$
  - در بدنه R_3: تاپل‌های پیوند شدنی از رابطه چپ

گونه‌های خاص عملگر پیوند – نیم‌پیوند (ادامه)

$$R_3 := S \ltimes_{S.CITY=P.PCITY} (P \text{ RENAME } CITY \text{ AS } PCITY)$$

R_3 (S#, . . . , CITY)

S1	C1
S2	C2
S4	C4
S5	C5

کاربرد این عملگر چیست؟

- **تمرین:** عملگر نیم‌پیوند در SQL شبیه‌سازی شود.

برون‌پیوند (Outer Join)

- Theta هر چیزی می‌تواند باشد.
- سه گونه دارد:

۱- Left O. J. $\quad ⟕_C$

۲- Right O. J. $\quad ⟖_C$

۳- Full O. J. $\quad ⟗_C$

- عملکرد $R_4 := R_1 ⟗_C R_2$:
  - $H_{R_4} = H_{R_1} \cup H_{R_2}$
  - در بدنه R_4: تاپل‌های پیوند شدنی از دو رابطه و

تاپل‌های‌های پیوندناشدنی از رابطه چپ گسترش‌یافته با هیچ‌مقدار (Null Value)

گونه‌های خاص عملگر پیوند – برون‌پیوند (ادامه)

$R_4 := S \rtimes P$

R_4 (S#, . . . , CITY, P#, . . . , W)

S1	C1	P1	5
S1	C1	P3	4
S2	C2	P2	6
S4	C4	P4	7
S5	C5	P5	10
S3	C3	?	?
S6	C6	?	?

کلید R_4 (CK_{R4}) چیست؟ بی‌تردید کلید اصلی ندارد.

- مشکل Outer Join:

۱- از نظر ریاضی رابطه نیست، چون کلید اصلی ندارد.

۲- مصرف حافظه زیاد

این عملگرها در عمل چه کاربردی دارند؟

آیا عملگرهای Outer Join خاصیت جابجایی دارند؟

عملگر تقسیم (Divide)

مفروضند رابطه‌های:

$$R_1\,(\underbrace{A_1, A_2, \ldots, A_n}_{X}, \underbrace{B_1, B_2, \ldots, B_m}_{Y})$$

$$R_2\,(B_1, B_2, \ldots, B_m)$$

شرط عمل:

- $R_3(X) := R_1(X, Y) \div R_2(Y) \longrightarrow H_{R_2} \subseteq H_{R_1}$

عملکرد:

۱- $H_{R_3} = X = H_{R_1} - H_{R_2}$

۲- در بدنه R_3: بخش X از تاپلهایی از R_1 که حاوی تمام مقادیر Y از R_2 باشند.

عملگر تقسیم (ادامه)

$R_1(S\#, P\#) \div R_2(P\#) = R_3(S\#)$

S1	P1	P1	S1
S1	P2	P2	
S1	P3	P3	
S2	P1		
S2	P2		
S3	P1		

$R_1(S\#, P\#) \div R_4(P\#) = R_5(S\#)$

S1	P1	P1	S1
S1	P2	P2	S2
S1	P3		
S2	P1		
S2	P2		
S3	P1		

Outer Divide چیست؟

به نام آنکه جان را فکرت آموخت

بخش چهارم:
مقدمات پیاده‌سازی و SQL

مرتضی امینی

نیمسال دوم ۹۴-۹۵

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

- برای پیاده‌سازی طراحی منطقی انجام شده در یک سیستم مدیریت پایگاه داده‌ها نیاز به یک زبان پایگاهی داریم.
- زبان SQL زبان استاندارد انجام عملیات پایگاهی در پایگاه داده‌های رابطه‌ای (از دیدگاه کاربردی: جدولی) است.
- دستورهای Structured Query Language (SQL):
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Data Control Language (DCL)
- چند دستور از DDL:
  - CREATE TABLE ایجاد جدول
  - DROP TABLE حذف جدول
  - ALTER TABLE تغییر جدول
- **نکته:** در دستورات SQL در دو طرف مقادیر متنی یا رشته‌ای از single quote استفاده می‌شود (بسیاری از سیستم‌های پایگاه داده double quote را هم می‌پذیرند) ولی در اطراف مقادیر عددی چیزی قرار نمی‌گیرد.

تعریف و حذف پایگاه داده و شِما

- **دستور تعریف پایگاه داده**

CREATE DATABASE *DatabaseName*

- **دستور حذف پایگاه داده**

DROP DATABASE *DatabaseName*

- در اغلب سمپادها می‌توان در یک پایگاه داده چند شما تعریف کرد.
- **دستور تعریف و حذف شِما**

CREATE SCHEMA *SchemaName*

DROP SCHEMA *SchemaName*

- **شمای پایگاه داده‌ها** عبارت است از تعریف (توصیف) ساختهای منطقی طراحی شده و نوعی برنامه است شامل تعدادی دستور برای تعریف و کنترل داده‌ها.
- در واقع شِما شامل همه جداول، نوعها، دامنه‌ها، دیدها و محدودیتهای مرتبط با یک برنامه کاربردی است.

تعریف جدول

تعریف جدول‌ها: شِمای پایگاه جدولی

دستور تعریف جدول CREATE TABLE

CREATE TABLE *TableName*
{ (*columnName* *dataType* [**NOT NULL** | **UNIQUE**]
[**DEFAUTL** *defaultOption*][***CHECK*** (*searchCondition*)] **[, ...]**)}
[**PRIMARY KEY** (*listOfColumns*),]
{[**UNIQUE** (*listOfColumns*),][, …]}
{[**FOREIGN KEY** (*listOfForeignKeyColumns*)
REFERENCES *ParentTableName* [(*listOfCandidateKeyColumns*)],
[**ON UPDATE** *referentialAction*]
[**ON DELETE** *referentialAction*]][, …]}
{[**CHECK** (*searchCondition*)][, …]}

- می‌توان جدول را به صورت موقت نیز (با استفاده از CREATE TEMPORARY TABLE) ایجاد کرد. جدول موقت حاوی داده‌های ناپایا است و پس از اینکه برنامه کاربر (SQL Session) اجرایش تمام بشود، این جدول توسط سیستم حذف می‌شود.

انواع داده‌ای

- انواع داده‌های قابل استفاده در تعریف ستون‌ها عبارتند از:
  - کاراکتری: CHAR(n), VARCHAR(n)
  - بیتی: BIT [VARYING] (n)
  - عددی: NUMERIC(p, q), REAL, INTEGER, SMALLINT, FLOAT(p), DOUBLE PRECISION
  - زمانی: DATE, TIME, TIMESTAMP, INTERVAL
  - ....
- در برخی DBMSها، نوع داده‌های خاصی پشتیبانی می‌شود که امکان ذخیره، بازیابی و پردازش داده‌های از آن نوع را برای کاربر تسهیل می‌نماید. به طور مثال نوع داده جغرافیایی در PostgreSQL.

محدودیتهای صفتی (ستونی)

- **Default**: تعیین مقدار پیش‌فرض یک ستون
- **Not Null**: ستون ناهیچ‌مقدار
- **Unique**: یکتایی مقادیر ستون(ها)
- **Primary Key**: کلید اصلی (می‌توان تعدادی از ستونها را با یکدیگر به عنوان کلید اصلی تعریف کرد)
- **Foreign Key .... References ....**: کلید خارجی (می‌توان تعدادی از ستونها را با یکدیگر به عنوان کلید خارجی تعریف کرد)
- **Check**: تعیین محدودیت مقداری برای مقادیر ستون

مثالی از تعریف جدول

مثال شِمای پایگاه داده جدولی:

```
CREATE TABLE STT
( STID          CHAR(8) NOT NULL,
STNAME          CHAR(25),
STLEV           CHAR(12),
STMJR           CHAR(20),
STDEID          CHAR(4)
)
PRIMARY KEY (STID);
CHECK STMJR IN { 'bs', 'ms', 'doc', '???' }
```

```
CREATE TABLE COT
( COID          CHAR(6) NOT NULL,
COTITLE         CHAR(16),
CREDIT          SMALLINT,
COTYPE          CHAR(1),
CODEID          CHAR(4),
)
PRIMARY KEY (COID);
```

محدودیت صفتی (ستونی) [کلاز کنترلی]

مثالی از تعریف جدول (ادامه)


```
CREATE TABLE SCT
( STID          CHAR(8)  NOT  NULL ,
COID            CHAR(6)  NOT  NULL ,
TR              CHAR(1) ,
YR              CHAR(5) ,
GRADE           DECIMAL(2 , 2)
)
PRIMARY  KEY ( STID, COID )
CHECK  0 ≤ GRADE ≤ 20
FOREIGN  KEY  (STID)  REFERENCES  STT  (STID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
FOREIGN  KEY  (COID)  REFERENCES  COT (COID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
```

محدودیت صفتی (ستونی) [کلاز کنترلی]

حذف جدول

دستور حذف جدول DROP TABLE

DROP TABLE ***tablename*** **[CASCADE| RESTRICT]**

- **CASCADE** باعث می‌شود که همه اشیاء وابسته به جدول (مانند دیدهای تعریف شده بر روی آن یا محدودیتهایی مانند کلید خارجی وابسته به آن) نیز به صورت خودکار حذف شود.
- **RESTRICT** در صورت وجود دیگر اشیاء وابسته به جدول، از حذف آن جلوگیری می‌کند. پیش‌فرض این دستور، RESTRICT است.

DROP TABLE SCT

تغییر جدول

دستور تغییر جدول ALTER TABLE

اضافه کردن ستون، تغییر تعریف ستون، حذف ستون و ...

ALTER TABLE *tableName*

[ADD [COLUMN] *columnName dataType* **[NOT NULL] [UNIQUE]**

[DEFAULT *defaultOption*] **[CHECK (***searchCondition***)]]**

[DROP [COLUMN] *columnName* **[RESTRICT | CASCADE]]**

[ADD [CONSTRAINT [*constraintName***]]** *tableConstraintDefinition*]

[DROP [CONSTRAINT *constraintName* **[RESTRICT | CASCADE]]**

[ALTER [COLUMN] SET DEFAULT *defaultOption*]

[ALTER [COLUMN] DROP DEFAULT]

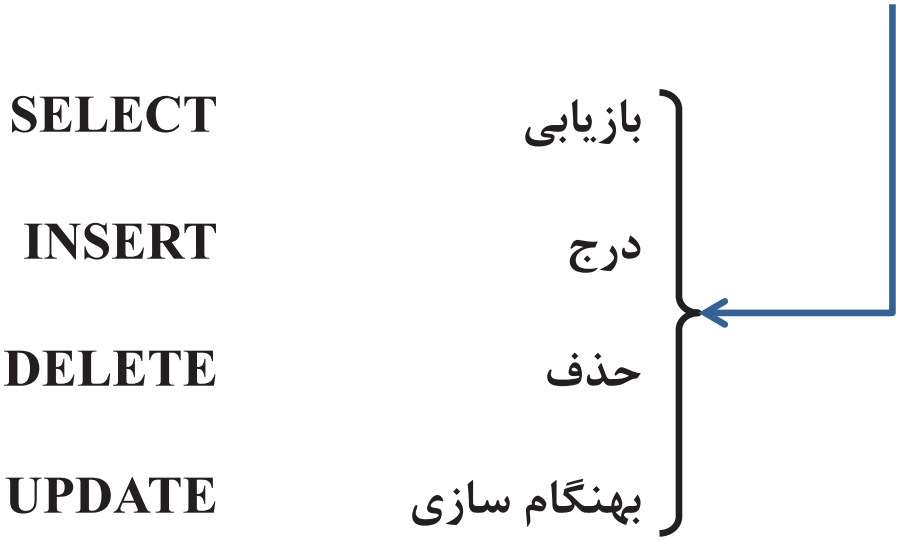
. . .

اضافه کردن ستون «وضعیت» به جدول اطلاعات دانشجو

ALTER TABLE STT

ADD COLUMN STATE CHAR(10)

- عملیات در TDB : **دستور های DML**



دستور بازیابی SELECT

SELECT [ALL | DISTINCT] *item(s) list*
FROM *table(s)* *expression*
[WHERE *condition(s)*]
[ORDER BY *Col(s)*]
[GROUP BY *Col(s)*]
[HAVING *condition(s)*]

- خروجی دستور SELECT یک جدول است.
- از DISTINCT برای حذف سطرهای تکراری در جدول نتیجه استفاده می‌شود.
- در شرط WHERE می‌توان از =، <>، <، >، >=، =>، BETWEEN، LIKE و IN استفاده کرد.


```
SELECT  STT.STID    AS   SN ,
        STT.STNAME   AS   SName
        FROM     STT
        WHERE   STT.STMJR=‘phys’
                 AND
                 STT.STLEV=‘bs’
```


```
SELECT   STT1. STID        AS   SN ,
         STT1. STNAME   AS   SName
        FROM     STT   AS   STT1
        WHERE    STT1. STMJR=‘phys’
                 AND
                  STT1. STLEV=‘bs’
```

بازیابی داده‌ها (ادامه)

مثال

یک کپی از جدول با نام جدید، نام‌گذاری جدول جواب:

```
(SELECT  S.*
        FROM  S )  AS  MyS
ORDER  BY  SNAME   یا   2
```

شماره ستون

- مرتب شده:
- پیش فرض صعودی: (Ascending)
- نزولی (Descending): باید قید شود.

مثال

قابلیت‌های پیشرفته (Advanced features):

```
SELECT  S#, CITY
        FROM    S
        WHERE   SNAME   { LIKE / NOT LIKE }
```

- LIKE '%N' → با N تمام شود
- LIKE 'M%' → با M شروع شود
- LIKE '__A__' → دقیقا ۵ کاراکتر، کاراکتر سوم A

BETWEEN

مثال

```
SELECT P#
        FROM P
        WHERE    WEIGHT BETWEEN (5,15)
                          یا
        WHERE   WEIGHT >=5   AND WEIGHT <=15
```

- شماره قطعاتی را بدهید که وزن آنها بین ۵ و ۱۵ است.

مثال NULL

```
SELECT  S#, CITY
        FROM    S
        WHERE   STATUS { IS NULL
                       { IS NOT NULL
```

گنجای؟ بررسی برخورد یک package با NULL؟

عملگرهای جبر مجموعه‌ها

tablename1 **op** *tablename2* **[CORRESPONDING [BY {*column*, [, *column* …]}]]**

$$op \in \{ \text{UNION [ALL]},\ \text{INTERSECT [ALL]},\ \text{EXCEPT [ALL]} \}$$

- اگر از گزینه CORRESPONDING BY استفاده شود، عمل درخواست شده روی ستون‌های تصریح شده انجام می‌شود.
- اگر CORRESPONDING بدون BY استفاده شود، عمل درخواست شده روی ستون‌های مشترک انجام می‌شود.
- اگر از این گزینه استفاده نشود، عمل روی تمام ستون‌های دو جدول انجام می‌شود.
  - **شرط استفاده:** برابری Heading: هم‌نامی و هم نوعی ستون (های) دو جدول
- **توجه:** تکراری‌ها در نتیجه اجرای عملگرهای جبر مجموعه‌ها حذف می‌شوند مگر آنکه از ALL استفاده شود.

عملگرهای جبر مجموعه‌ها (ادامه)

مثال

شماره تهیه کنندگانی را بدهید که حداقل یک قطعه تولید می‌کنند.

```
SELECT  S.S#,
        FROM    S
        INTERSECT
SELECT  SP.S#,
        FROM    SP
```

مثال

تست سازگاری پایگاه داده‌ها: هر فردی که قطعه ای تولید کرده باید یکی از افراد ثبت شده در جدول تولیدکنندگان باشد.

```
SELECT  SP.S#,
        FROM    SP
        EXCEPT
SELECT  S.S#,
        FROM    S
```

مدل دیگر

SP *EXCEPT* S **Using S#** یا **Corresponding by S#**

عملگرهای جبر مجموعه‌ها (ادامه)

مثال: شماره تهیه کنندگانی را بدهید که هیچ قطعه‌ای تولید نمی‌کنند.

```
SELECT  S.S#,
        FROM    S
        EXCEPT
SELECT  SP.S#,
        FROM    SP
```

- تمرین: این مثال‌ها به طرز دیگر هم نوشته شود.

توابع جمعی (گروهی)

Aggregation Functions

- AVG ← میانگین
- MIN ← مینیمم
- MAX ← ماکزیمم
- SUM ← جمع
- COUNT(*) / COUNT ← تعداد عبارات ناهیچمقدار / تعداد کل سطرها

مثال: بیشینه وضعیت تهیه کنندگان در شهرهای c1 یا c2

```
SELECT  MAX ( STATUS )   AS   SMAX
        FROM     S
        WHERE  CITY='c1'
               OR
               CITY='c2'
```

توابع جمعی (ادامه)

تعداد انواع قطعات تولیدی توسط تولیدکنندگان

```
SELECT  COUNT (DISTINCT P#)   AS   N1
        FROM   SP
```

تعداد انواع قطعات قابل تولید

```
SELECT  COUNT (*)   AS   N2
        FROM   P
```

تعداد کل قطعات تولیدی توسط s2

```
SELECT  SUM (QTY)   AS   N3
        FROM   SP
        WHERE   S# = 's2'
```

NULL و توابع جمعی؟ (در سه پکیج بررسی شود)

GROUP BY

- سطرهای جدول داده شده در کلاز FROM را گروه بندی می‌کند، به نحوی که مقدار ستون(های) گروه‌بندی در گروه یکسان است.

مثال: تعداد کل قطعات تولیدی توسط هر تولیدکننده

```
SELECT  S#  AS  SN , SUM (QTY)  AS  SQ
        FROM    SP
        GROUP BY    S#
```

SP گروه‌بندی شده

S#	P#	QTY
s1	p1	…
s1	p2	…
s1	p4	…
s2	p2	…
s2	p3	…
s3	p5	…
…	…	…

جدول جواب

SN	SQ
s1	280
s2	100
s3	203
…	…

گروه‌بندی (ادامه)

در کلاز SELECT نمی توان نام ستونی را آورد که در کلاز GROUP BY نباشد، غیر از ستون‌هایی که با توابع جمعی به دست آمده‌اند.

HAVING

- امکانی است برای دادن شرط یا شرایط ناظر به گروه سطرها

شماره تهیه‌کنندگانی را بدهید که بیش از ۱۰۰ قطعه تولید کرده‌اند.

```
SELECT  S#
        FROM SP
        GROUP BY S#
        HAVING SUM(QTY) > 100
```

گروه‌بندی (ادامه)

- تمرین : شماره دانشجویانی را بدهید که در ترم دوم سال ۸۷-۸۸ بیش از ۲۰ واحد گرفته باشند.
- تمرین : شماره دانشجویانی را بدهید که در ترم دوم سال ۸۷-۸۸ بیش از ۷ درس گرفته باشند.

GROUP BY و HAVING در SQL افزونه‌اند، اما نوشتن QUERY بدون آنها پیچیده است.

HAVING بدون GROUP BY؟

به چند روش می‌توان یک کپی از جدول ساخت؟

بازیابی از بیش از یک جدول

نام تهیه کنندگان قطعه ’p2‘ را بدهید:

در جدول S

در جدول SP

روش اول

```
SELECT  SNAME
        FROM    S, SP
        WHERE   SP.S# = S.S#   AND   SP.P# = ‘p2’
```

شبیه سازی عملگر پیوند

ضرب دکارتی در SQL

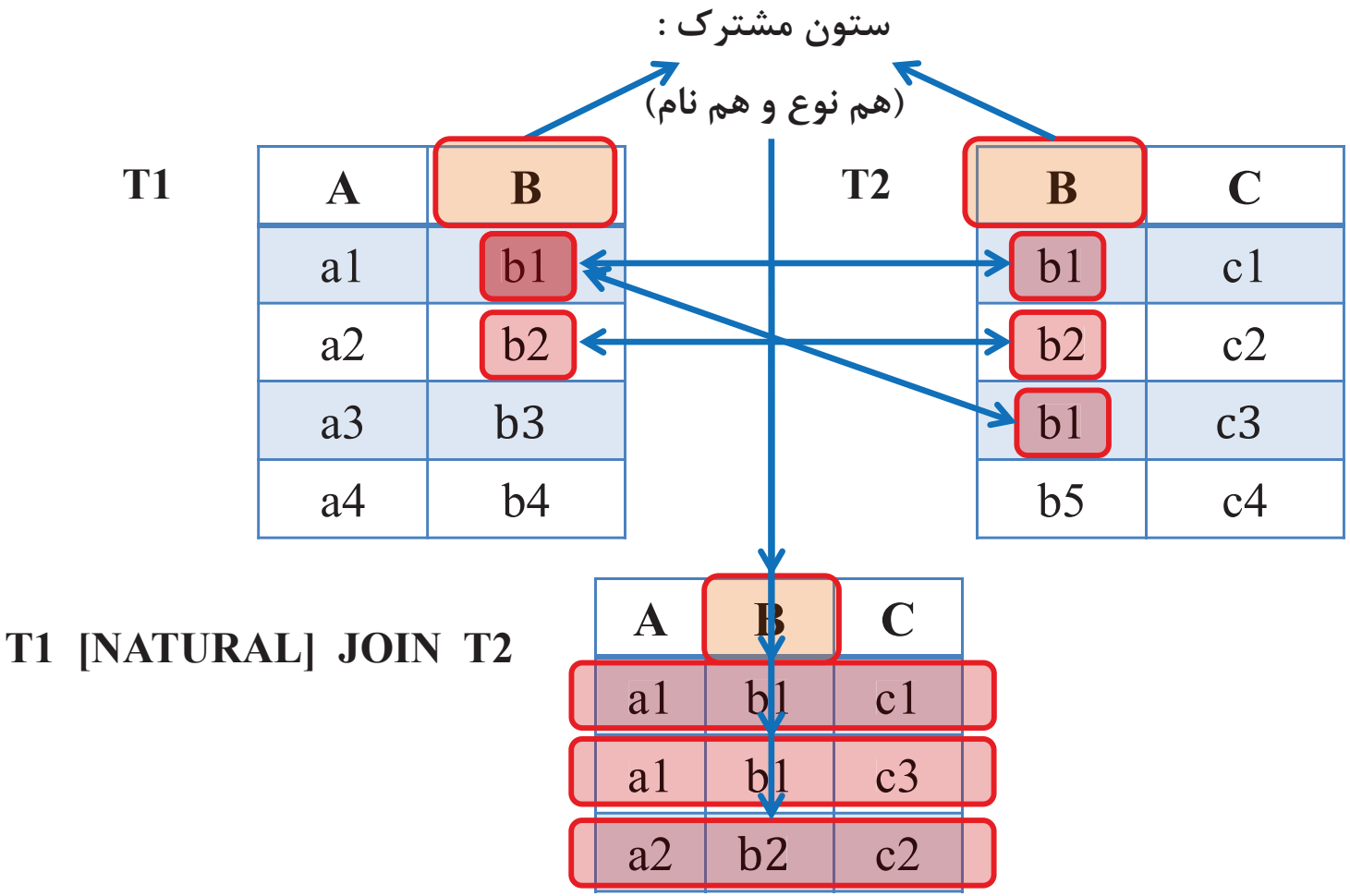
```
SELECT  T1.* , T2.*
        FROM   T1, T2
```

مکانیزم اجرا از دید برنامه‌ساز:

- به ازای هر سطر جدول S، بررسی می‌کند که آیا S# آن در SP وجود دارد یا نه و P# آن سطر در SP، p2 است یا نه. اگر درست بود SNAME آن سطر جزو جواب است.

بازیابی از بیش از یک جدول – عملگر پیوند یا JOIN

- **پیوند:** ارائه مقدماتی (غیر ریاضی)
- T1 [NATURAL] JOIN T2



بازیابی از بیش از یک جدول – عملگر پیوند یا JOIN (ادامه)

توضیح مقدماتی عملگر پیوند:

- صرف نظر از جزئیات تئوریک، سطرهای دو جدول را که مقدار ستون(های) مشترکشان یکسان است، به هم پیوند می‌زند.

روش دوم

نام تهیه کنندگان قطعه ’p2‘ را بدهید:

```
SELECT  SNAME
        FROM    S  [NATURAL]  JOIN  SP
        WHERE   P# = ‘p2’
```

S

S#	SNAME	…
s1	sn1	…
s2	sn2	…
s3	sn3	…
s3	sn4	…
…	…	…

SP

S#	P#	QTY
s1	p1	100
s1	p2	120
s1	p3	500
s2	p1	50
…	…	

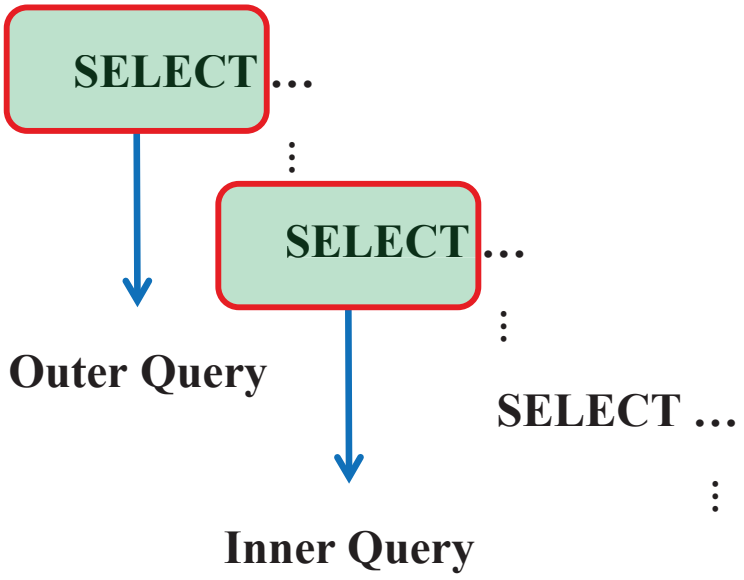
S [NATURAL] JOIN SP

S#	SNAME	…	P#	QTY
s1	sn1	…	p1	100
s1	sn1	…	p2	120
s1	sn1	…	p3	500
s2	sn2	…	p1	50
…	…	…		

بازیابی از بیش از یک جدول – زیرپرسش

زیر پرسش یا SubQuary

تعریف: یک SELECT است در درون SELECT دیگر. ← پرسش تو در تو



بازیابی از بیش از یک جدول – عملگر تعلق

IN و NOT IN : عملگر تعلق

مثال

روش سوم

```
SELECT  SNAME
        FROM   S
        WHERE  S#  IN  (SELECT  S#   FROM   SP
                                     WHERE  P# = 'p2' )
```

روش چهارم

یا

= ANY

روش پنجم

یا

= SOME

عملگر تعلق

مکانیزم اجرا:

- سیستم ابتدا SELECT درونی را اجرا می‌کند، آنگاه به ازای هر سطر S بررسی می‌کند که S# در مجموعه جواب SELECT درونی هست یا نه.

بازیابی از بیش از یک جدول – پرسش های بهم بسته

تعریف

دو پرسش درونی و بیرونی (در یک پرسش تو در تو) را **بهم بسته (Correlated)** گوییم هرگاه در کلاز WHERE پرسش درونی به ستونی از جدول موجود در کلاز FROM پرسش بیرونی، ارجاع داشته باشیم.

- **توجه:** نحوه اجرای پرسش‌های **بهم‌بسته** با طرز اجرای پرسش‌های **نابهم‌بسته** متفاوت است: در حالت بهم‌بسته، سیستم پرسش درونی را به ازای هر سطر از جدول پرسش بیرونی یک بار اجرا می‌کند.

مثال

روش ششم

```
SELECT  SNAME
        FROM   S
        WHERE  'p2'  IN  ( SELECT  P#   FROM   SP
                           WHERE   SP.S# = S.S# )
```

روش هفتم یا = ANY

روش هشتم یا = SOME

زیرپرسش بهم‌بسته یا CORRELATED

بازیابی از بیش از یک جدول (ادامه)

❑ امکان

$$\left\{\begin{array}{ll} \text{theta} & \text{ANY} \\ \text{theta} & \text{SOME} \\ \text{theta} & \text{ALL} \end{array}\right\}$$

$$\text{theta} \in \left\{\begin{array}{l} = \\ \neq \\ < \\ \leq \\ \geq \\ > \end{array}\right\}$$

مثال: شماره تهیه کنندگانی را بدهید که مقدار وضعیت آنها بیشینه نباشد.

```
1- SELECT   S#
          FROM   S
          WHERE   STATUS < ANY ( SELECT  DISTINCT   STATUS   FROM   S )
```

چون جواب SELECT تک مقداری است نیازی به ANY نیست.

```
2- SELECT   S#
          FROM   S
          WHERE   STATUS < ( SELECT   MAX (STATUS)   FROM   S)
```

بازیابی از بیش از یک جدول (ادامه)

مثال

روش نهم

```
SELECT  SNAME
        FROM   S
        WHERE  0 < ( SELECT  COUNT(*)
                     FROM   SP
                     WHERE  SP.S#  =  S.S#
                            AND
                            SP.P# = 'p2' )
```

سور وجودی (از حساب رابطه‌ای)

EXISTS و NOT EXISTS

- امکان بررسی وجود یا عدم وجود سطر در جدول بازگشتی

روش دهم

```
SELECT  SNAME
        FROM   S
        WHERE  EXISTS  ( SELECT *
                         FROM   SP
                         WHERE  SP.S# = S.S#
                                AND
                                SP.P# = 'p2' )
```

روش‌های دیگر ؟

عملیات ذخیره‌سازی

دستورهای INSERT, UPDATE, DELETE

درج INSERT:

INSERT INTO *table-name* **[***(col1,col2, ...)***]**
VALUES (*one row*) | *subquery*

بهنگام‌سازی UPDATE:

UPDATE *table-name*
SET *col = value / experession [, col = value / experission] ...*
⋮
WHERE *condition(s) / subquery*

حذف DELETE:

DELETE FROM *table-name*
WHERE *condition(s) / subquery*

مثال درج سطری (سطر کامل – سطر ناقص):

```
INSERT   INTO   STT
         VALUES    ( '222' , 'st2' , 'IT' , 'bs' , 'D17' )
 INSERT   INTO   STT
         VALUES    ( '333' , 'st3' , Null , 'ms' , Null )
```

مثال درج گروهی:

اطلاعات دانشجویان مقطع کارشناسی ارشد رشته کامپیوتر در جدول موقت T1 درج شود.

```
CREATE    TEMPORRAYR   TABLE   T1
         ( STN,  .... )

INSERT     INTO     T1
         ( SELECT    STT.*
                  FROM    STT
                           WHERE    STJ = 'comp'
                                    AND
                                    STL = 'ms'  )
```

بهنگام‌سازی چند سطر:

تعداد واحد تمام درس‌های عملی گروه آموزشی D11 را برابر یک کن.

```
UPDATE    COT
    SET    CREDIT = '1'
        WHERE    COTYPE = 'p'  AND    CODEID = 'D11'
```

بهنگام‌سازی در بیش از یک جدول:

```
UPDATE    STT
    SET    STID = 88104444
        WHERE    STID = 88107777

UPDATE    STCOT
    SET    STID = 88104444
        WHERE    STID = 88107777
```

اگر دستور دوم اجرا نشود؟

نمره دانشجویان گروه آموزشی D111 در درس ’com222‘ در ترم دوم سال ۸۵-۸۶ را ناتمام اعلان کن.

```
UPDATE    STCOT
    SET    STCOT.GRADE = ‘U’
        WHERE    STCOT.TR = ‘2’   AND   STCOT.YRYR = ‘85-86’
                AND    STCOT.COID = ‘COM222’
                AND    STID  IN  ( SELECT    STID
                                   FROM    STT
                                       WHERE    STT.STDEID = ‘D111’) ;
```

حذف

مثال: حذف تکدرس: درس com111 را برای دانشجوی 88104444 حذف کنید.

```
DELETE   FROM   STOCOT
         WHERE   STID = 88104444
                 AND
                 COID = 'COM111'
```

کجای؟: آیا این حذف باید انتشار یابد؟

مثال: **حذف از بیش از یک جدول:**

```
DELETE   FROM   DEPT
         WHERE   DEID = 'D333'

UPDATE    STT
    SET    DEID = 'Null'
         WHERE   DEID = 'D333'
```

به نام آنکه جان را فکرت آموخت

بخش هشتم:
طراحی پایگاه داده رابطه‌ای

مرتضی امینی

نیمسال دوم ۹۴-۹۵

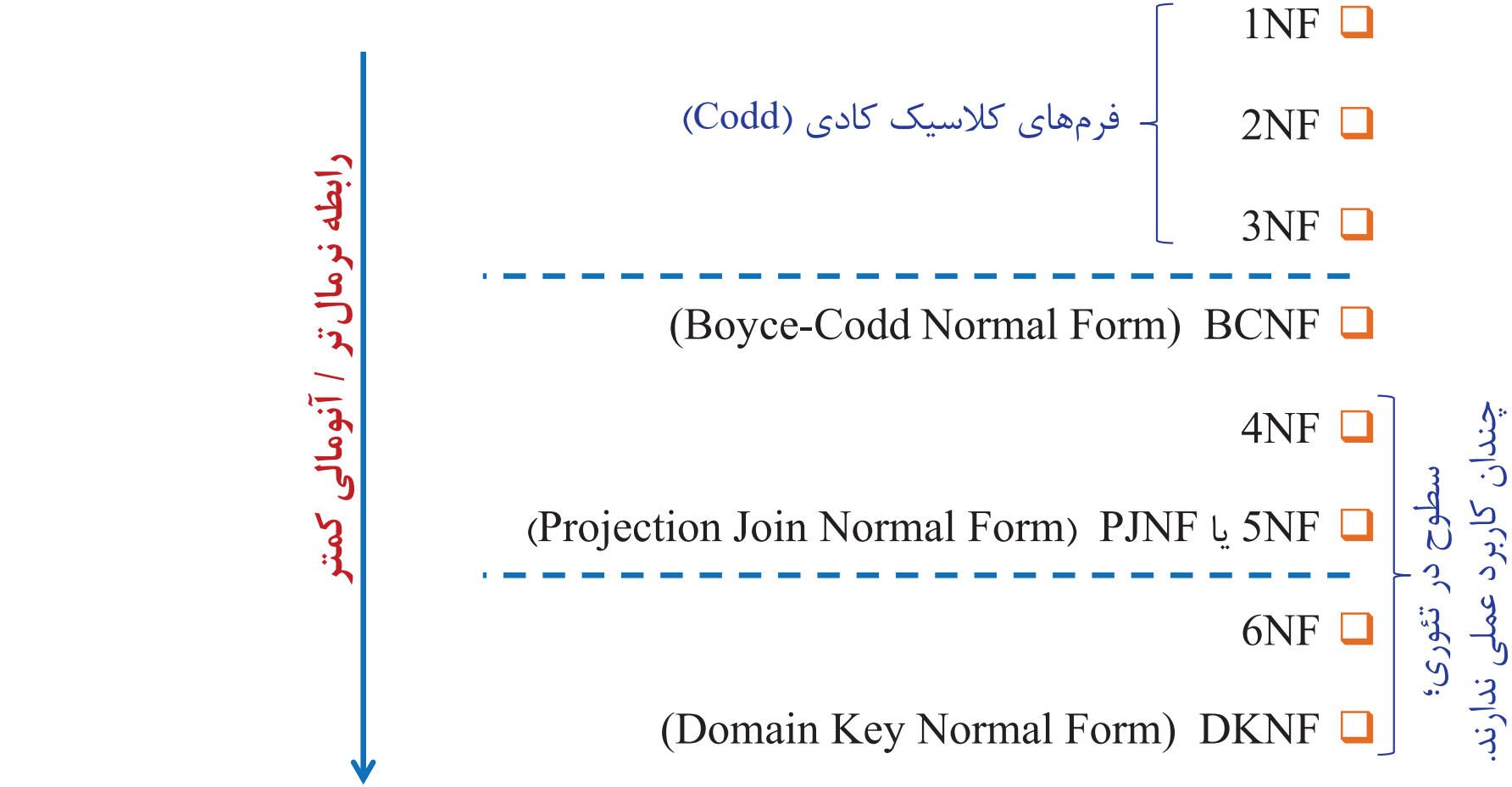
(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)

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.

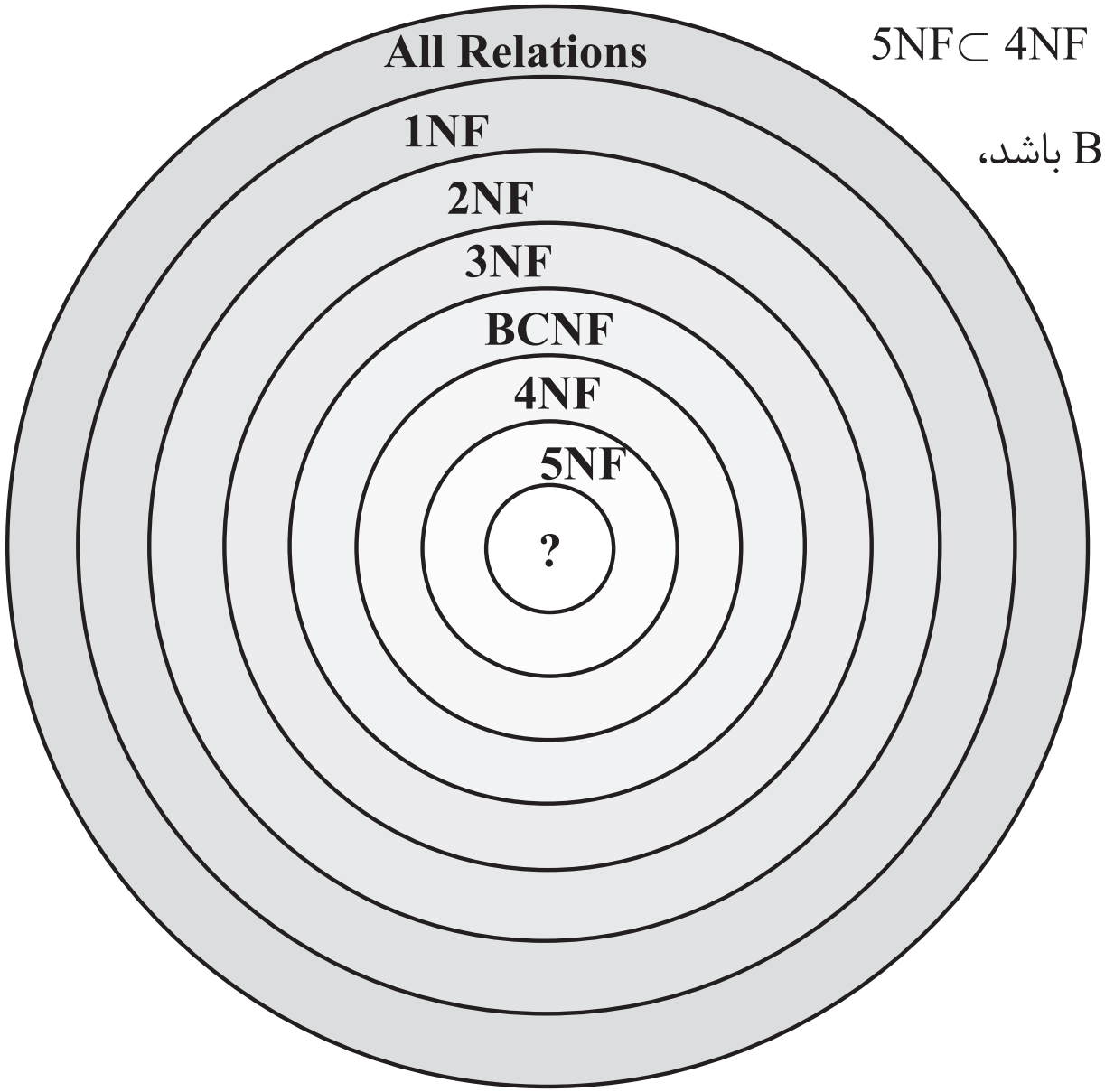
فرم‌های نرمال

- نرمال بودن رابطه (نرمالیتی)، فرم‌ها (صورت‌ها/ سطوح/ درجات) [NF: Normal Forms] مختلفی دارد.
- **فرم‌های نرمال:**



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

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:

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

مثال: 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.

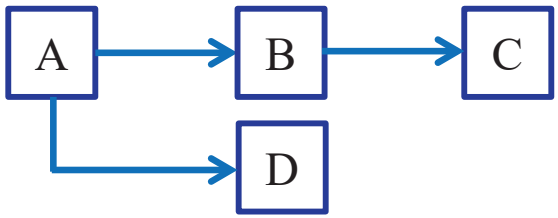
(۴) اگر K در رابطه R، SK یا CK باشد و $G \subseteq H_R$ آنگاه داریم: $K \rightarrow G$.

(۵) نمایش FDهای رابطه R به روشهای مختلف:

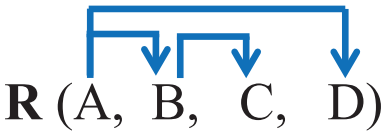
- به صورت یک مجموعه:

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

- 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.

وابستگی تابعی– قواعد آرمسترانگ

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)

۳- محاسبه مجموعه کاهش‌ناپذیر FDهای یک رابطه

سه شرط دارد:

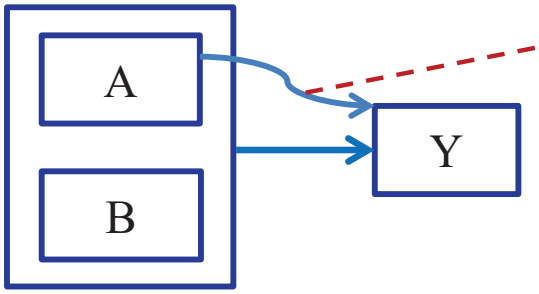
۱- هیچ FD در آن افزونه نباشد.

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.



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

فرم‌های نرمال کلاسیک کادی

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

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

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

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.

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)

مثال

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:

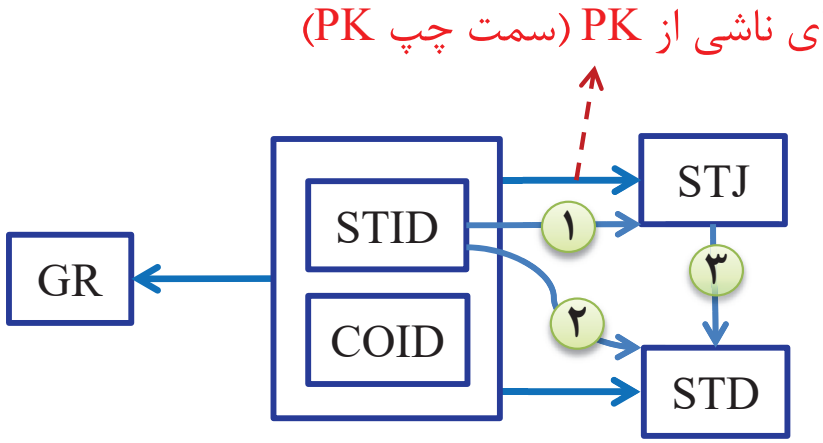
۱- یک دانشجو در یک رشته تحصیل می‌کند.

۲- یک دانشجو در یک دانشکده تحصیل می‌کند.

۳- یک رشته در یک دانشکده ارائه می‌شود.

R (<u>STID, COID</u>, 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

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

- 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.

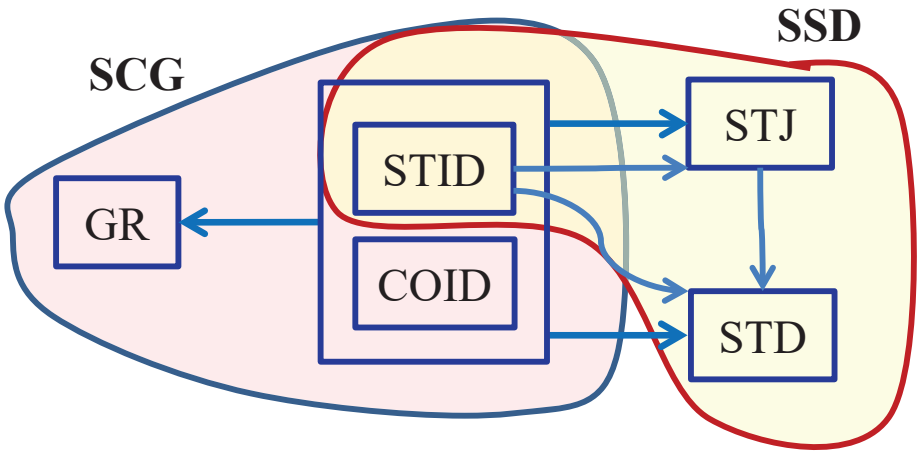
فرم‌های نرمال کلاسیک کادی (ادامه)

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

SCG (STID, COID, GR) و **SSD (STID, STJ, STD)**

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

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.

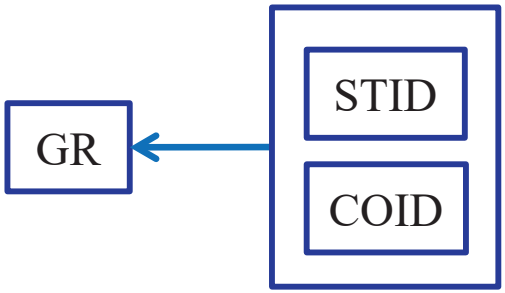
۲- حذف کن: ⟨’444’, ‘CO1’, 13⟩

بدون مشکل از SCG حذف می‌شود.

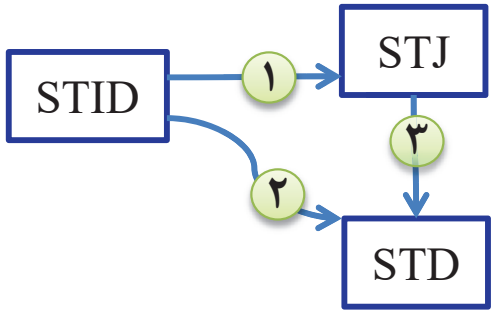
۳- بهنگام‌سازی کن: تغییر رشته دانشجوی 777 را به Chem

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. $R = SCG \bowtie SSD$

- 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 ⟨'Chem', '666'⟩ 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:

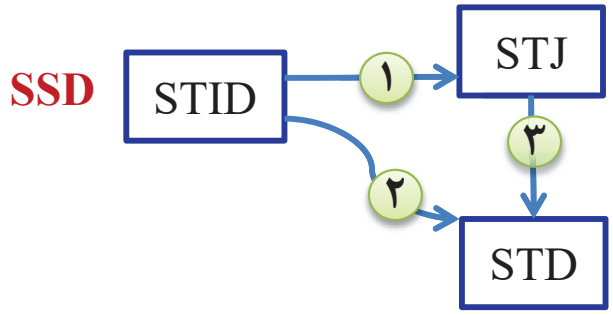
«شماره دانشکده رشته فیزیک را عوض کنید». به تعداد تمام دانشجویان این رشته باید بهنگام‌سازی شود.

SSD باید نرمال‌تر شود.

فرم‌های نرمال کلاسیک کادی (ادامه)

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)

STJ	STD
Phys	D11
Math	D12

و

SJ (STID, STJ)

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.

پیش‌فرض یا بدیهی

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

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

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.

[بحث تکمیلی] تجزیه خوب (ادامه)

- 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

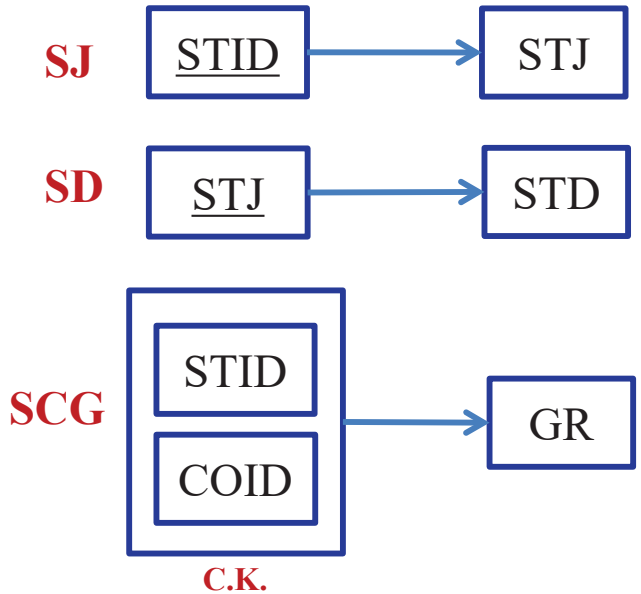
- **اصطلاح:** در وابستگی تابعی A→B (A Determines B) به A دترمینان گویند.

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.

مثال: 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.

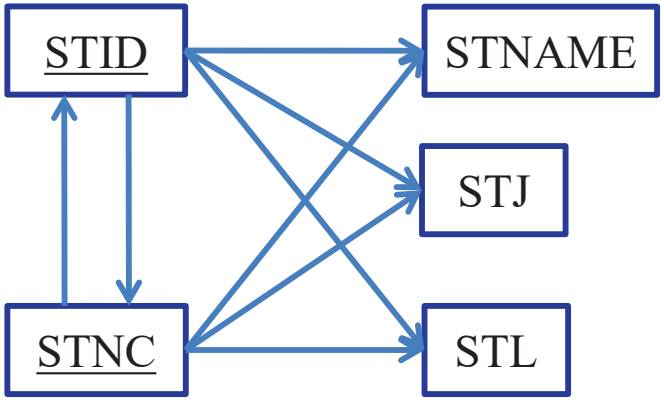
فرم نرمال BCNF (ادامه)

مثال برای حالت 1-II

دو دترمینان، هر دو هم CK هستند.

ST (STID, STNAME, STNC, STJ, STL, …)

C.K. C.K.

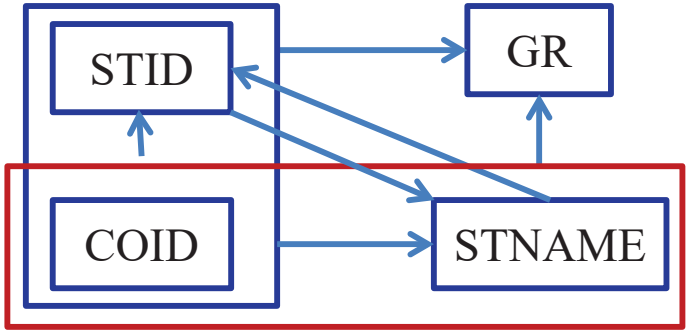


مثال برای حالت 2-II

(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.