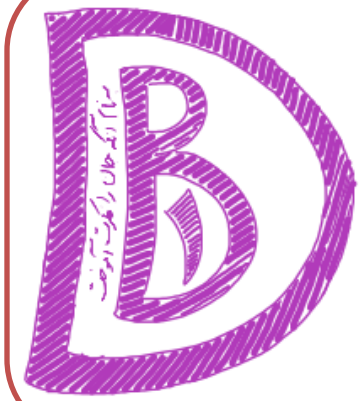
به نام آنکه جان را فکرت آموخت



معرفی درس:

طراحی پایگاه داده‌ها (۴۰۳۸۴)

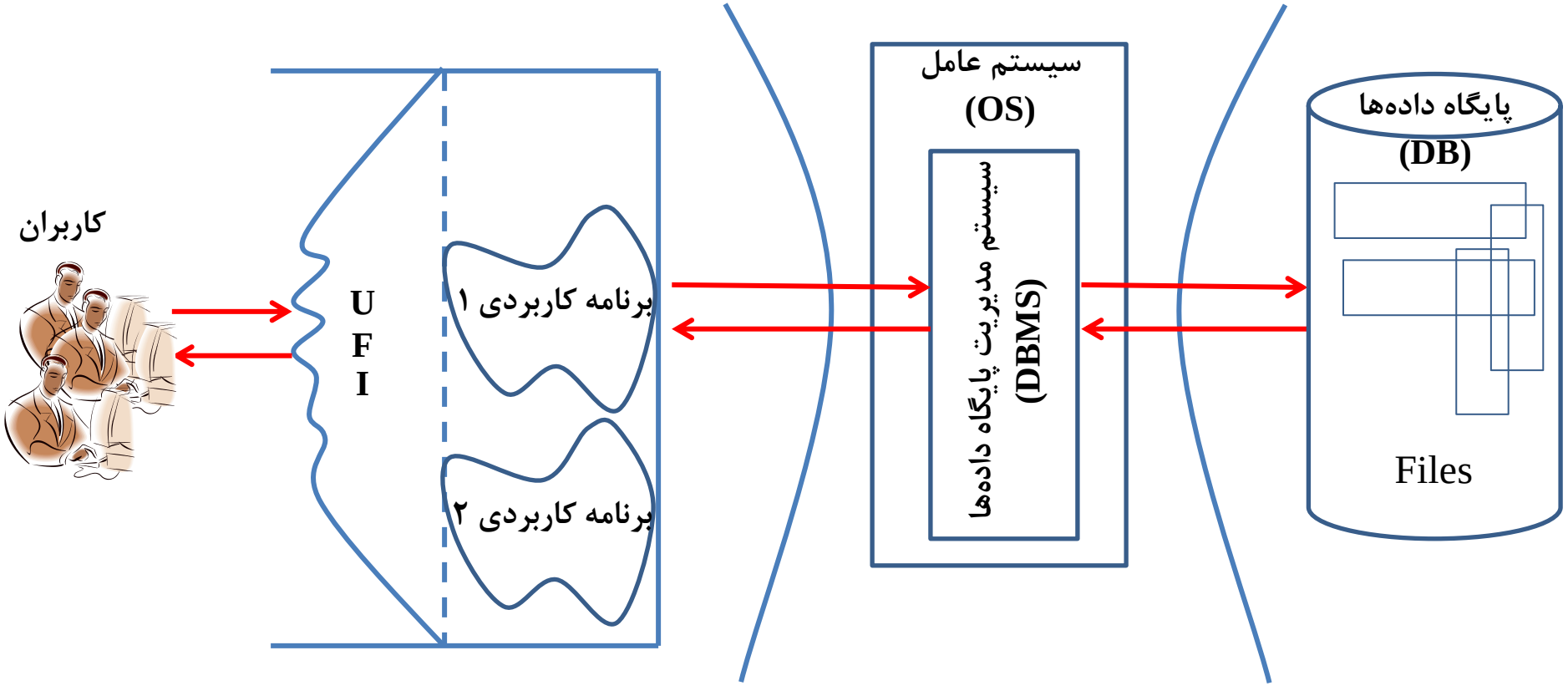
مرتضی امینی

نیمسال اول ۹۲-۹۳

مقدمه

- وجود حجم زیادی از داده‌ها و اطلاعات ذخیره شده ← پایگاه داده‌ها
- توسعه سیستم‌های اطلاعاتی یا برنامه‌های کاربردی برای استفاده از اطلاعات
- سیستمی برای ذخیره، جستجو، بازیابی و به‌روزرسانی اطلاعات ← سیستم مدیریت پایگاه داده‌ها

سیستم‌های اطلاعاتی پایگاهی



تولید سیستم پایگاهی

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

امکانات موردنیاز در ایجاد پایگاه داده‌ها

❑ **سوال:** در مدلسازی، طراحی و پیاده‌سازی پایگاه داده‌ها چه امکاناتی نیاز است؟

یک روش و زبان مدلسازی داده‌ها

روش نمودار روابط موجودیت‌ها (ER)

یک مدل داده‌ای برای طراحی منطقی

مدل رابطه‌ای و جدولی

یک زبان برای انجام عملیات

جبر رابطه‌ای و حساب رابطه‌ای

یک زبان برای تعریف، کنترل و انجام عملیات پایگاهی

زبان SQL

یک سیستم مدیریت پایگاه داده‌ها

...

سرفصل‌های درس

معرفی درس طراحی پایگاه داده‌ها | ۶

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، 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

۵- سیستم مدیریت پایگاه داده‌ها (DBMS)

- ریزفعالیت‌های ایجاد سیستم پایگاهی، مزایا و معایب تکنولوژی پایگاهی، استقلال داده‌ای فیزیکی و منطقی، 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, 5th Edition, 2007.
- **Database Systems**, By T. Connolly and C. Begg, 4th Edition, 2005.
- **Database Management Systems**, By R. Ramakrishnan and J. Gehrke, 2nd Edition, 2000.
- **Database System Concepts**, By A. Silberschartz, H.F. Korth and S. Sudarshan, 5th Edition, 2006.

شیوه ارزیابی

- میان‌ترم (۶/۵ نمره)
- پایان‌ترم (۷/۵ نمره)
- تمرین‌های نظری (۲/۵ نمره)
- پروژه عملی – مستمر در طی ترم (۲/۵ نمره)
- کوئیزهای موردی (۱ نمره)
- نمرات تشویقی - ارائه مطلب، کار اضافه، حل تمرینهای اضافی، فعال بودن در کلاس، ... (۱ نمره)

جمـــــع: ۲۱ نمره !

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش اول :
مقدمه

مرتضی امینی

نیمسال اول ۹۲-۹۳

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

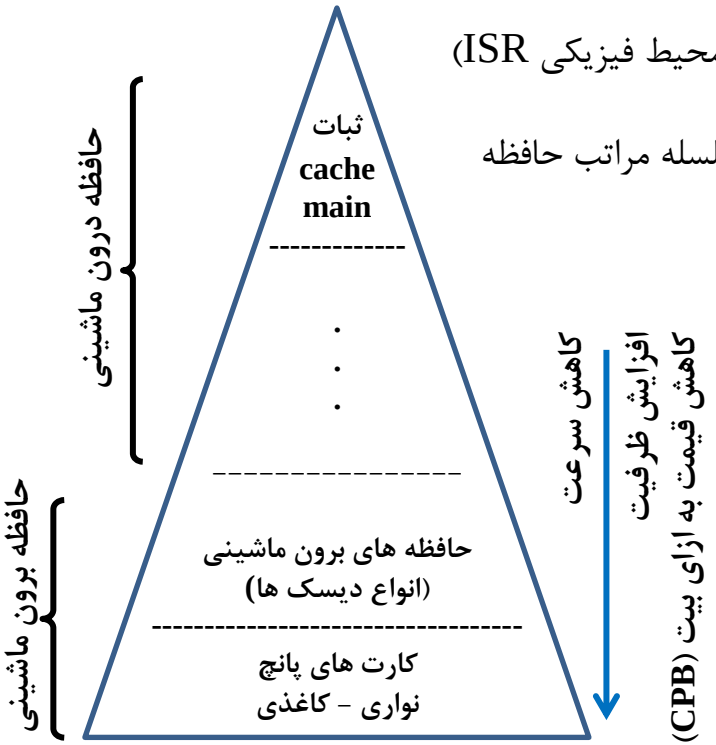
- هر سیستم نرم‌افزاری از مجموعه ای از داده‌های ذخیره شده استفاده می‌کند.

در قالب تعدادی فایل (محیط فیزیکی ISR)

در کجا؟ ← در یک سلسله مراتب حافظه

به لحاظ ساختاری

- ساختمند (structured): well-formatted است، فرمت ثابت و از پیش تعیین شده دارد.
- نیم ساختمند (semi-structured)
- ناساختمند (un-structured)



انواع سیستم نرم‌افزاری:

- بنیادی یا پایه (سیستم‌های عامل)
- نیمه بنیادی (DBMS، DMS، کامپایلرها، اسمبلرها، و ...)
- کاربردی (برنامه‌های کاربردی)
- ابزاری: انواع toolها

محیط فیزیکی «ذ.ب.ا»

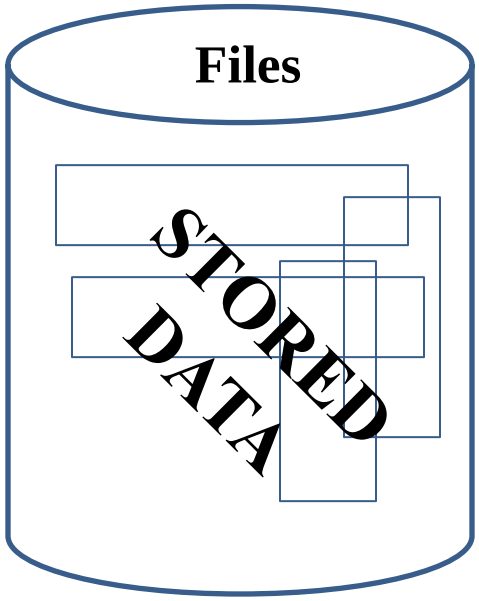
کنجکاوی: دلایل استفاده از این سلسله مراتب حافظه چیست؟

کنجکاوی: چه داده ای، برای چه مدتی، در کدامیک از مراتب سلسله مذکور قرار می‌گیرد؟

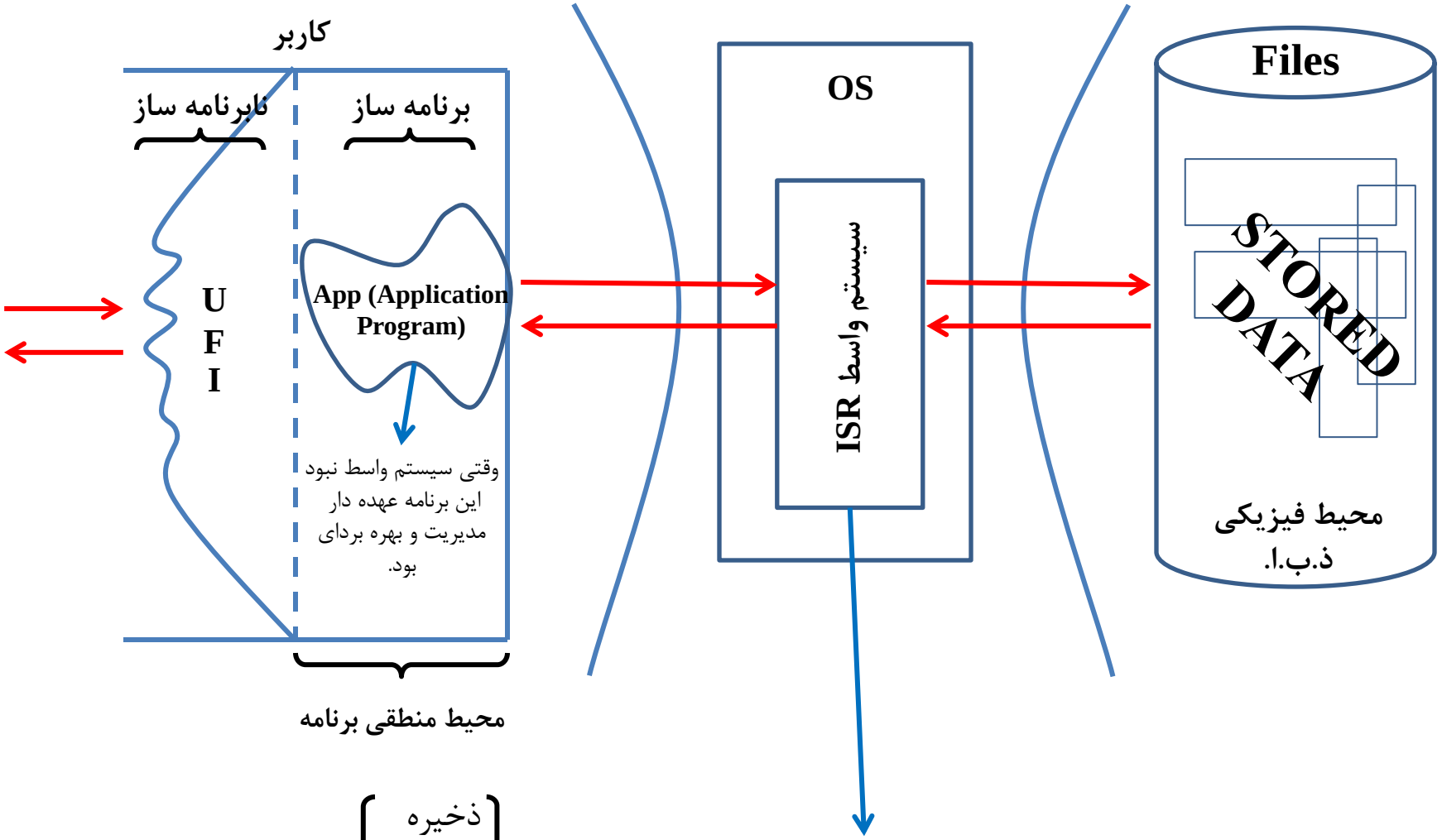
کنجکاوی: خصوصیات عمومی فایل‌ها چیست؟

- **محیط فیزیکی "ذ.ب.ا." (ذخیره و بازیابی اطلاعات) یا ISR (Information Storage and Retrieval)**

ISR: باید { ایجاد / مدیریت / بهره برداری } شود. ← نیاز به یک سیستم واسط ذ.ب.ا داریم.



سیستم واسط ذ.ب.ا.

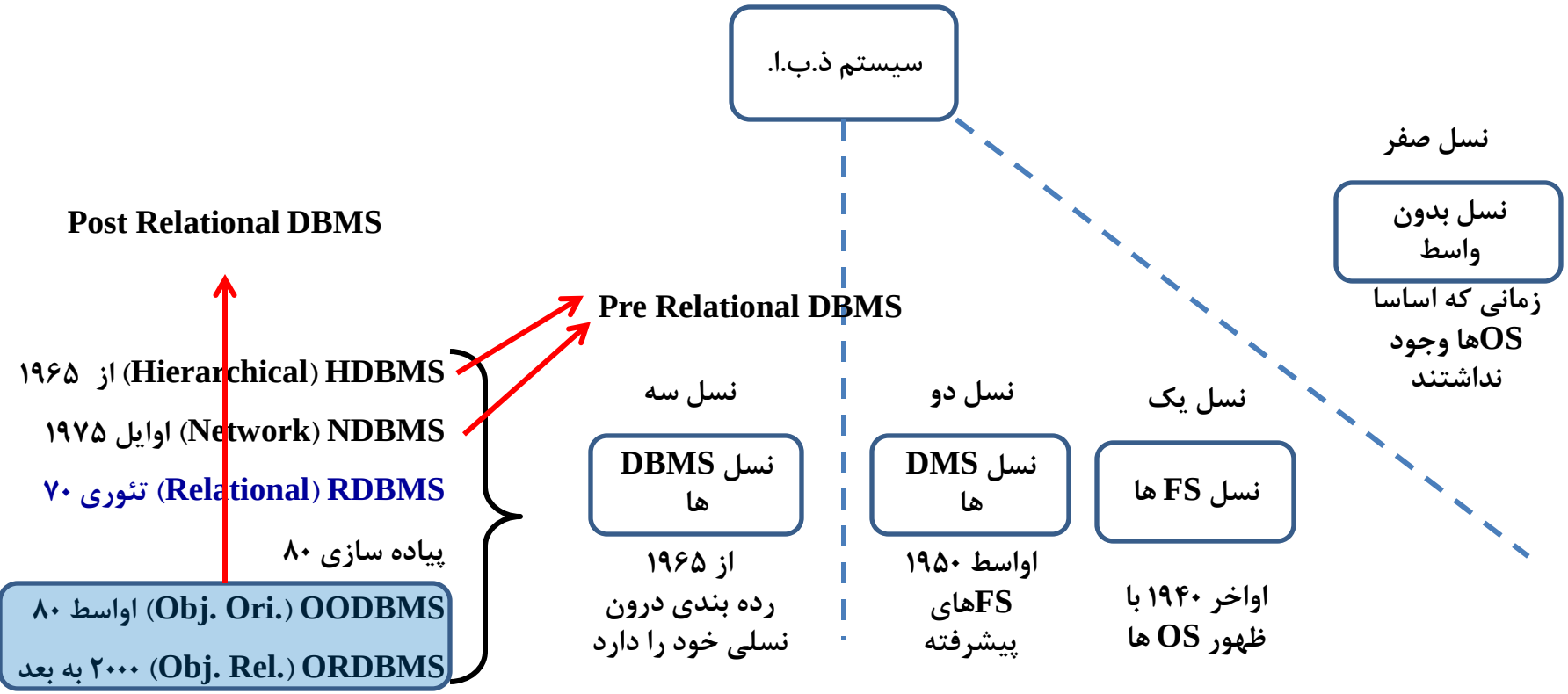


این سیستم این امکان را می‌دهد تا کاربر داده‌های خود را { ذخیره / بازیابی / پردازش / ... } کند.

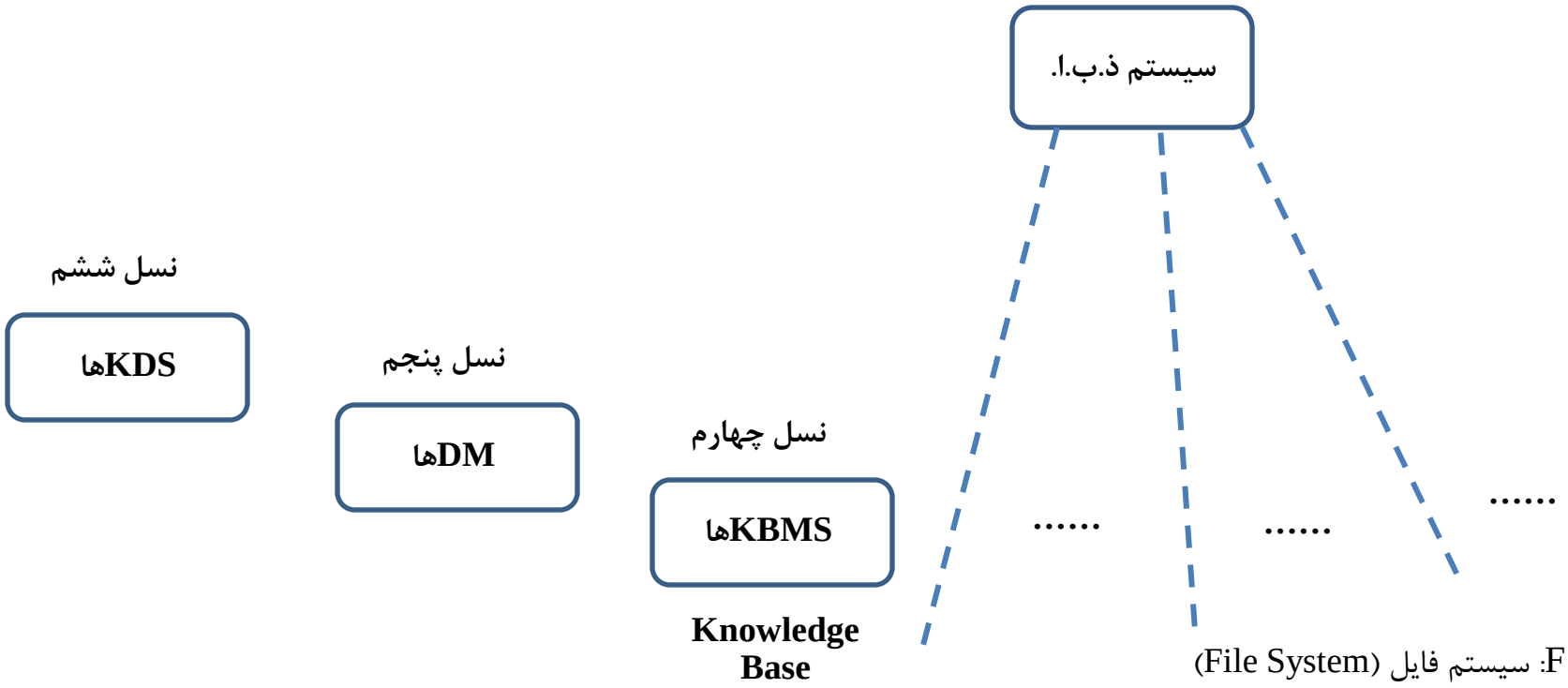
سیر تحول سیستم ذ.ب.ا. (۱)

کنجکاوی: رده بندی از مفهوم کاربر ارایه کنید؟ به بیان دیگر گونه های دیگر کاربر کدامند؟

- **سیستم واسط "ISR" سیر تحول خاص خود را دارد :**
- ۶ نسل تکنولوژیک قابل بازیابی است (به طور کلی) [دیدگاه نرم‌افزاری]



سیر تحول سیستم ذ.ب.ا. (۲)



- FS: سیستم فایل (File System)
- DMS: سیستم مدیریت داده‌ها (Data Management System)
- DBMS: سیستم مدیریت پایگاه داده‌ها (DataBase Management System)
- KBMS: سیستم مدیریت پایگاه دانش (Knowledge Base Management System)
- DM: سیستم داده‌کاوی (Data Mining System)
- KDS: سیستم کشف دانش (Knowledge Discovery System)

سیر تحول سیستم ذ.ب.ا. (۳)

- در این نسل‌بندی، نسل بعدی نسل قبلی را منسوخ نمی‌کند. نسل بعدی نسل قبلی را تکمیل می‌کند و از آن استفاده می‌کند.
- انواع نیازهای پردازشی، کنترلی، و عملیاتی سبب ایجاد نسل‌های سیستم «ذ.ب.ا.» شد.

داده – اطلاع – دانش

- **داده (Data)**
  - تعریف اول ANSI: نمایش بوده‌ها، پدیده‌ها، مفاهیم یا شناخته‌ها به طرزی صوری و مناسب برای برقراری ارتباط، تفسیر یا پردازش توسط انسان یا هر امکان خودکار
  - تعریف دوم ANSI: هر نمایشی اعم از کاراکتری (نویسه‌ای) یا کمیت‌های قیاسی که معنایی به آن قابل انتساب باشد (توسط انسان یا یک مکانیسم خودکار)
- **اطلاع (Information)**
  - تعریف دقیق و جامعی از مفهوم اطلاع وجود ندارد.
  - تعریف اول [LIPS92]: اطلاع، داده پردازش شده است.
  - تعریف دوم [روحا ۷۸-الف]: معنایی که انسان به داده منتسب می‌کند، از طریق قراردادهای شناخته شده‌ای که در نمایش داده به کار می‌روند.
  - برخی داده را همان مقدار واقعا ذخیره شده و اطلاع را معنای آن می‌دانند. بنابراین اطلاع دارای خاصیت اطلاع‌دهندگی و ارتباط‌دهندگی است، در حالیکه داده مجرد این خاصیت را ندارد.

دانش (Knowledge)

- تعریف [FROS87]: دانش عبارت است از نمایش نمادین جنبه‌هایی از بخشی از جهان واقع (جهان موردنظر یا محیط مطرح)
  - مثال: شنبه هوا بارانی است. حسن فرزند علی است.
- تعریف دوم [روحا ۹۱]: دانش منطقی نوعی شناخت است که از یک مجموعه از اطلاعات بر اساس یک مجموعه از قواعد مشخص، معمولا با روش استقراء حاصل می‌شود. حصول این شناخت می‌تواند توسط انسان یا یک سیستم خودکار انجام شود.

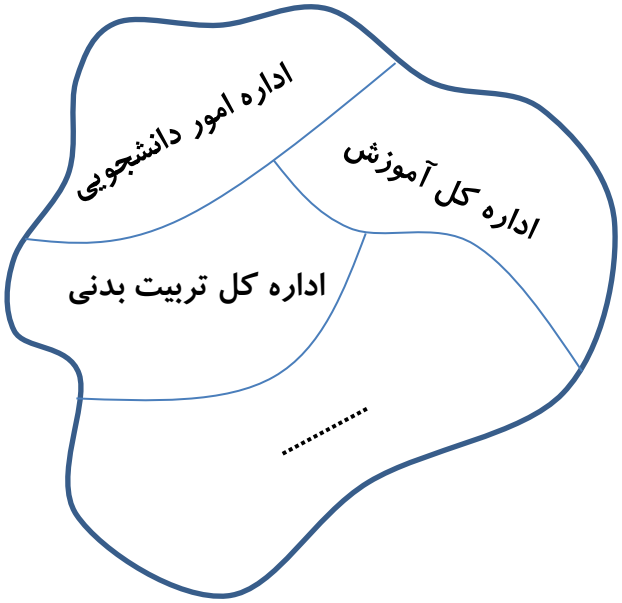
تعریف پایگاه داده:

مجموعه‌ای است از داده‌های ذخیره شده، پایا ، مجتمع ، به هم مرتبط ، و حتی الامکان فاقد افزونگی، (دارای معماری خاصّ خود، مبتنی بر یک مدل داده ای مشخص) ، تحت کنترل یک سیستم متمرکز، مورد استفاده یک یا چند کاربر در یک سازمان (در یک محیط) به طور اشتراکی و همروند.

Persistent
Integrated
Interconnected
Redundancy
Data Model
Shared
Concurrent
Centralized System
Multi User

پایگاه داده – مثال مقدماتی

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 داده‌ای پردازشی کاربران ناظر به آنهاست. (یعنی به داده‌هایی در مورد آنها نیاز دارند.)

مثال مقدماتی

- **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.

ادامه مثال مقدماتی (مشی فایلینگ)

کارهای لازم در مشی فایلینگ به طور خلاصه:

- توجه: این کارها معمولا برای هر زیرمحیط به طور جداگانه انجام می شود. ← تعدادی سیستم کاربردی جدا (نامجتمع) و بی ارتباط در یک محیط ...

۱- مطالعه و شناخت محیط

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

۳- تعیین مشخصات سیستم کاربردی System Specification

۴- [انتخاب پیکربندی سخت افزار و نرم افزار H/S]

۵- [انتخاب یک FS و/یا DMS]سیستم واسط ISR

۶- طراحی تعدادی فایل (طبق مشخصات سیستم)

۶-۱- تعیین فرمت رکورد

۶-۲- تعیین ساختار فایل

ساختار فایل: ساختاری که براساس آن فقره داده ها (رکوردها) در سطح منطقی [و/یا فیزیکی] با یکدیگر مرتبطند.

ساختار فایل یک امکان برای نمایش ارتباط بین فقره داده‌هاست (Data Items) خواه در سطح نمایش منطقی باشد یا فیزیکی.

کنجکاوی: چند نوع ساختار فایل وجود دارد؟

۶-۳- نحوه دسترسی به رکوردها – استراتژی دسترسی

۶-۴- اندازه فایل ها

۶-۵- میزان گسترش چه میزان باشد

۶-۶- ارتباط با فایل های دیگر

۶-۷- عملیات مجاز در فایل ها + کاربران

مثال مقدماتی (مشی فایلینگ)

- **کارهای لازم در مشی فایلینگ به طور خلاصه : (ادامه)**

۷- طراحی واسطهای کاربری (UFI)

۸- طراحی تعدادی برنامه کاربردی (Application Program) [ضمن تعیین تراکنش(ها)]

۹- تولید برنامه‌های { ایجاد / کنترل / پردازش } فایل‌ها

۱۰- ایجاد محیط فیزیکی «ذ.ب.ا.» به طور آزمایشی (برای داده های تست)

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

[۱۲- ایجاد محیط فیزیکی «ذ.ب.ا.» با داده‌های واقعی و حجم واقعی و انجام تست مرحله سوم]

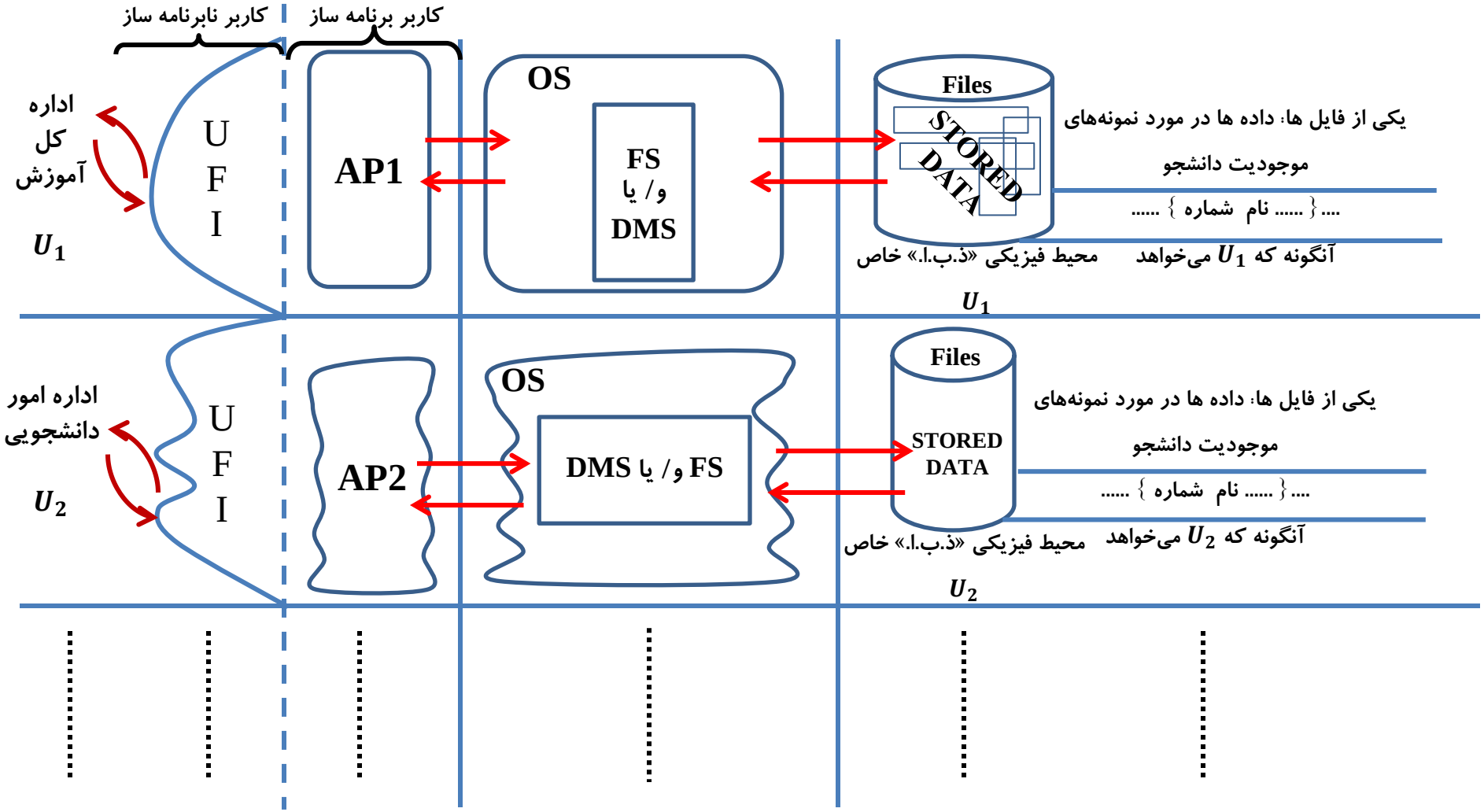
۱۳- رفع اشکال‌ها در هر مرحله

۱۴- ایجاد محیط فیزیکی واقعی با نصب، پیکربندی و ورود داده‌های اولیه (Data Entry)

۱۵- آغاز بهره برداری و نگهداری سیستم

۱۶- رفع معایب و بهینه‌سازی سیستم

نمایش شماتیک مشی فایلینگ



معایب 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 داده های 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)

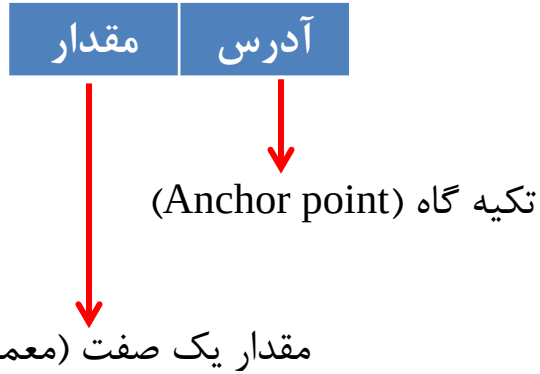
توضیح مفهوم افزونگی:

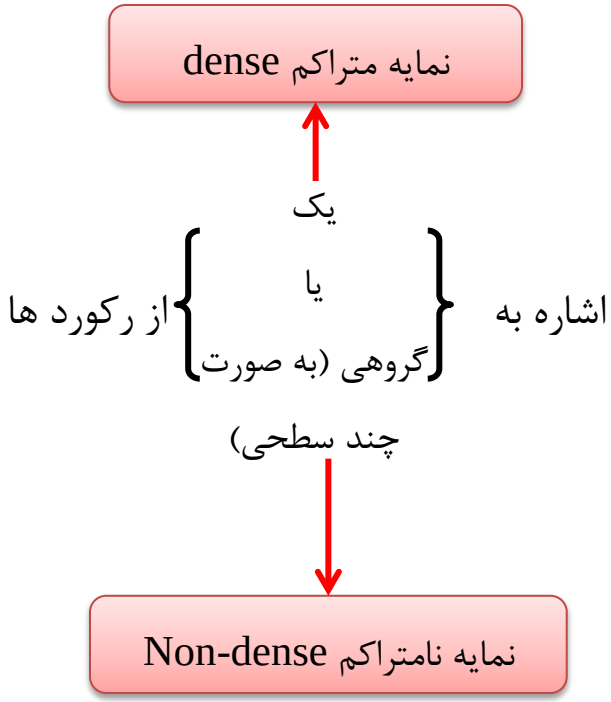
- **افزونگی در معنای محدود** (یعنی درون فایلی -intrafile redundancy - در مباحث فایلینگ)
  - عبارت است از تکرار ذخیره سازی مقادیر (value) یک صفت یا بیش از یک صفت در فایل داده‌ای یا فایل کمکی آن.
- این نوع افزونگی گونه‌هایی دارد:
  - ۱- **طبیعی:** ناشی از ماهیت داده های محیط (مثل صفت رشته دانشجو که برای دانشجویان مختلف می‌تواند یکسان و در نتیجه تکراری باشد)
    - کنجکاوی: برای کاهش مصرف حافظه در حالت افزونگی طبیعی چه باید کرد؟
  - ۲- **تکنیکی:** ناشی از استفاده از یک تکنیک *معمولا* برای افزایش سرعت (مثل نمایه سازی [شاخص بندی Indexing])

نمایه سازی (نمونه‌ای از افزونگی تکنیکی)

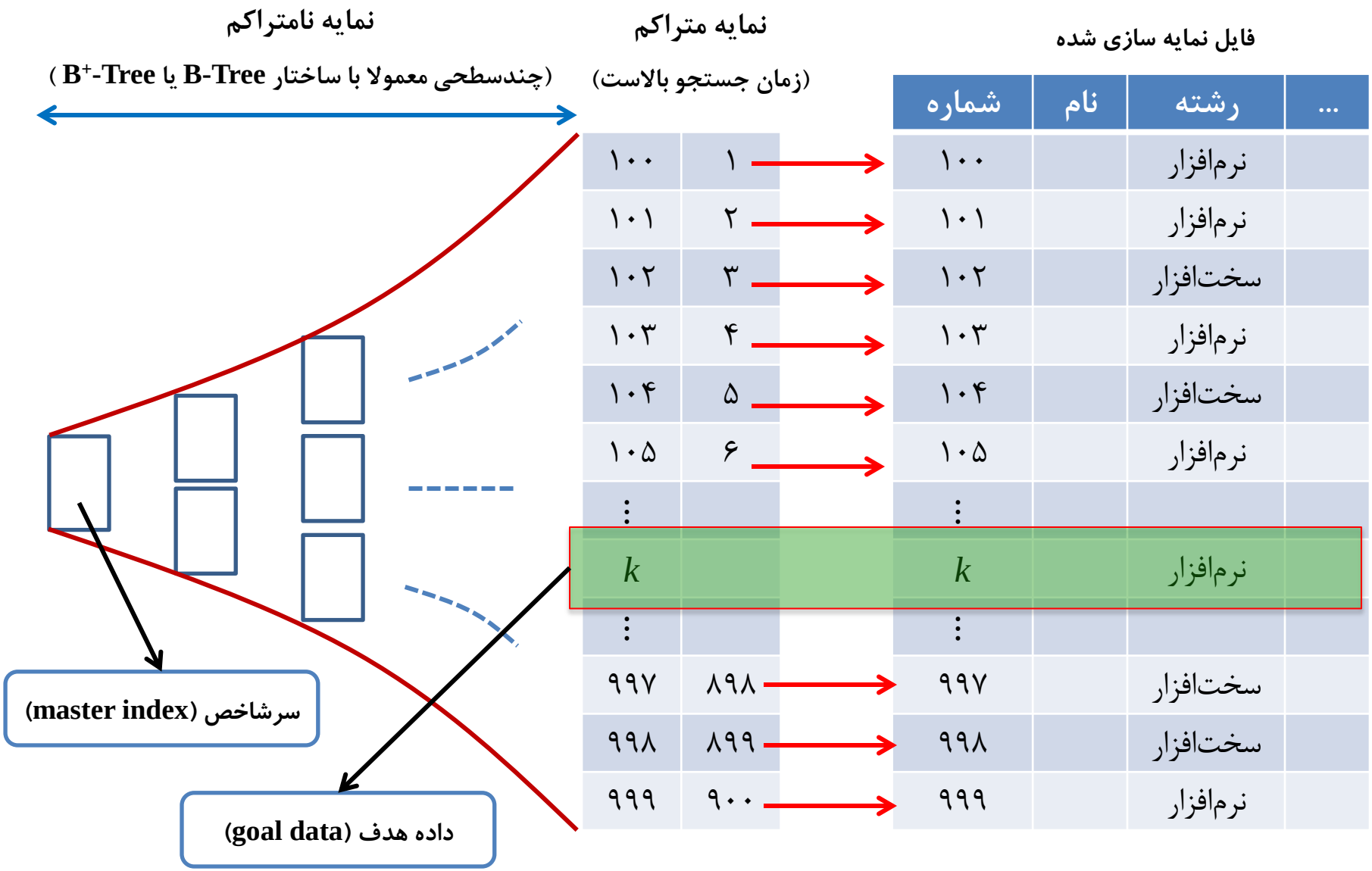
یادآوری

- تشکیل شده از تعدادی درایه (مدخل-entry)





نمایه سازی (ادامه)



افزونگی (ادامه)

- **افزونگی در معنای گسترده (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.
  - **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)

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)]

مجتمع ← به منظور ایجاد یک سیستم یکپارچه

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)

به طور مثال به منظور افزایش کارایی

۱۷- آغاز بهره برداری و نگهداری از سیستم

۱۸- گسترش سیستم

یکی از ویژگی های DBMS گسترش پذیری سیستم است.

۱۹- رفع معایب و بهینه‌سازی سیستم

کنجاوی؟

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

تراکنش Transaction:

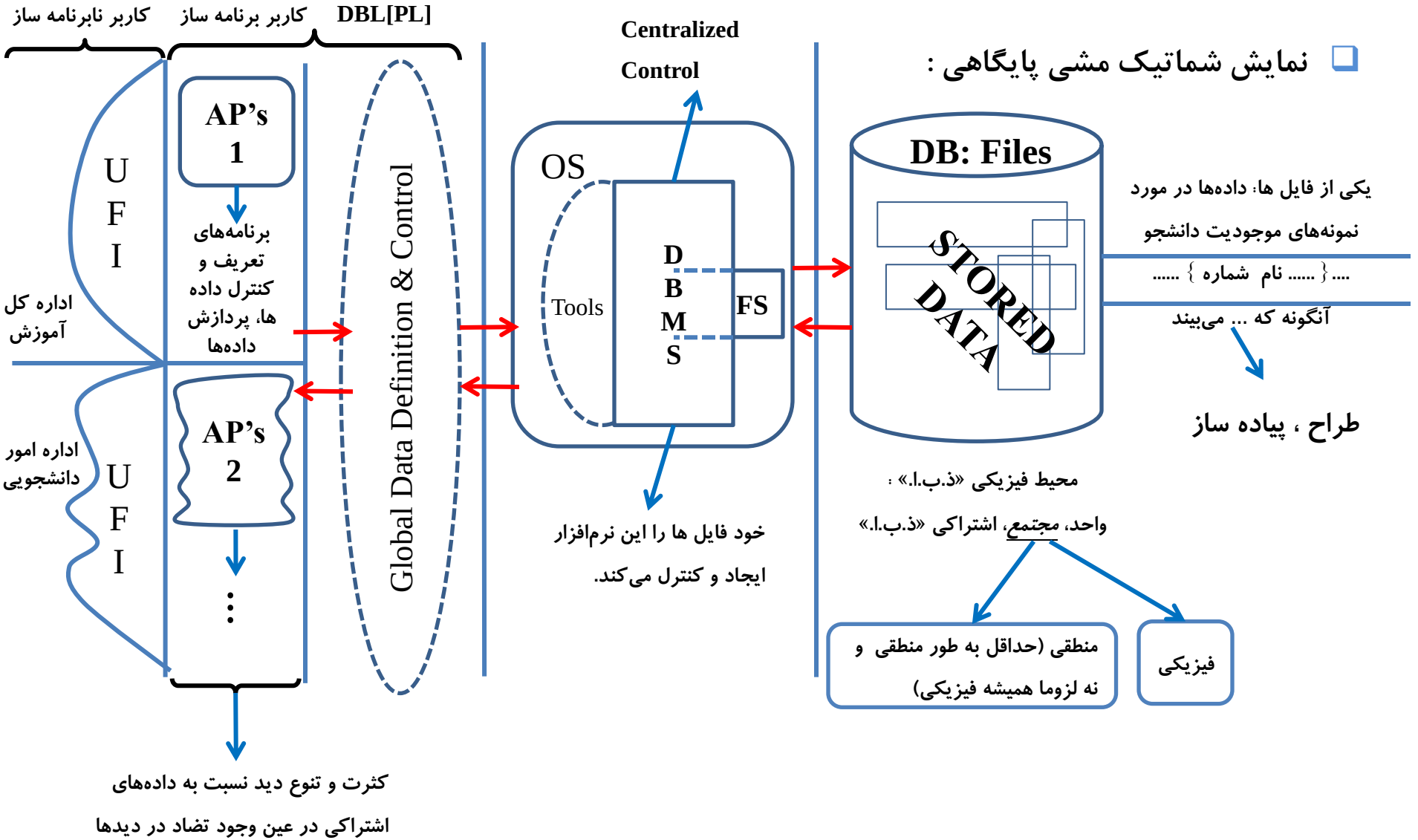
- دنباله ای از عملیات («قطعه برنامه») که معمولاً یک عمل تغییردهنده (درج، حذف، به روزرسانی) در محیط ذخیره‌سازی داده‌ها انجام می‌دهد و یا باید به تمامی اجرا شود و یا اجرا نشده تلقی می‌شود.
- دارای خواص ACID (**A**tomicity **C**onsistency **I**solation **D**urability)

یا همه یا هیچ — **سازگاری** — **انفراد و جدایی** — **دوام (پایداری)**

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
```

ادامه مثال مقدماتی (مشی پایگاهی)



- نمایش شماتیک مشی پایگاهی :

ادامه مثال مقدماتی (مشی پایگاهی)

چگونه از این کثرت دید می توان به آن «وحدت» رسید؟ ← ۱- خود نرم افزار DBMS
۲- معماری پایگاه داده

- تمرین: مزایای مشی پایگاهی چیست؟ ← (طبق معلومات فعلی: عکس معایب مشی فایلینگ)
- تمرین: چند سطح تعریف داده داریم؟

عناصر محیط پایگاهی

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 دخیره 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.

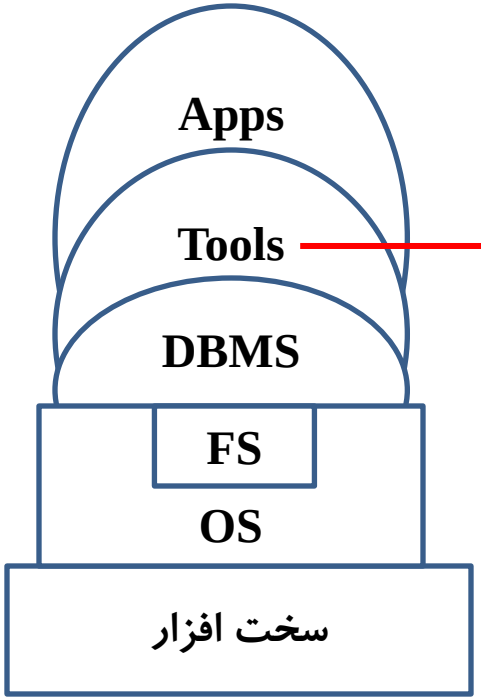
- تکنیک های تولید نسخه پشتیبان؟
- سطوح مختلف Back up

- **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

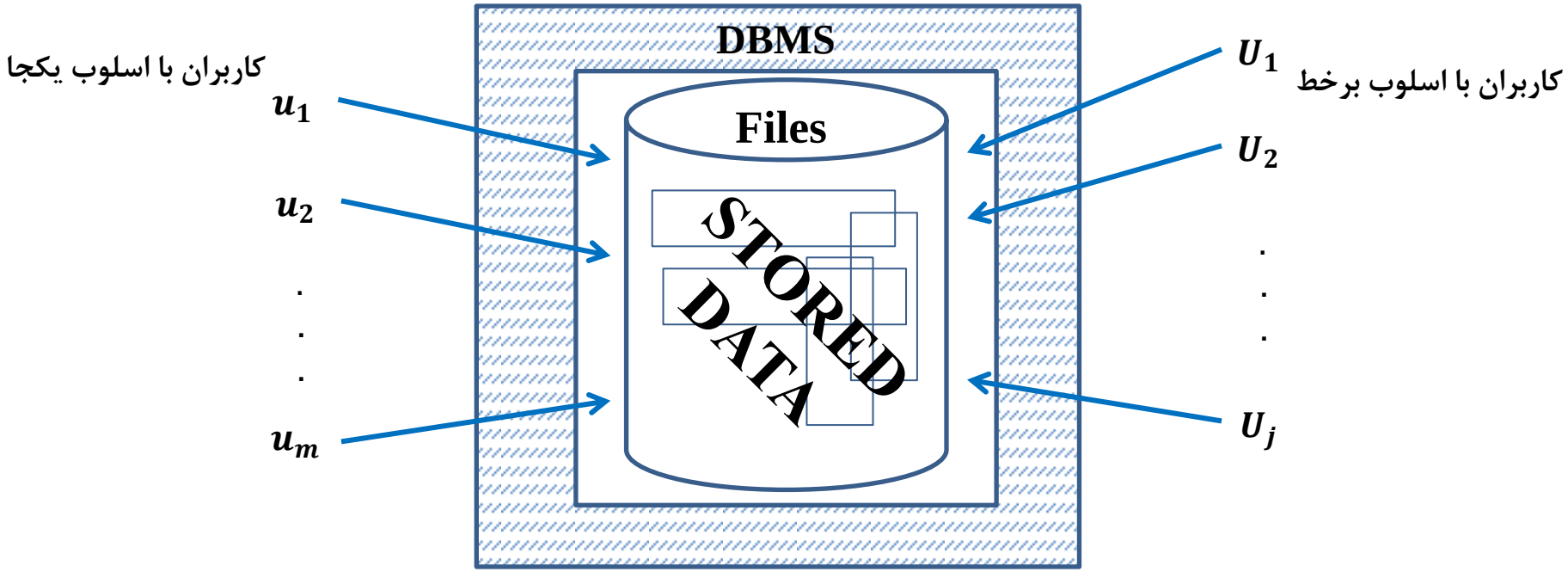
- **انواع نرم افزارهای مطرح در محیط پایگاهی:**
  - سیستم عامل و سیستم فایل (OS و FS)
  - سیستم مدیریت پایگاه داده‌ها (DBMS)
  - ابزارها (Tools)
  - برنامه‌های کاربردی (Apps)



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 – (۳) کاربر (انواع)

- **انواع کاربر از نظر اسلوب عملیاتی:**
  - **Batch- یکجا** (تعدادی برنامه یا پرس‌وجو جمع آوری می‌شود و به صورت یکجا به سیستم داده می‌شود و جواب آن بر می‌گردد.)
  - **Online – برخط – پیوسته** (یک برنامه یا پرس‌وجو به سیستم داده می‌شود، اجرا می‌شود، و جوابش برمی‌گردد.)
  - **Interactive-تعاملی –** بسته به اینکه چه جوابی داده شود عمل دیگری از کامپیوتر درخواست می‌شود.
    - Online لزوما Interactive نیست اما Interactive لزوما Online است.

- سیستم پایگاهی به صورت پیش فرض چند کاربره (multi-user) است.

عناصر محیط پایگاهی – (۴) داده

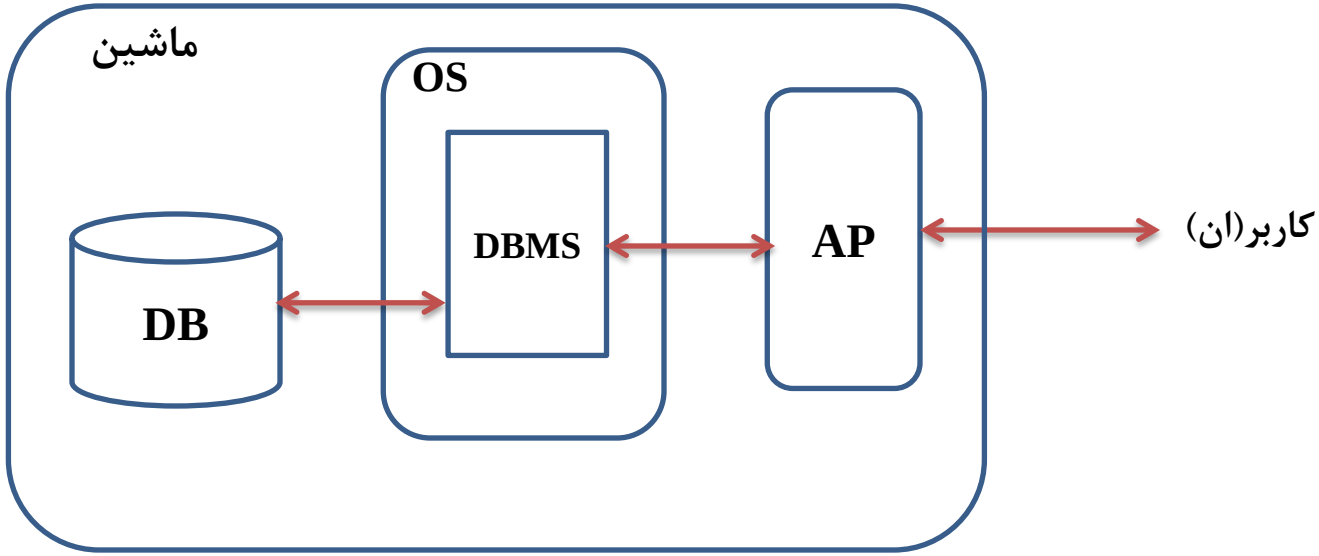
- **داده‌های ذخیره شده** در یک سیستم پایگاهی عبارتند از:
  - داده‌های کابران
  - داده‌های سیستمی
- مباحث مرتبط با داده در محیط پایگاهی در ادامه درس مطرح می‌گردد.

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
    - معماری موبایل

معماری متمرکز

- 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

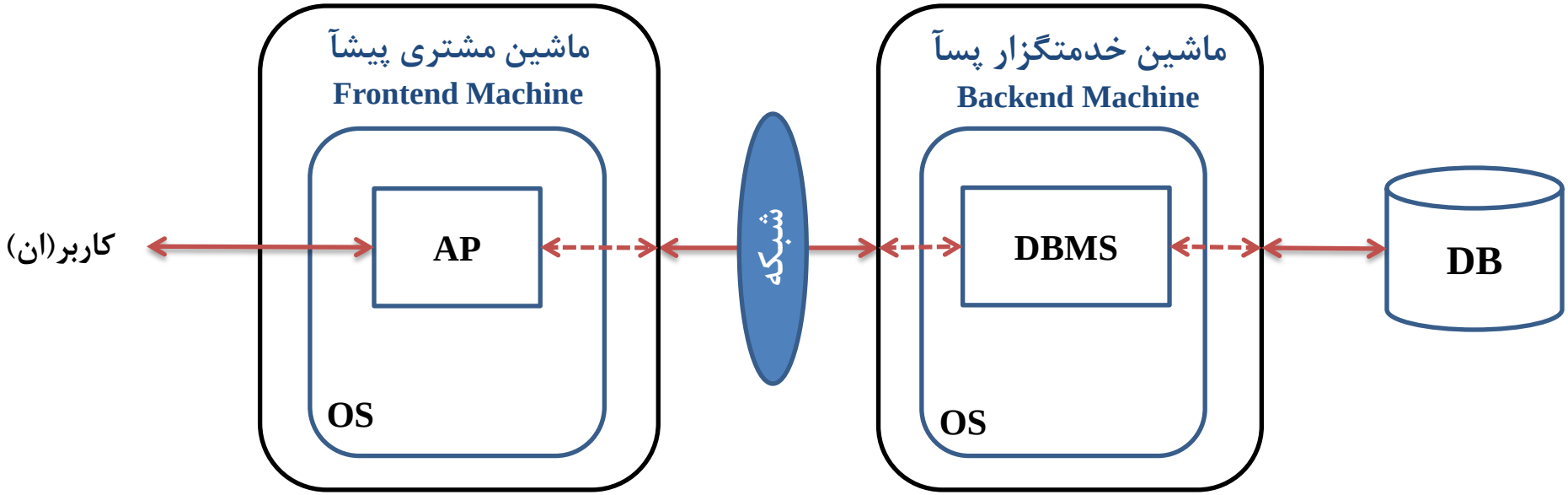
- **دلیل:** دلیل اصلی استفاده از معماری مشتری-خدمتگزار (Client-Server): تقسیم وظایف سیستم
- **تعریف:** هر ماشینی (فیزیکی یا منطقی) که خدمتی را به ماشین دیگر بدهد، **خدمتگزار** نامیده می‌شود.

نمونه‌هایی از انواع خدمتگزارها: File Server، Print Server، Message Server، DB Server

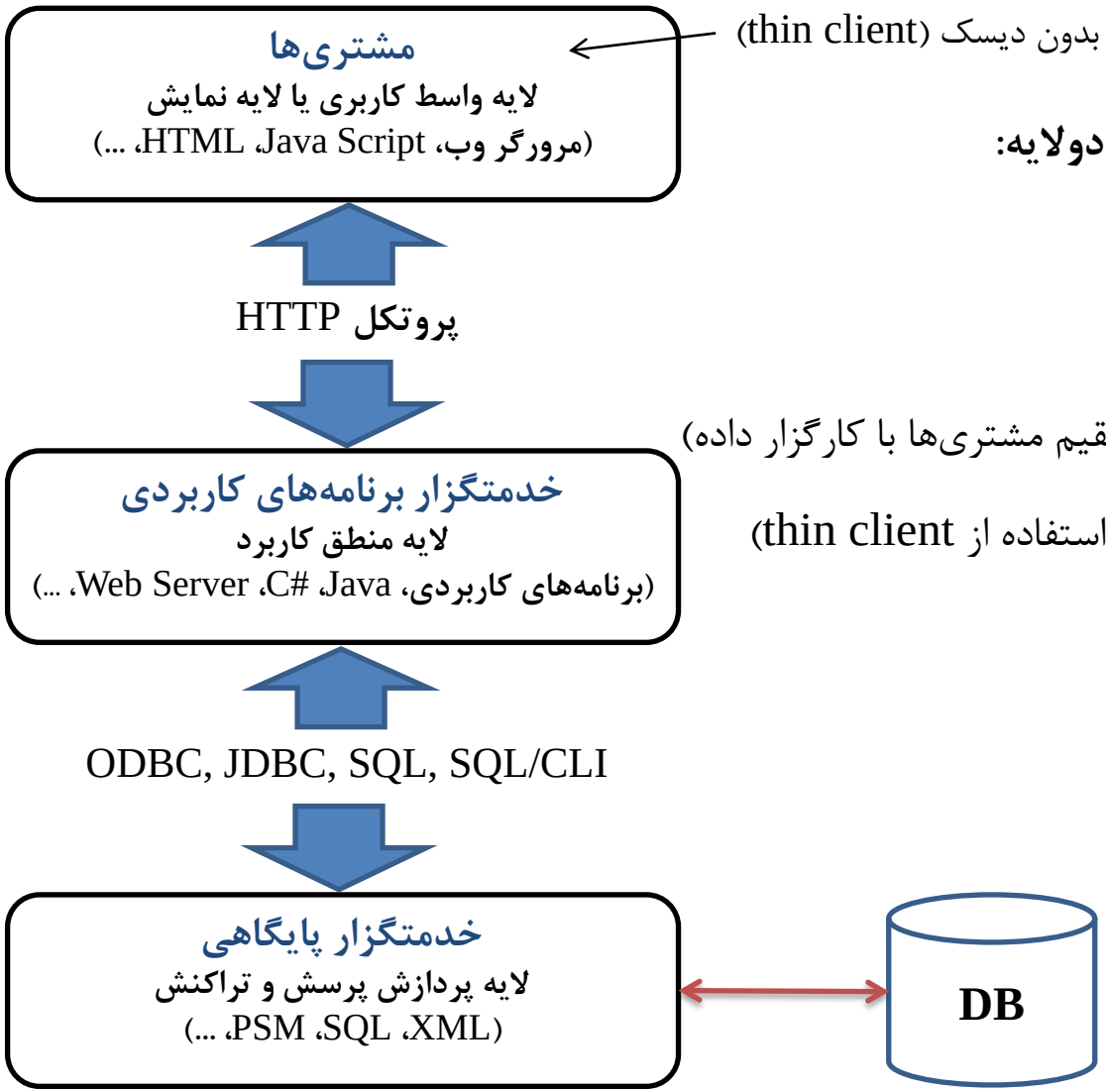
- **انواع معماری مشتری- خدمتگزار**
  - معماری تک مشتری- تک خدمتگزار
  - معماری چند مشتری- تک خدمتگزار
  - معماری تک مشتری- چند خدمتگزار
  - معماری چند مشتری- چند خدمتگزار

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 مشتری– خدمتگزار سه لایه

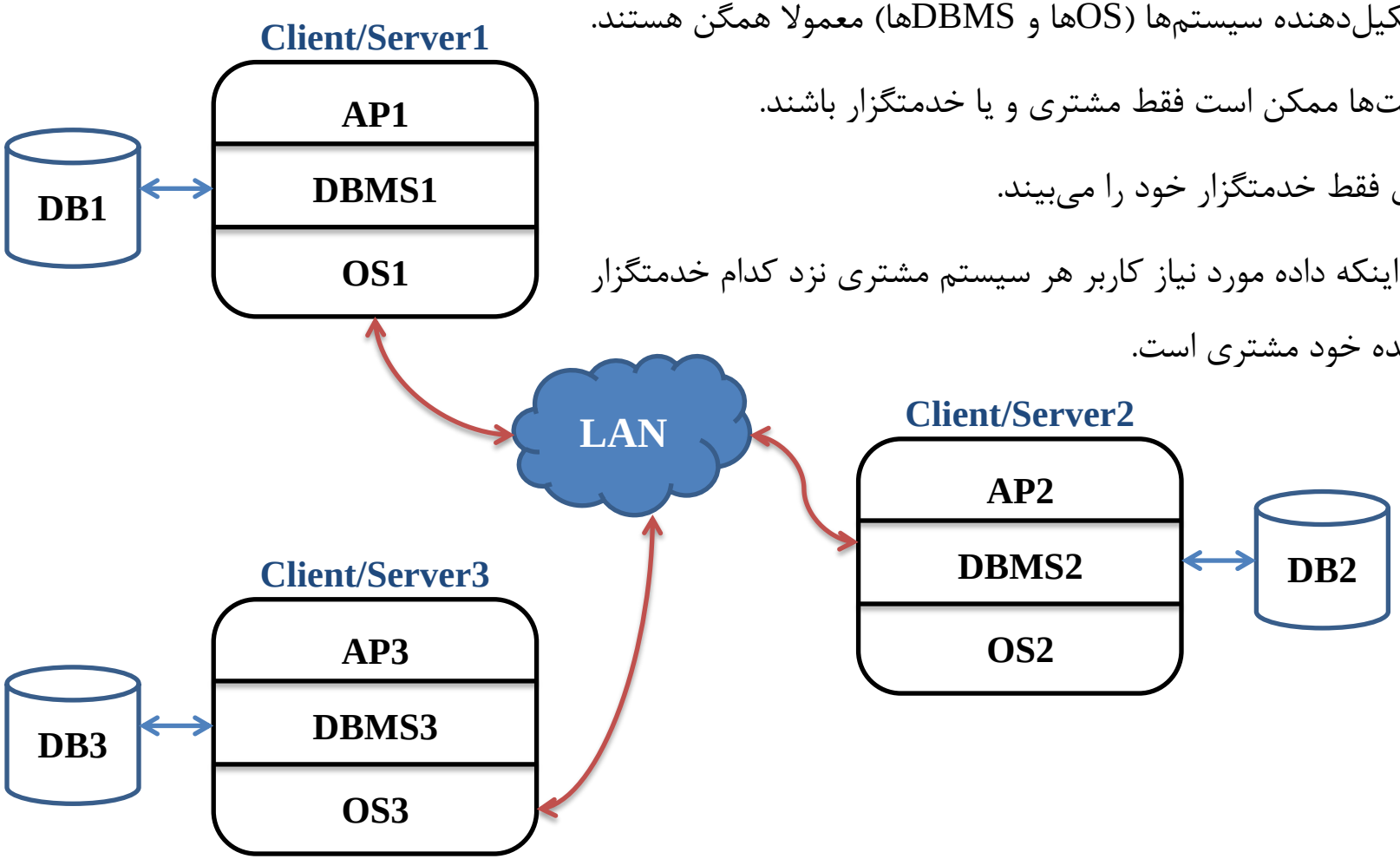


ماشین‌های ساده، ارزان و حتی بدون دیسک (thin client)

- **برخی مزایای معماری سه لایه نسبت به دولایه:**
  - گسترش‌پذیری بهتر
  - کارایی بالاتر
  - امنیت داده‌ای بیشتر (عدم ارتباط مستقیم مشتری‌ها با کارگزار داده)
  - قابلیت کاهش هزینه سخت افزاری (با استفاده از thin client)

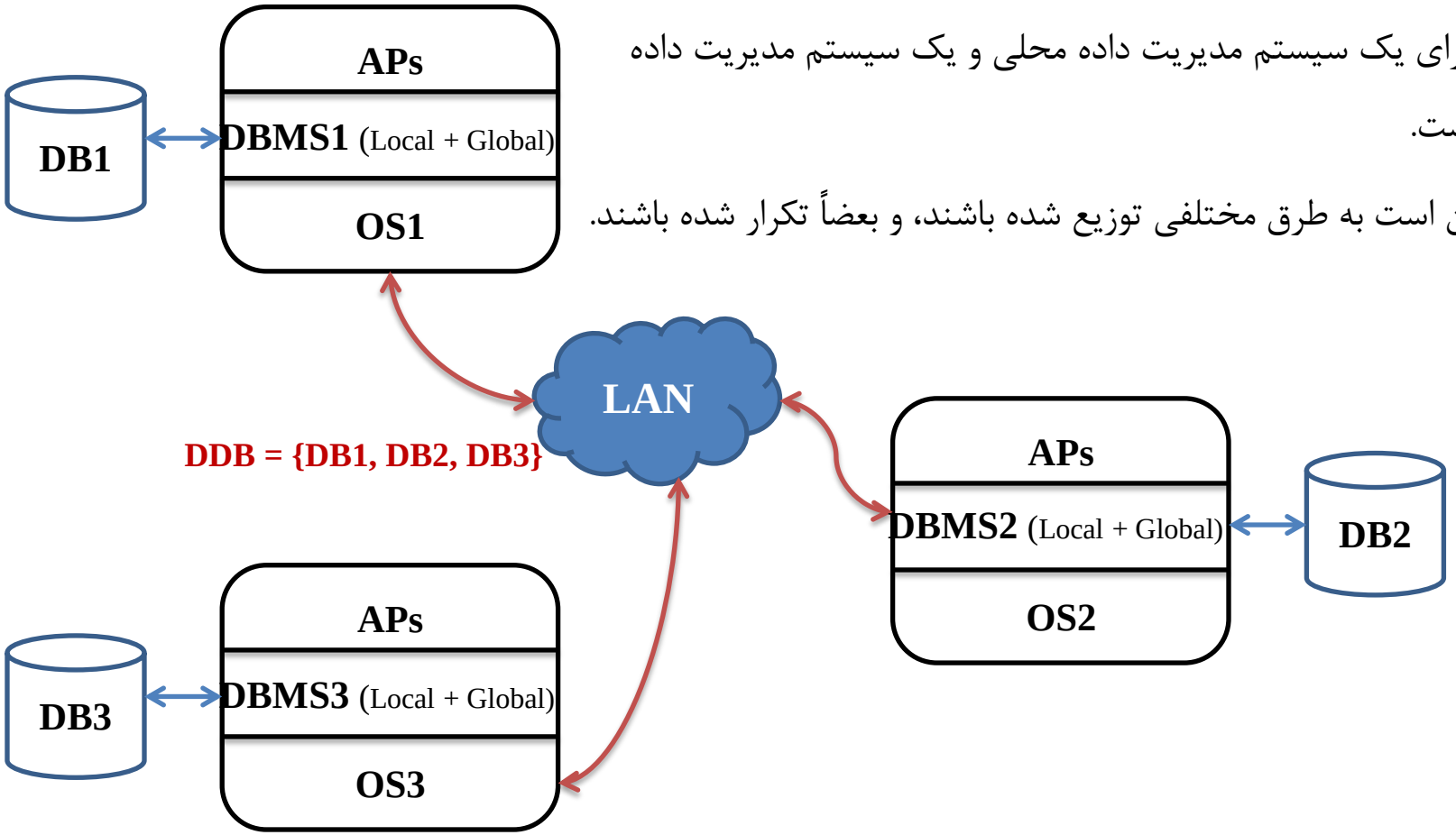
معماری چند مشتری–چند خدمتگزار

- سیستم‌های پایگاهی همزمان یا ناهمزمان ایجاد می‌شوند.
- اجزای تشکیل‌دهنده سیستم‌ها (OSها و DBMSها) معمولا همگن هستند.
- برخی سایت‌ها ممکن است فقط مشتری و یا خدمتگزار باشند.
- هر مشتری فقط خدمتگزار خود را می‌بیند.
- مسئولیت اینکه داده مورد نیاز کاربر هر سیستم مشتری نزد کدام خدمتگزار است برعهده خود مشتری است.



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.
- داده‌ها ممکن است به طرق مختلفی توزیع شده باشند، و بعضاً تکرار شده باشند.



پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش دوم :
مدلسازی معنایی داده‌ها

مرتضی امینی

نیمسال دوم ۹۲-۹۳

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

مدلسازی معنایی داده‌ها (Semantic Data Modeling)

- داده های ذخیره شدنی در DB
  - کاربری
  - سیستمی

ابتدا باید مدل شوند.

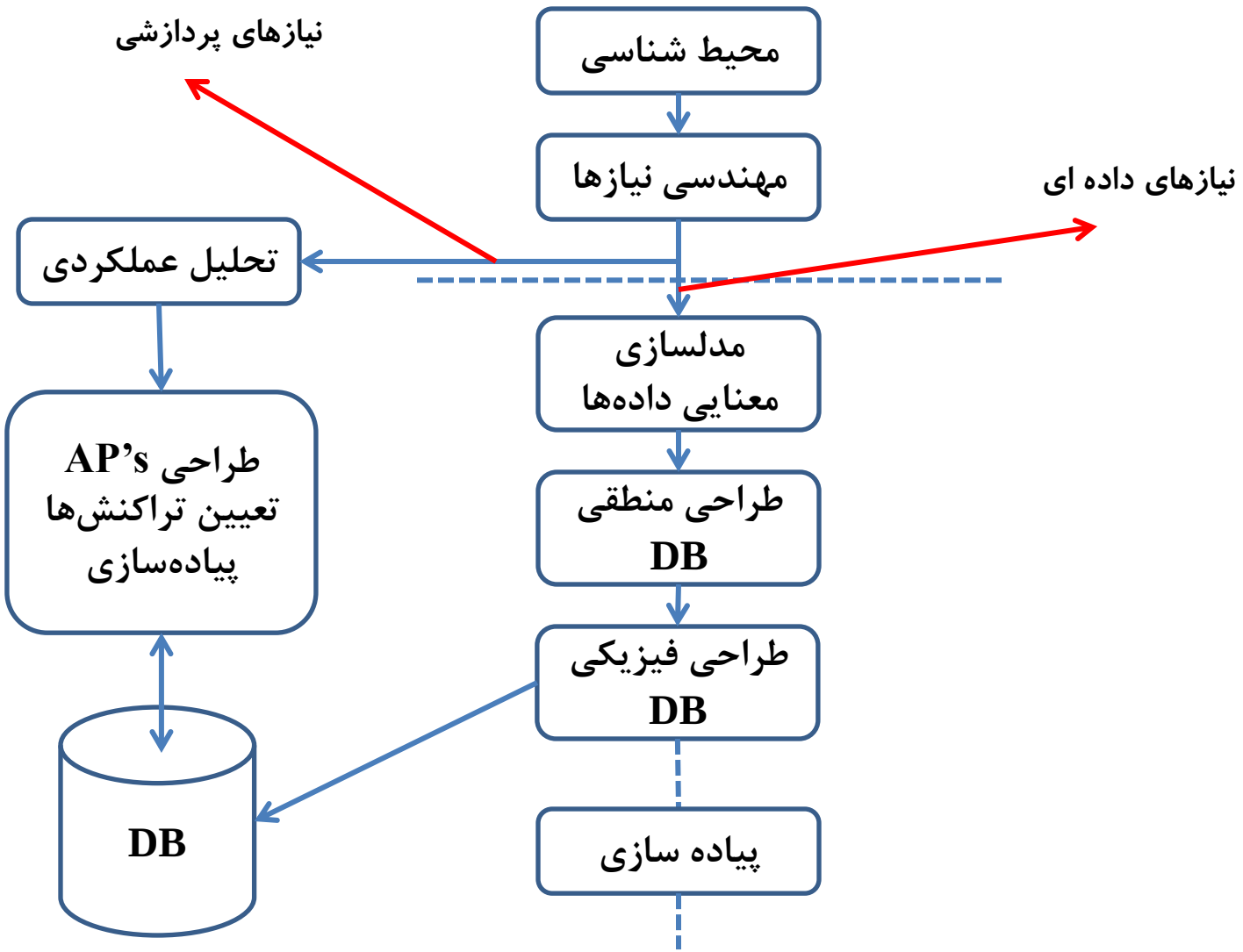
داده‌های کاربری

- موسومند به داده های عملیاتی
- پایا هستند: بعد از اجرای برنامه کاربر کماکان در سیستم ماندگارند[حسب تعریف]
- لزوماً همان داده های I/O نیستند. هر داده موجود در پایگاه داده لزوما داده ورودی نیست و هر داده خروجی از پایگاه لزوما در پایگاه ذخیره شده نیست (مانند داده‌های محاسبه شده از داده‌های موجود- میانگین نمرات)

داده‌های سیستمی

- سیستم تولید می‌کند برای انجام وظایفش

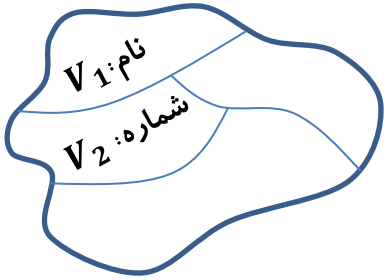
مراحل تولید سیستم اطلاعاتی



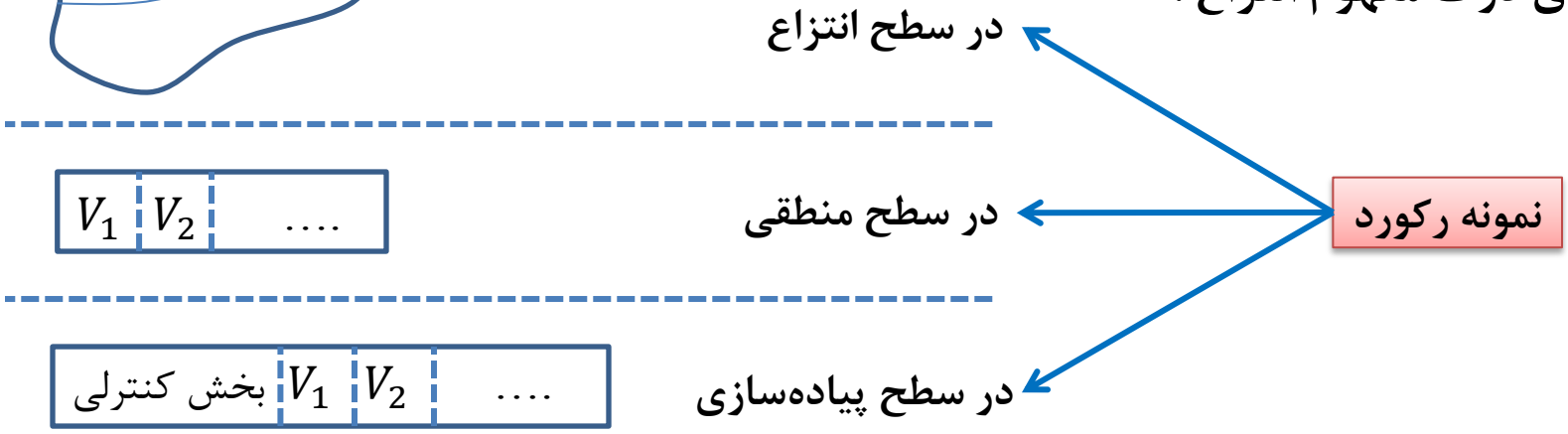
مدلسازی معنایی داده‌ها

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 :



مدلسازی

- **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 داده ها نیست بلکه برای مدلسازی و طراحی سیستم های نرم‌افزاری است. لذا با آن می‌توان پایگاه داده را مدل کرد.

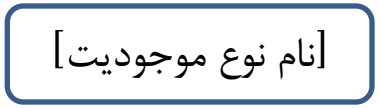
روش ER مبنایی

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

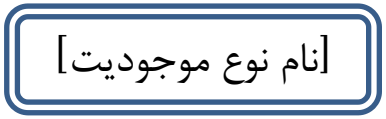
- **نمودار ER:**
  - نموداری است که سه مفهوم اساسی نوع موجودیت، صفت و نوع ارتباط در آن نمایش داده می‌شوند. در واقع این نمودار امکانی است برای نمایش مدلسازی و اولین طرح پایگاه داده‌ها در بالاترین سطح انتزاع.
  - برای رسم این نمودار به نمادهایی نیاز داریم. در این درس از نمادهای چِن استفاده می‌شود.

نمادهای نمودار ER مبنایی

- نوع موجودیت



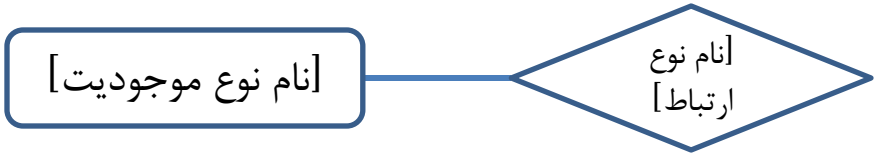
- نوع موجودیت ضعیف



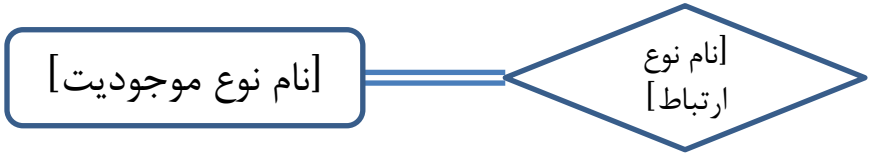
- نوع ارتباط

- نوع ارتباط موجودیت ضعیف با قوی

- مشارکت نوع موجودیت در نوع ارتباط



- مشارکت الزامی



نمادهای نمودار ER مبنایی (ادامه)

- صفت

- صفت شناسه اول

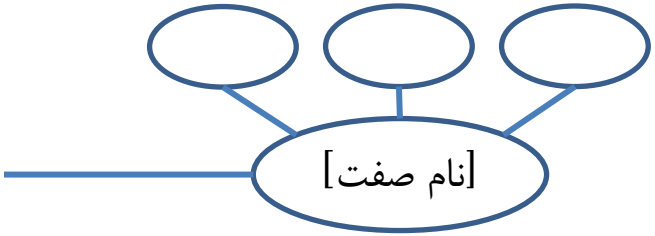
- صفت شناسه دوم (در صورت وجود)

- صفت شناسه مرکب (مثلا دو صفتی)

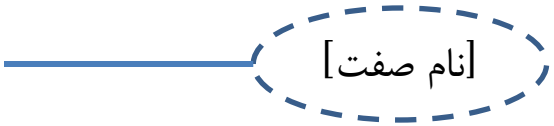
- صفت چندمقداری

نمادهای نمودار ER مبنایی (ادامه)

- صفت مرکب



- صفت مشتق (مجازی یا محاسبه‌شدنی)



- چندی ارتباط

ER مبنایی – نوع موجودیت

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

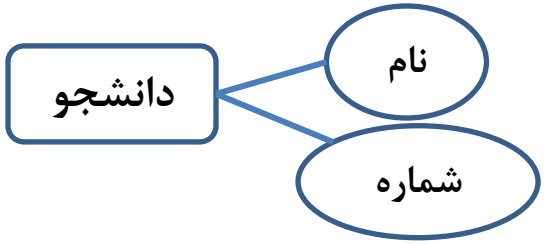
- نوع موجودیت‌ها
  - ۱- دانشجو
  - ۲- درس
  - ۳- استاد
  - ۴- کارمند
  - ...

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

کنکاوی

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).
  - نمونه‌هایی دارد (حداقل یک نمونه).
  - ارتباط(هایی) با نوع موجودیت(های) دیگر دارد.
  - نوع موجودیت دو گونه است.
    - قوی (مستقل) Strong
    - ضعیف (وابسته) Weak

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

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

کنجاوی؟ آیا نوع موجودیت ایزوله داریم؟

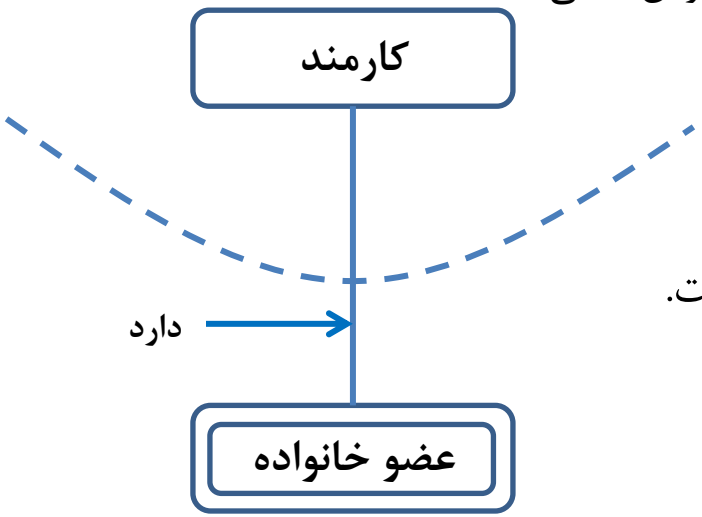
ER مبنایی – نوع موجودیت (ادامه)

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.

ER مبنایی – صفت

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)

ER مبنایی – صفت (ادامه)

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

معنای
نوع
طیف مقادیر
} صفت را مشخص می‌کند. و نه لزوماً نام صفت را.

۱- محدودیت میدانی

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** Entity Identifier (EID)
    - ۱- **یکتایی** مقدار Uniqueness
    - ۲- مقادیرش همیشه **معلوم** باشد (هیچمقدارناپذیر)

    ← ویژگی ذاتی

    - ۳- طول کد نمایش حتی‌الامکان کوتاه
    - ۴- مقادیرش حتی‌الامکان تغییر ناپذیر

    ← بهتر است که داشته باشد.
  - ۲- **ناشناسه**

- 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 می تواند ساختار سلسله مراتبی هم داشته باشد) مثال: آدرس (ترکیبی از استان، شهر، خیابان، ...)

ER مبنایی – صفت (ادامه)

- **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، حدّاکثر 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؟

ER مبنایی – صفت (ادامه)

- 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.

ER مبنایی – نوع ارتباط

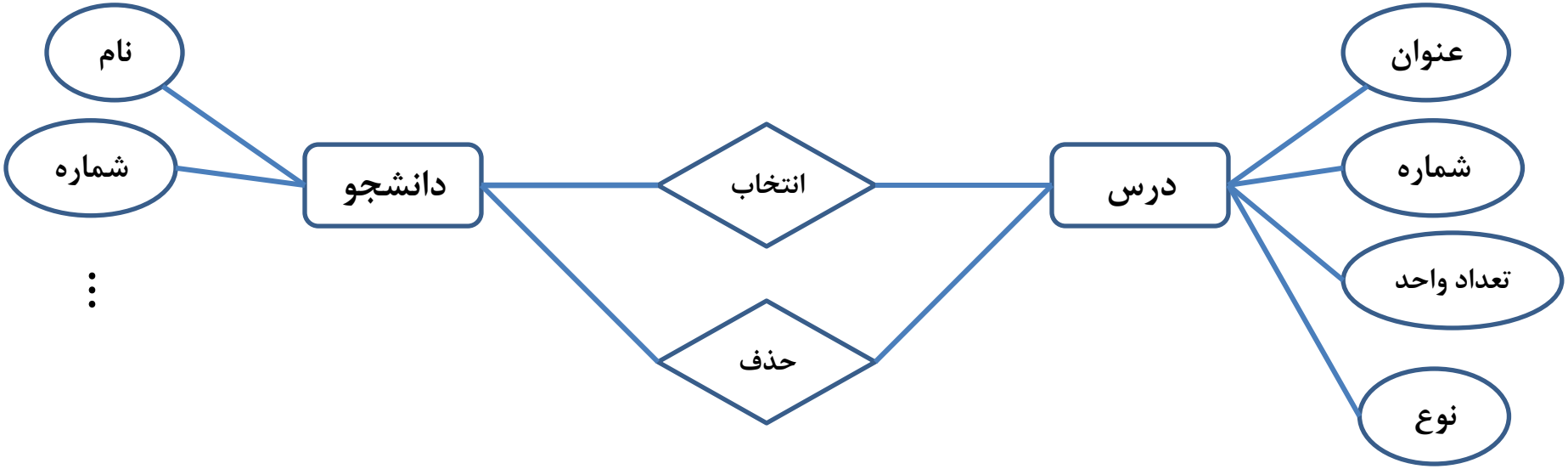
نوع ارتباط Relationship Type:

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

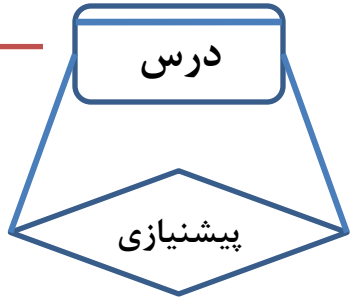
ارتباط نوع موجودیت‌های دانشجو و درس

- 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)

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

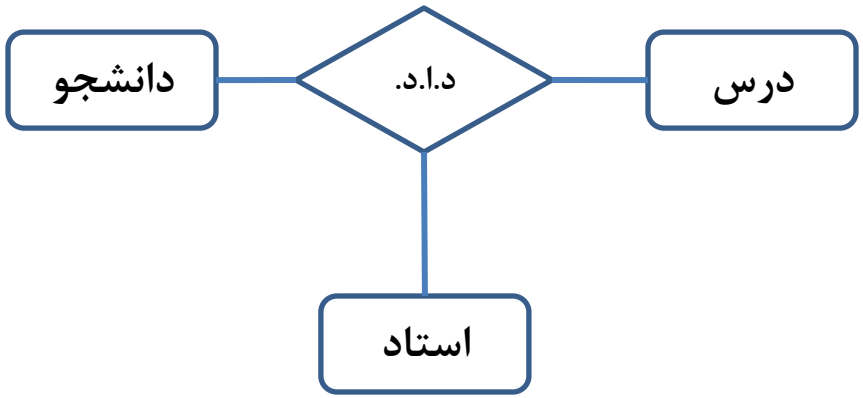
ER مبنایی – نوع ارتباط (ادامه)

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

ER مبنایی – نوع ارتباط (ادامه)

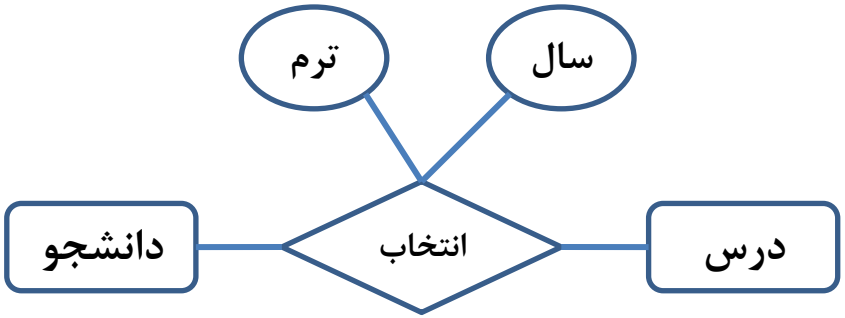
- مشارکت نوع موجودیت E در نوع ارتباط R
- **الزامی** (کامل): هر نمونه از موجودیت E لزوماً در یک نمونه ارتباط R مشارکت دارد.
- **غیر الزامی** (ناقص): حداقل یک نمونه موجودیت E وجود دارد که در هیچ نمونه ارتباط R مشارکت ندارد.
- الزامی بودن مشارکت از محدودیت‌های معنایی محیط، ناظر به نوع ارتباط است.

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.
- در مواردی که به ظاهر نتوانیم با نمونه موجودیت‌های یک ارتباط آن را مشخص نماییم، می‌توانیم از صفت چندمقداری (برای رعایت نکته فوق) استفاده کنیم.

ER مبنایی – نوع ارتباط (ادامه)

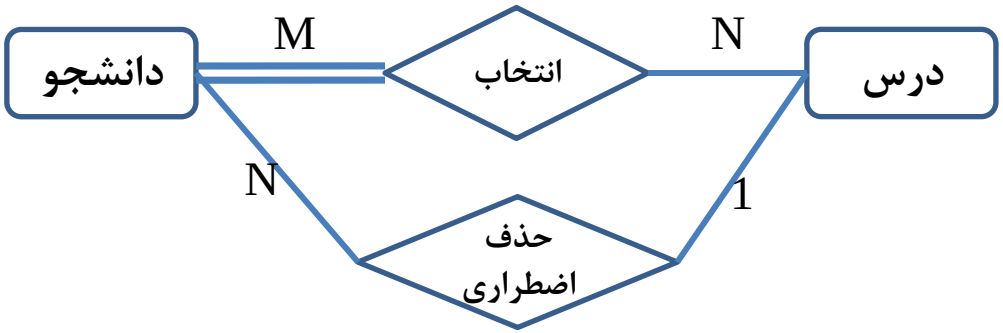
چندی ارتباط Multiplicity یا Cardinality Ratio:

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>1) و در صورت مشارکت غیرالزامی، (n=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>1) ارتباط دارد و برعکس.
- **نکته:** چندی نوع ارتباط چندگانی (n>2) عبارت است از تعداد نمونه‌های یک نوع موجودیت شرکت کننده در آن نوع ار تباط، وقتی که تعداد نمونه‌های n-1 نوع موجودیت دیگر شرکت کننده در نوع ارتباط را ثابت فرض کنیم.

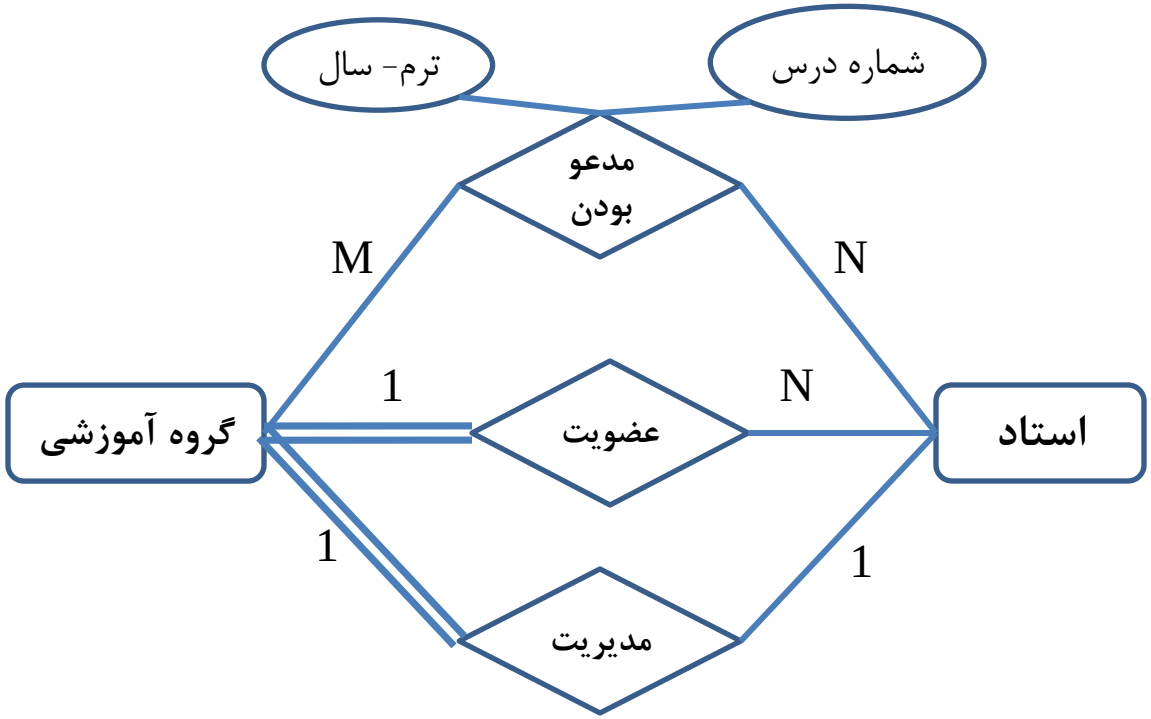
ER مبنایی – نوع ارتباط (ادامه)

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.

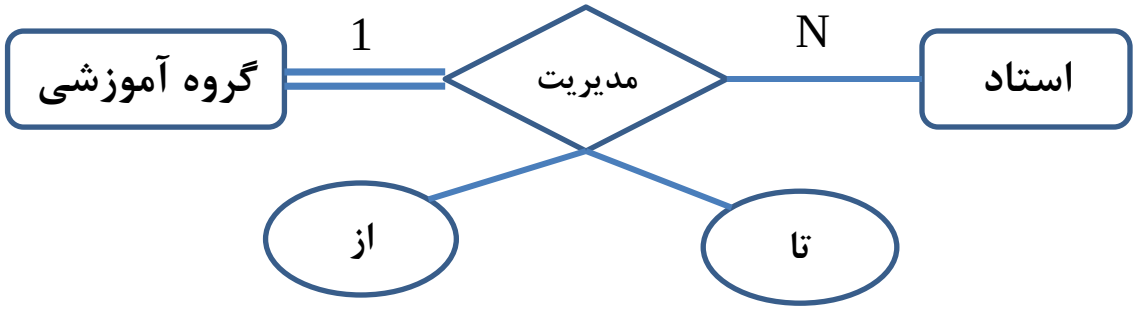


ER مبنایی – نوع ارتباط (ادامه)

مثال مثالی دیگر از چندی ارتباط

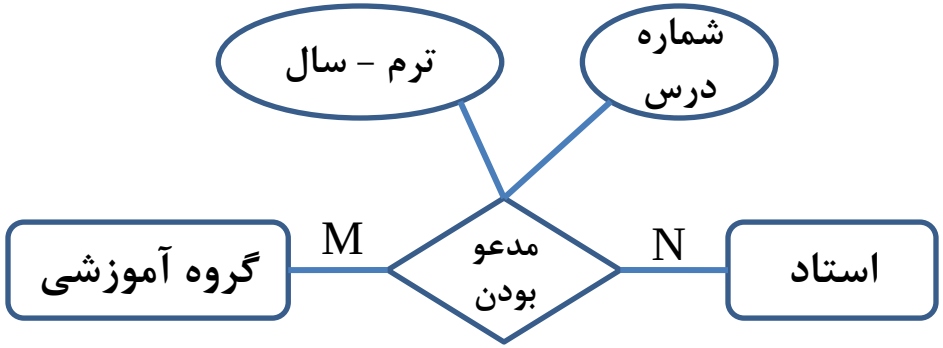


- **تذکر:** اگر به ارتباط صفت هایی از جنس زمان بدهیم، چندی ارتباط می‌تواند بسته به قواعد معنایی محیط تغییر کند.

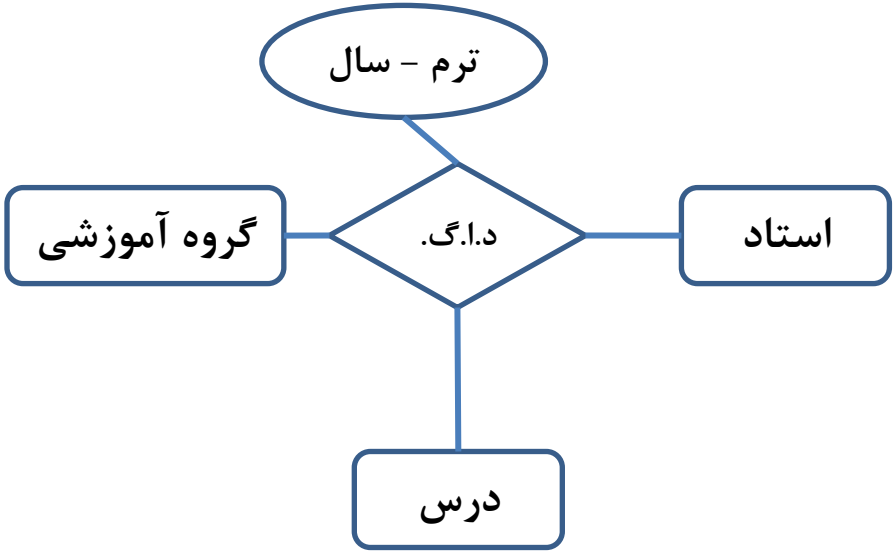


ER مبنایی – نوع ارتباط (ادامه)

گونه‌های دیگر مدل کردن نوع ارتباط مدعو بودن چیست؟

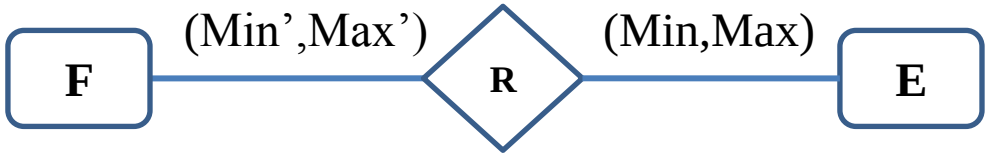


- با استفاده از نوع ارتباط سه گانی:



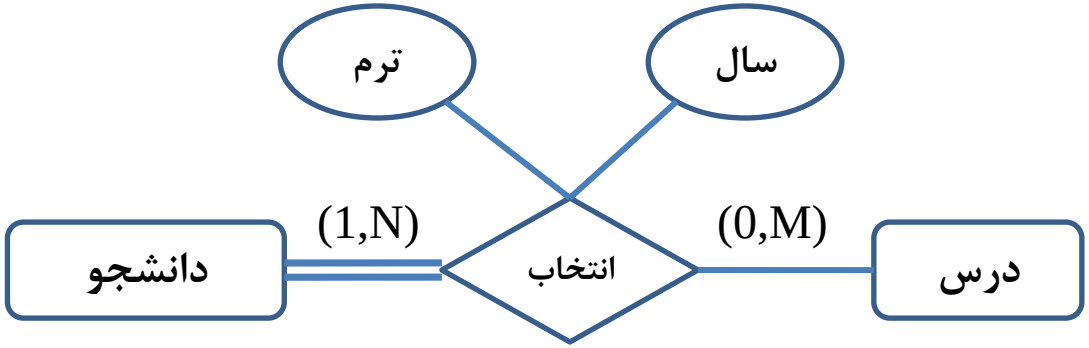
ER مبنایی – نوع ارتباط (ادامه)

- **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.

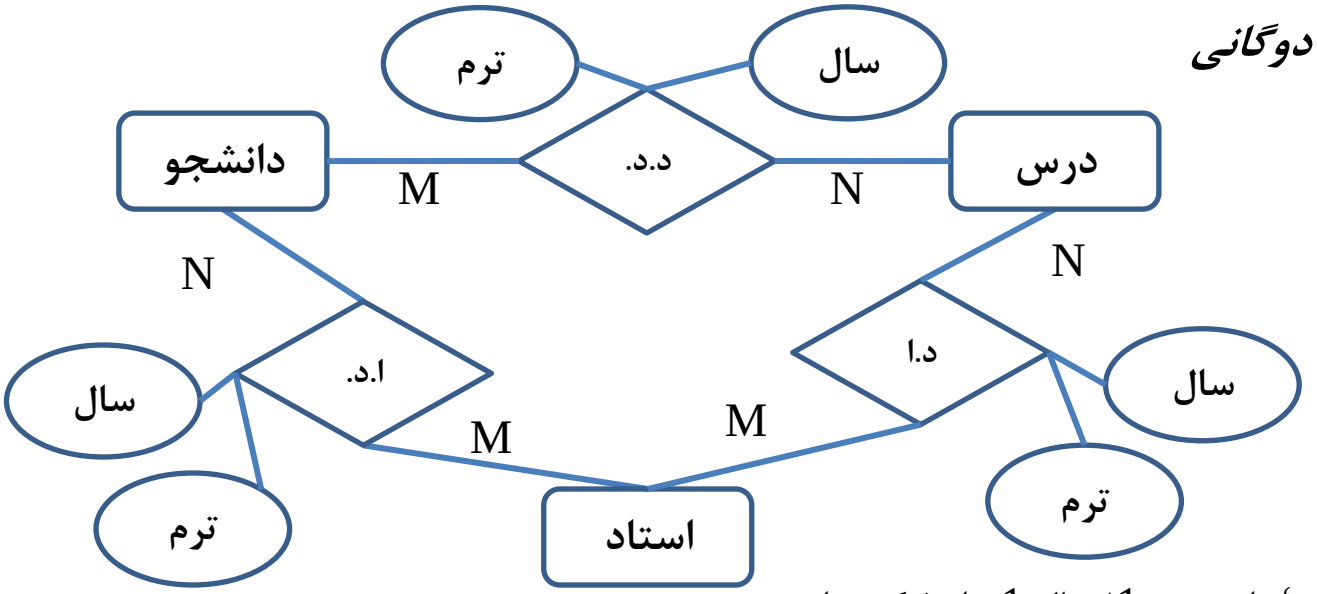
مثال: رابطه انتخاب درس توسط دانشجو



کنجکاوی؟ مزایای این روش نمایش چندی؟

ER مبنایی – نوع ارتباط (ادامه)

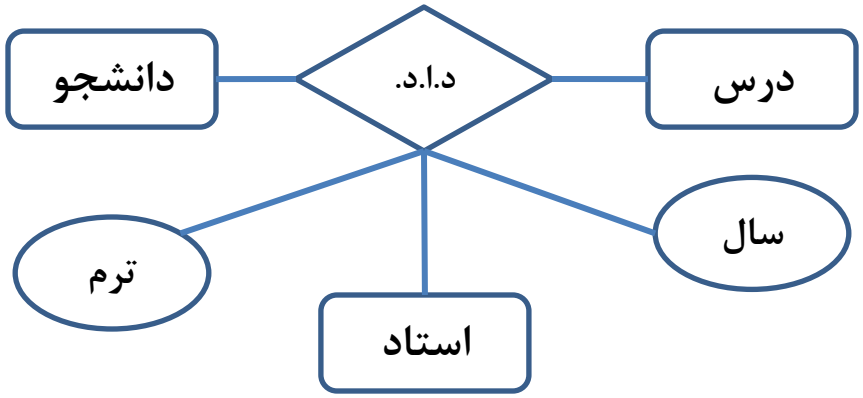
- نکته مهم در مورد ارتباط بین سه نوع موجودیت:
  - *مدل یک: سه ارتباط دوگانی*



- 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.

ER مبنایی – نوع ارتباط (ادامه)

- ***مدل دوم: ارتباط سه گانی***

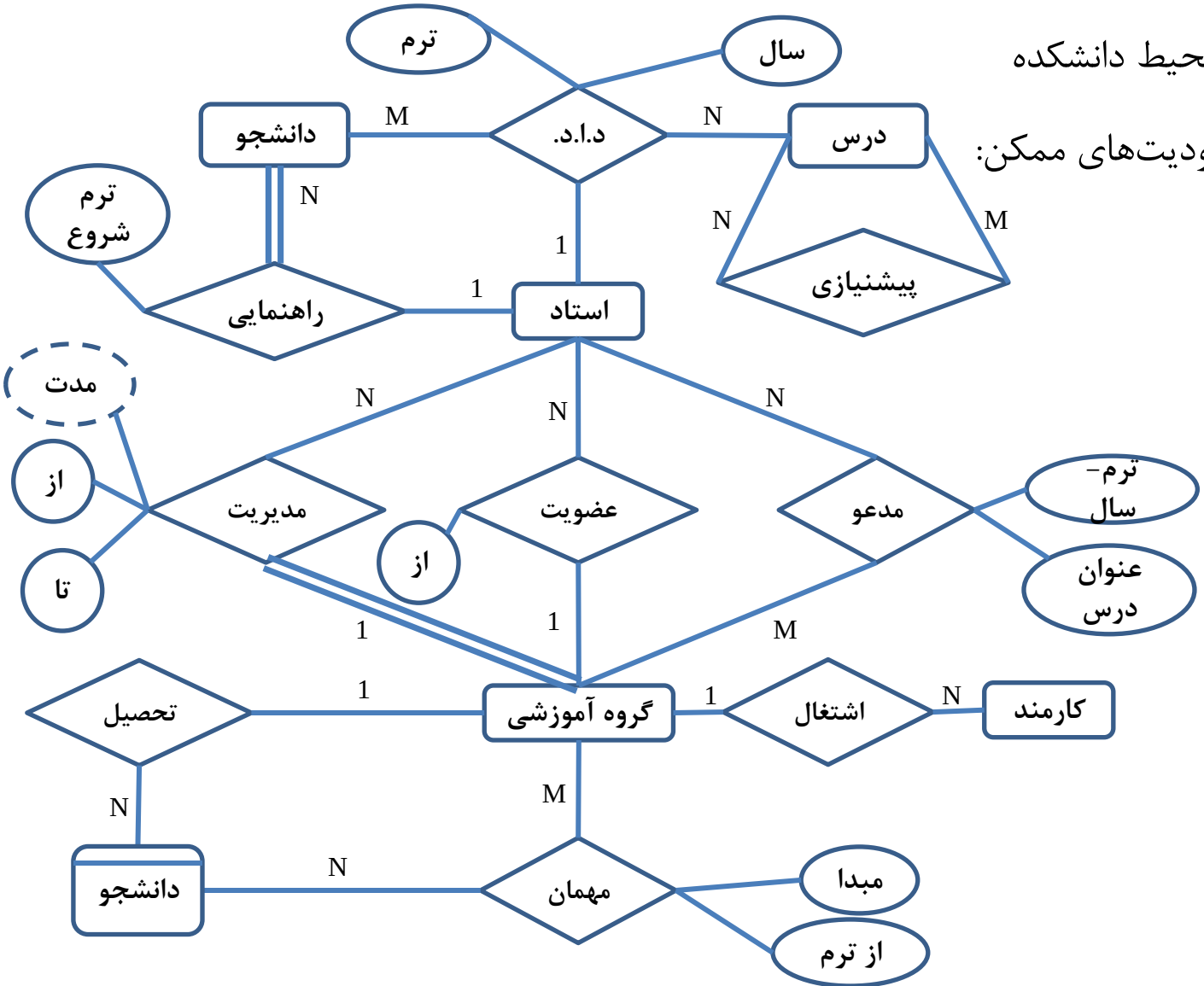


- در حالت سه ارتباط دوگانی اگر از فقره اطلاع‌های دوگانی، فقره اطلاع سه گانی را استنتاج کنیم در شرایطی که از لحاظ معنایی این استنتاج درست نباشد می گوییم دچار **دام پیوندی حلقه‌ای** شده‌ایم.

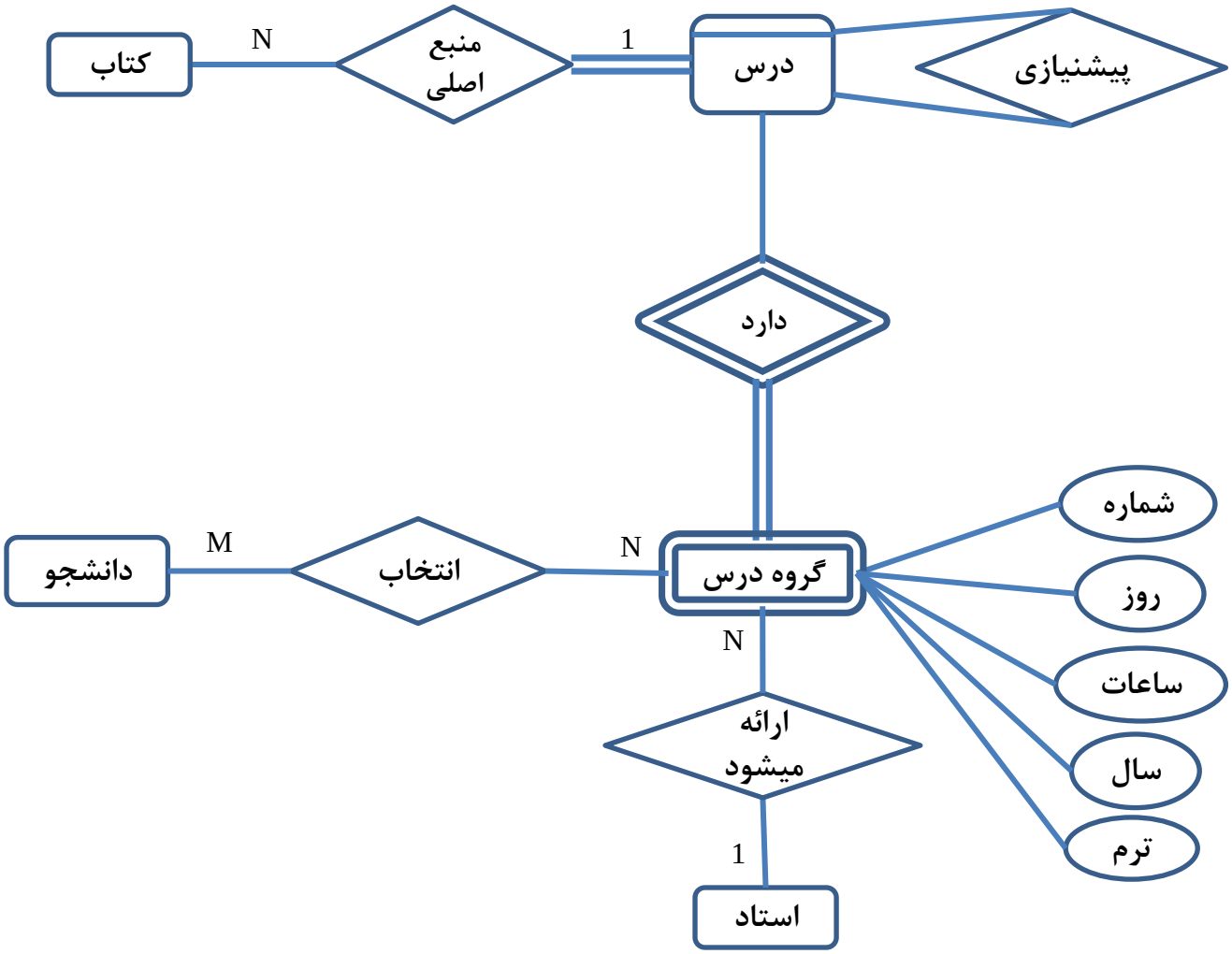
کنجکاوی؟ انواع دیگر دام چیست؟ (دام چندشاخه (چتری)، دام گسل (شکافت)، ...)

مثال: محیط دانشکده

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

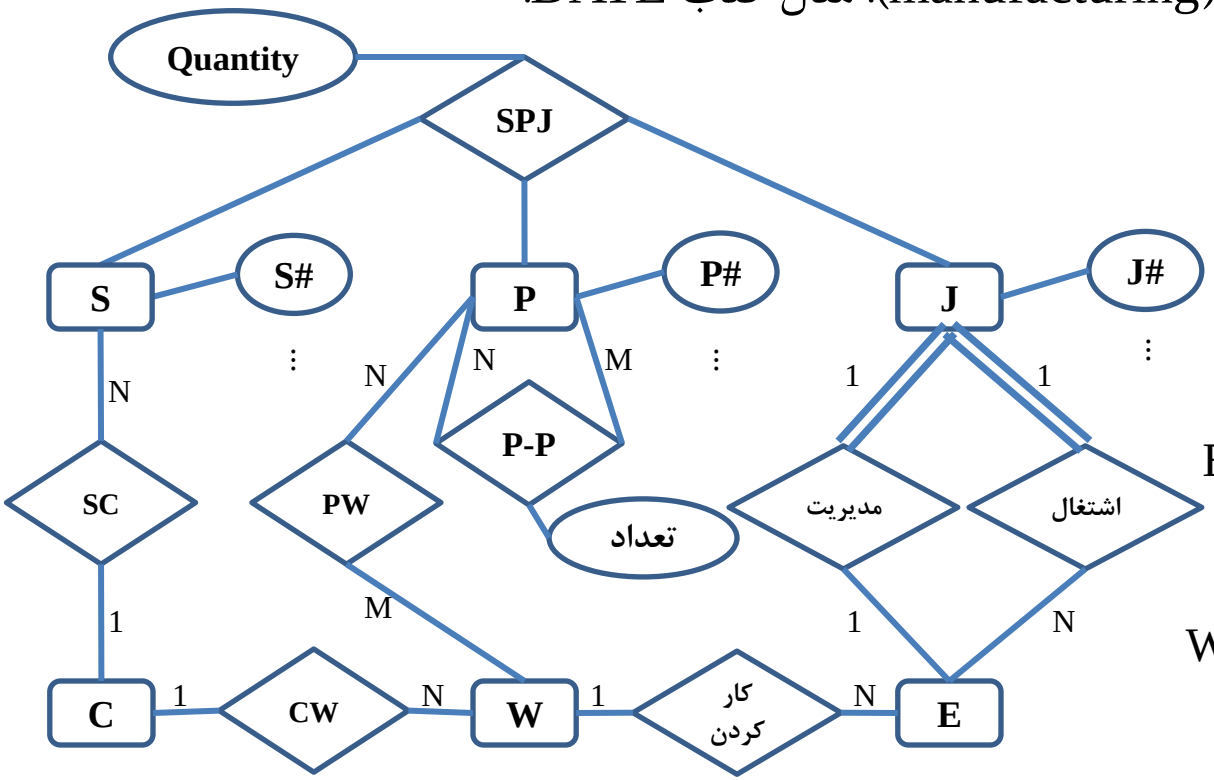


مثال: محیط دانشکده (ادامه)



مثال: محیط تولید

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

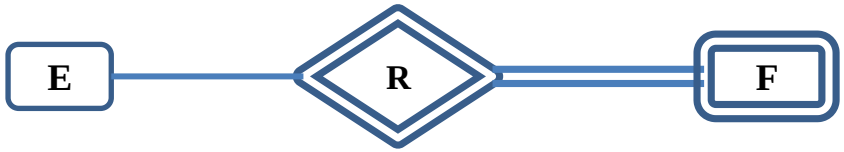


- نوع موجودیت ها:
  - S: Supplier
  - P: Part
  - J: Project
  - E: Employee
  - C: City
  - W: Warehouse

گسترش داده شود.

بحث تکمیلی : نوع موجودیت ضعیف

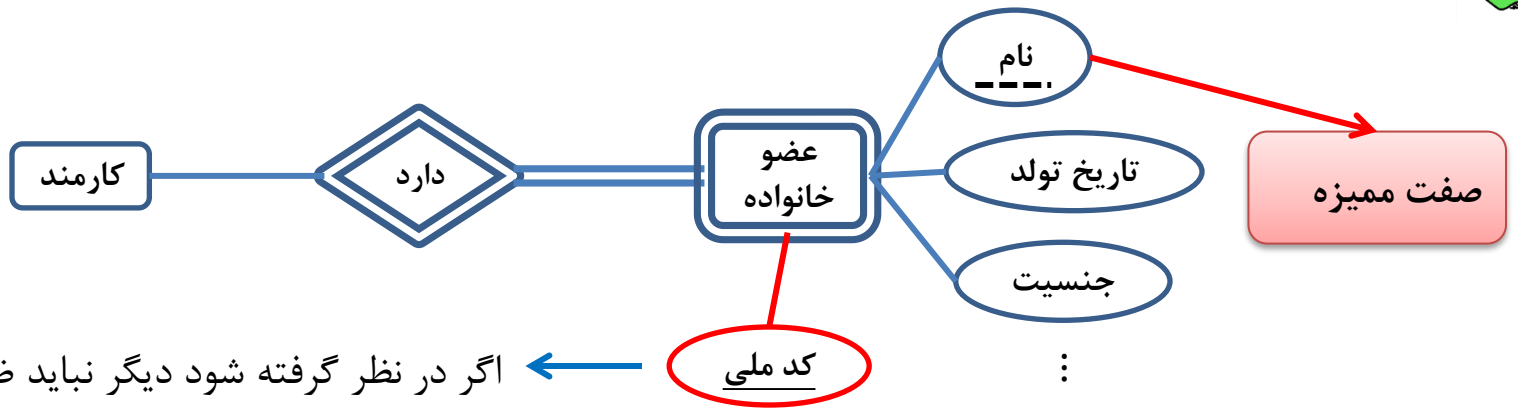
- **نوع موجودیت ضعیف:**
- نوع موجودیت F را ضعیف نوع موجودیت E گوییم هرگاه F با E «وابستگی وجودی» داشته باشد. (یعنی اگر E در مدلسازی مطرح نشود، F هم مطرح نباشد). علاوه‌براین نوع موجودیت ضعیف از خود شناسه ندارد.
- **طرز نمایش:**



- **تاکید:** قوی و ضعیف بودن نسبی است.
- نوع ضعیف از خود شناسه ندارد. بلکه از خود حداقل یک **صفت ممیزه-جداساز** (Discriminator) دارد.
- **صفت ممیزه (کلید جزئی):**
  - صفتی که یکتایی مقدار دارد اما نه در تمام نمونه های نوع ضعیف بلکه در بین مجموعه تمام نوع ضعیف‌های وابسته به یک نمونه از نوع موجودیت قوی (به صورت نسبی یکتاست).
  - در عمل اگر یک نوع موجودیت وابستگی وجودی به نوع موجودیت دیگر داشته باشد و از خود شناسه داشته باشد دیگر ضعیف دیده نمی‌شود.

بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

مثال عضو خانواده به عنوان یک موجودیت ضعیف



: اگر در نظر گرفته شود دیگر نباید ضعیف دیده شود.

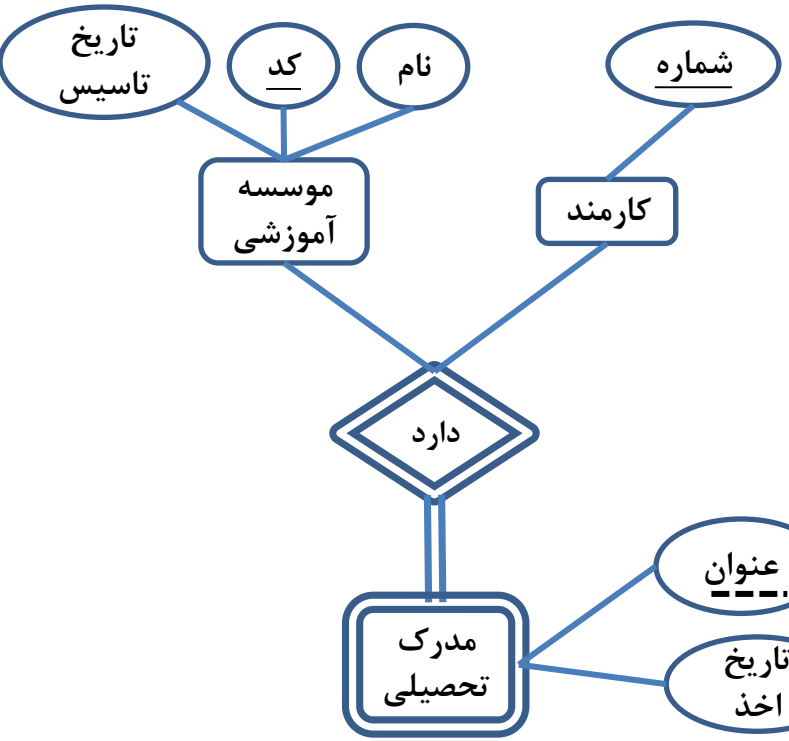
شماره کارمند	نام
۱۰۰	گلی سلی قلی
۲۰۰	ناجی تاجی سلی

بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

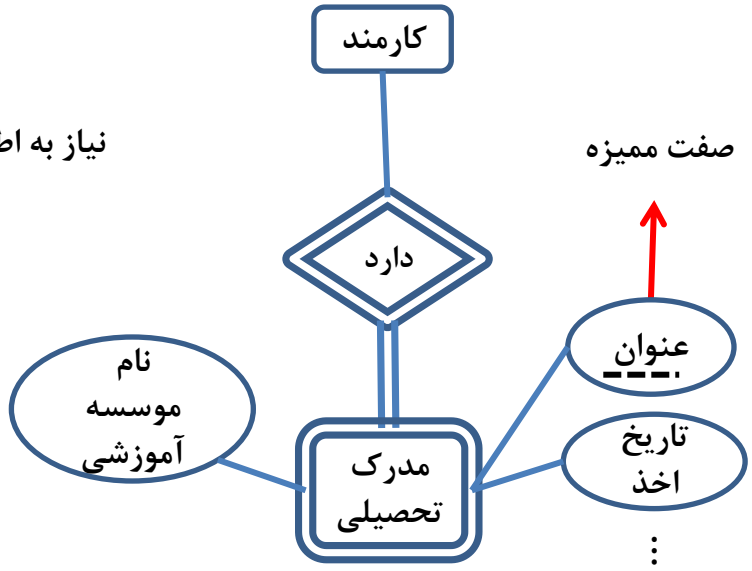
- به ارتباط قوی-ضعیف، **ارتباط شناسا** (Identifying Relation) گویند.
- مشارکت نوع ضعیف در ارتباط شناسا الزامی است.
- چندی ارتباط معمولا 1:N (در حالت خاص 1:1 تمرین: مثال قید شود).

بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

- درجه ارتباط شناسا معمولا ۲ و گاه بیشتر است.



نیاز به اطلاعات بیشتر از موسسه آموزشی

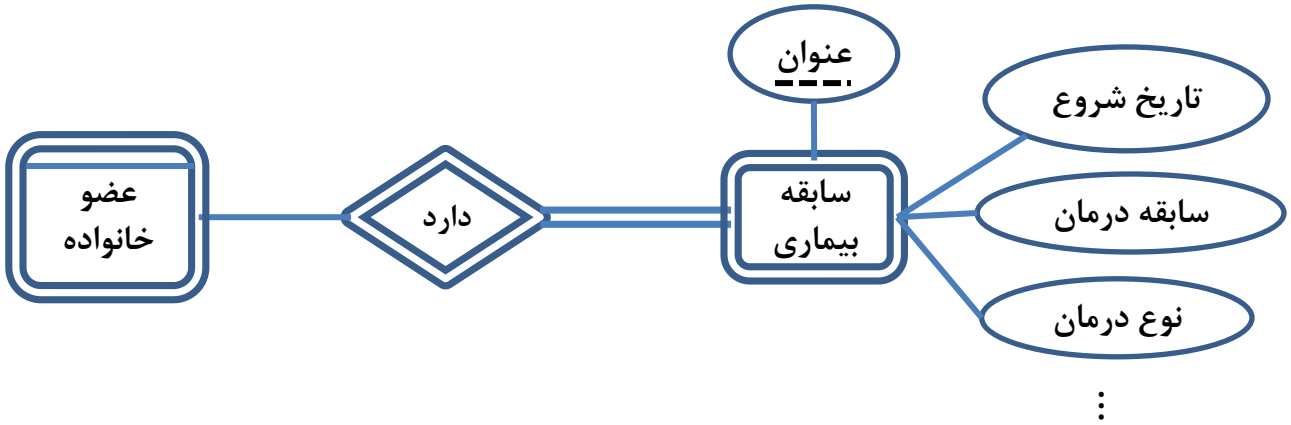


کنکاوی در این مدل آیا صفت ممیزه نسبت به دو قوی لزوما واحد است؟

کنکاوی آیا این محیط را می‌توان به گونه‌ای دیگر مدل کرد؟

بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

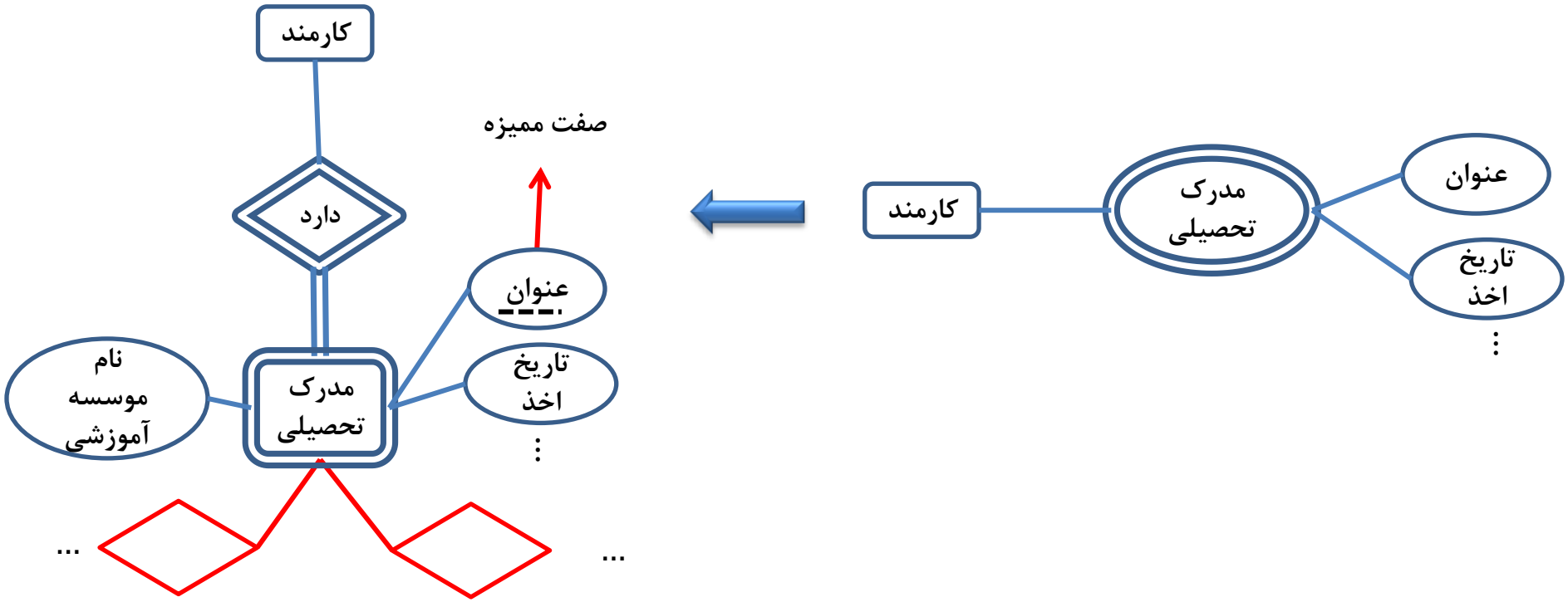
- نوع موجودیت ضعیف می‌تواند خود قوی برای نوع موجودیت ضعیف دیگر باشد.



اگر بخواهیم برای کارمند سابقه بیماری‌اش را نگه داریم چه کارکنیم؟

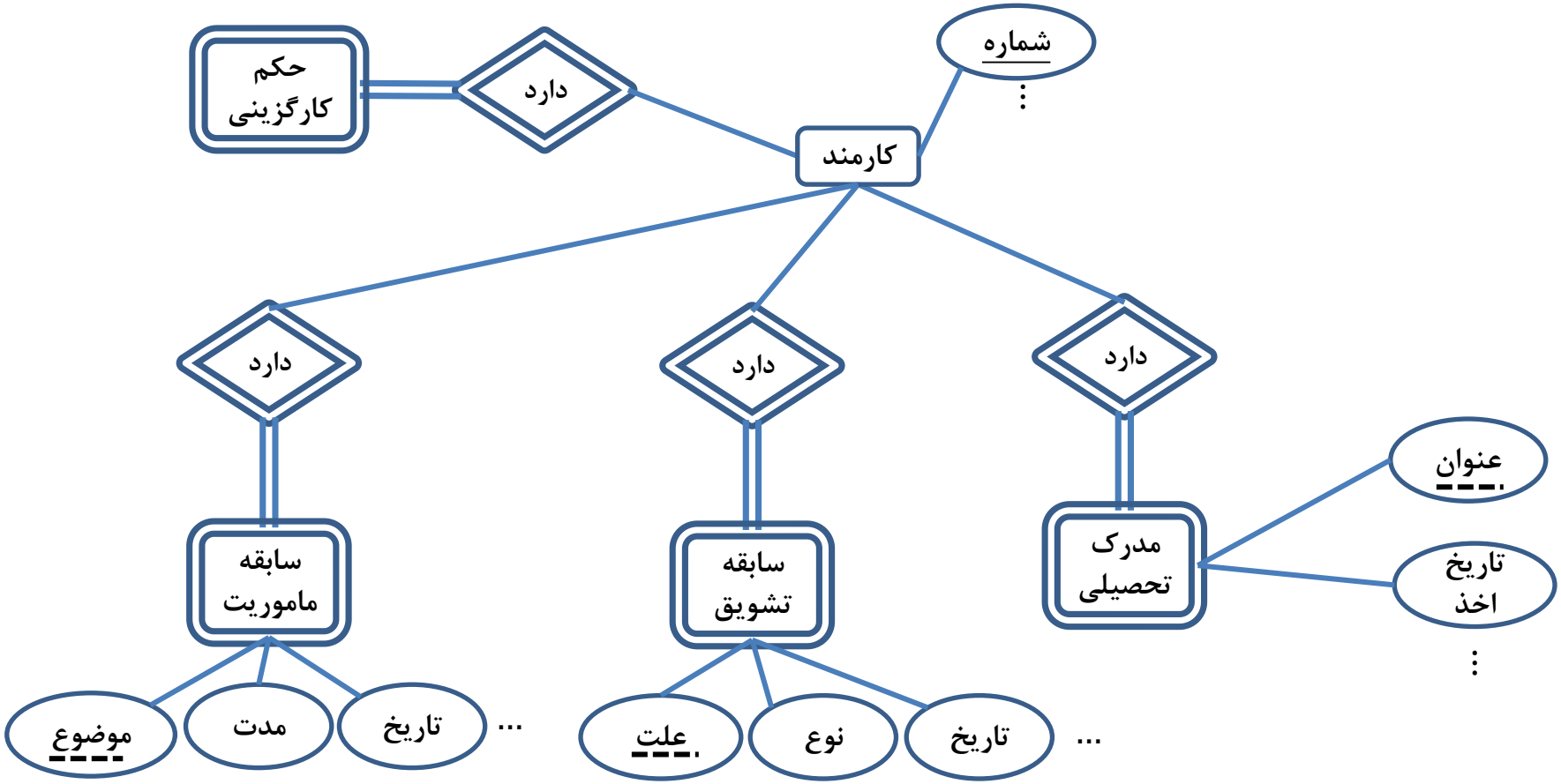
بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

- صفت چند مقداری (به خصوص مرکب) را همیشه می‌توان با مفهوم نوع موجودیت ضعیف مدل کرد (نمایش داد) اما عکس این تکنیک توصیه نمی‌شود.
- **دلیل:** انعطاف پذیری مدل را از نظر گسترش پذیری کاهش می‌دهد، زیرا نوع ضعیف می‌تواند خود نوع ارتباط‌هایی داشته باشد با دیگر نوع موجودیت‌ها، اما وجود ارتباط با صفت معنا ندارد.



بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

- مفهوم نوع موجودیت ضعیف به ویژه برای مدل کردن پدیده‌های تکرار شونده (در زمان) و وابسته به مفهوم دیگر استفاده می‌شود.

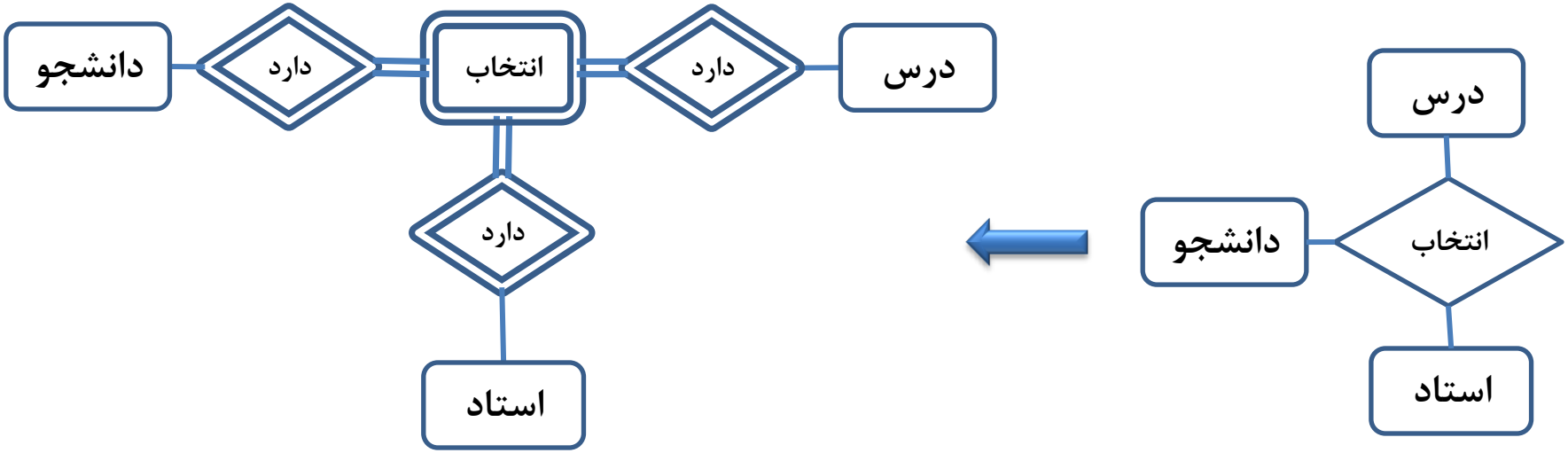


بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

تبدیل ارتباط سه‌گانی به ارتباطات دوگانی

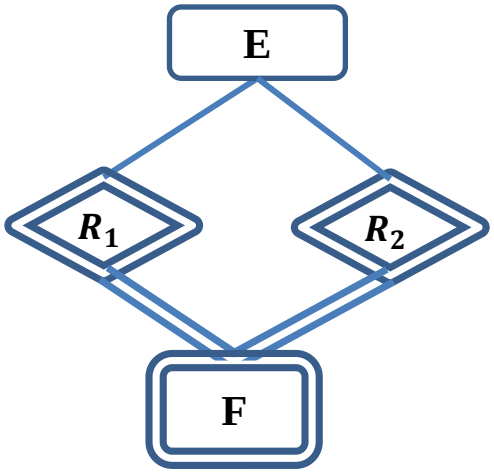
- از مفهوم نوع موجودیت ضعیف می‌توان برای تبدیل یک ارتباط سه‌گانی (یا n-گانی) به ارتباطات دوگانی استفاده کرد.
- اغلب ابزارهای طراحی مبتنی بر روش ER فقط ارتباطات دوگانی را پشتیبانی می‌کنند.

مثال: تبدیل رابطه سه‌گانه انتخاب به سه رابطه دوگانی.



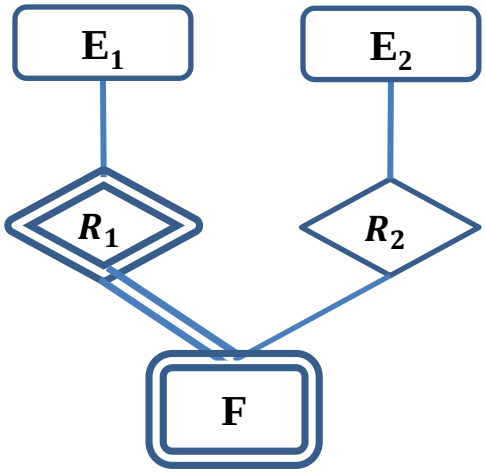
بحث تکمیلی : نوع موجودیت ضعیف (ادامه)

- می‌توان چند ارتباط شناسا بین یک نوع موجودت قوی و یک نوع موجودیت ضعیف داشت.



مثالی از مطلب فوق بیاورید.

- یک نوع موجودیت ضعیف می‌تواند در یک نوع ارتباط دیگر با نوع موجودیت قوی دیگر شرکت داشته باشد.



مثالی از مطلب فوق بیاورید.

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، آنرا به یک نوع ارتباط در نظر می‌گیریم.
  - اگر یک (چند) صفت در چند نوع موجودیت، مشترک باشند، آنرا به عنوان 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، آنرا به صورت نوع موجودیت ضعیف در نظر می‌گیریم.

ER گسترش یافته

- **EER**: Extended ER یا Enhanced ER

- ER مبنایی کمداشت هایی داشت در نمایش بعض نوع ارتباطها (که بعدا در حیطه شی‌گرایی 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:** ارتبباط بین یک نوع موجودیت عام است با نوع موجودیت (های) خاص آن که بر

  زبرنوع (Supertype) — زیرنوع (SubType)

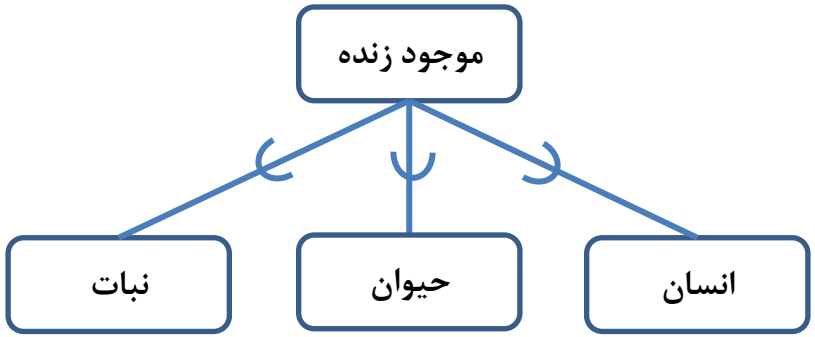
  اساس یک ضابطه مشخص بازشناسی می‌شود.

  صفت معرف — Defining Attribute

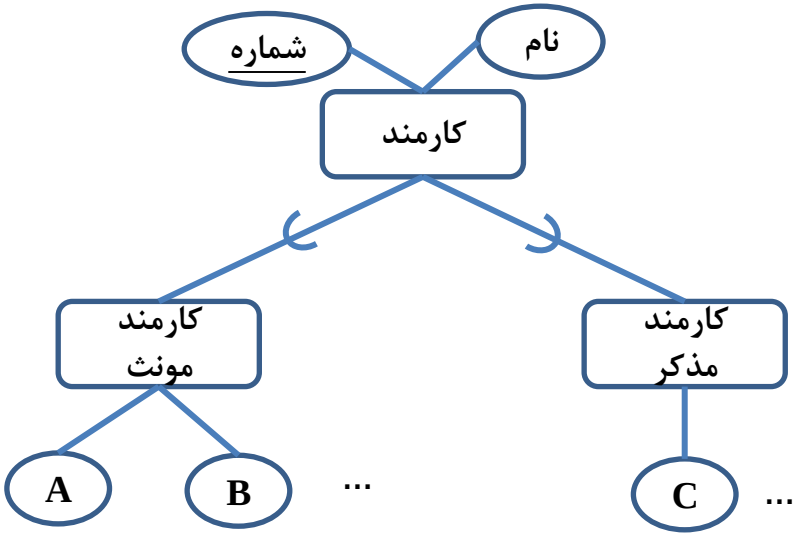
- طرز نوشتن: "F IS-A E"
- وقتی نوع های خاص یک نوع عام را بازشناسی می‌کنیم به آن تکنیک ویژه نمایی-تخصیص یا Specialization گوییم.
- عکس این تکنیک را تعمیم یا Generalization گوییم.

ارتباط "IS A" (ادامه)

مثال انواع موجودات زنده



مثال انواع کارمندان



ارتباط "IS A" (ادامه)

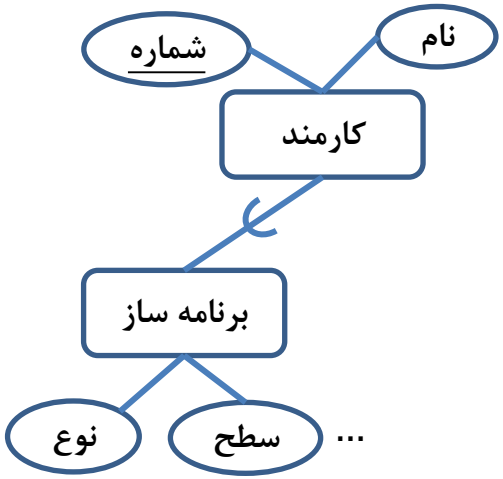
- نکات:
  - **زبرنوع** مجموعه صفاتی دارد مشترک در تمام **زیرنوع ها**
    - درنتیجه **زیرنوع** تمام صفات **زبرنوع** را به ارث می‌برد (وراثت صفات از نوع ساختاری).
    - مفهوم ارث‌بری با تکنیک ارتباط IS-A مدلسازی می‌شود.
    - 

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

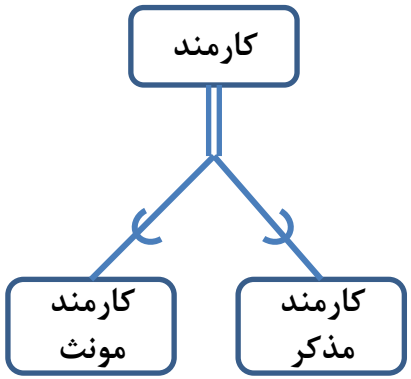
ارتباط "IS A" – تخصیص

- **تخصیص**
  - **۱- کامل:** تمام زیرنوع های زبرنوع با توجه به ضابطه در مدلسازی دخالت داده می‌شود. هر نمونه از زبرنوع، جزء مجموعه نمونه‌های حداقل یکی از زیرنوع‌ها است.
  - **۲- ناقص:** براساس ضابطه تمام زیرنوع های زیرنوع در نظر گرفته نمی‌شوند. هر نمونه از زبرنوع لزوما جزء مجموعه نمونه‌های یکی از زیرنوع‌ها نیست.

تخصیص ناقص: براساس مهارت کارمند فقط برنامه سازان را جدا کرده ایم.



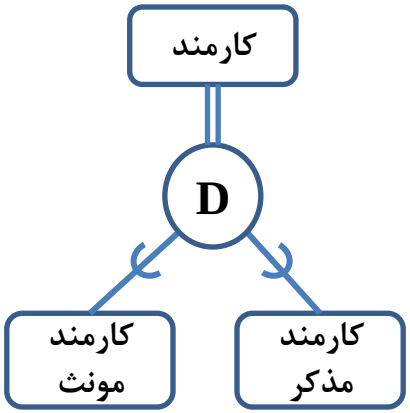
تخصیص کامل



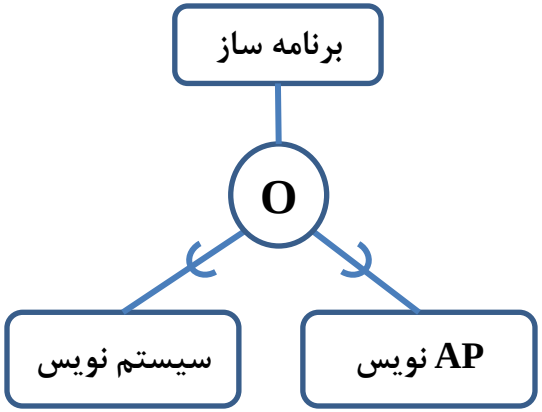
ارتباط "IS A" – تخصیص (ادامه)

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

مثال تخصیص مجزا

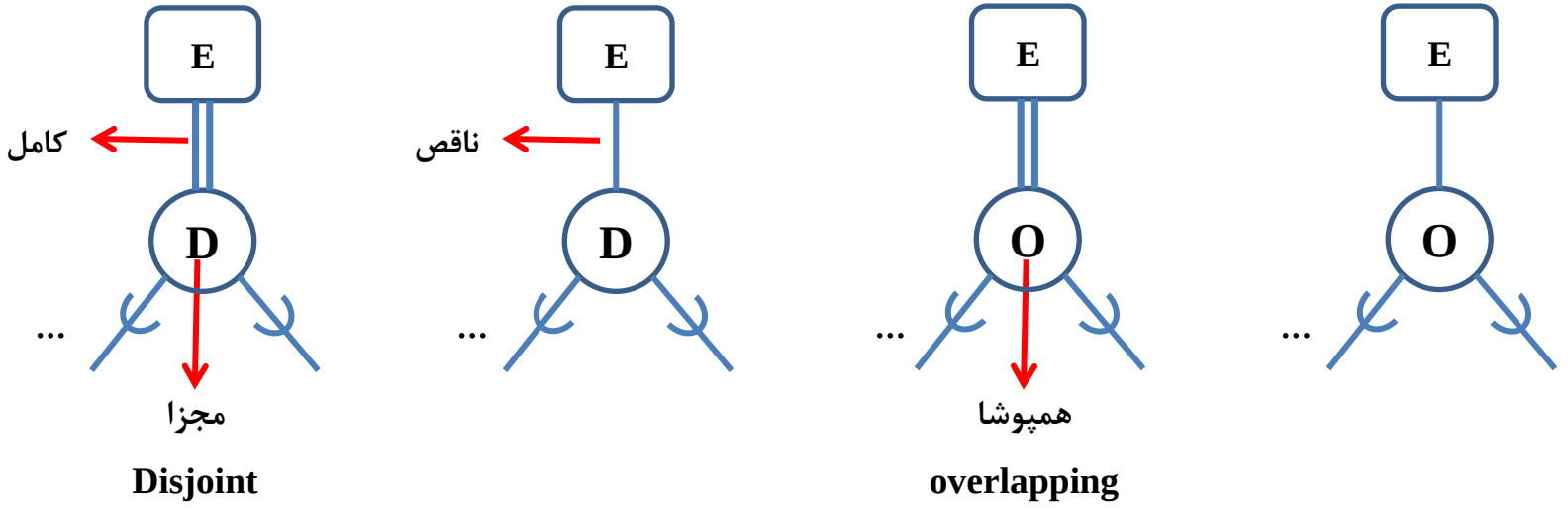


مثال تخصیص همپوشا



ارتباط "IS A" – تخصیص (ادامه)

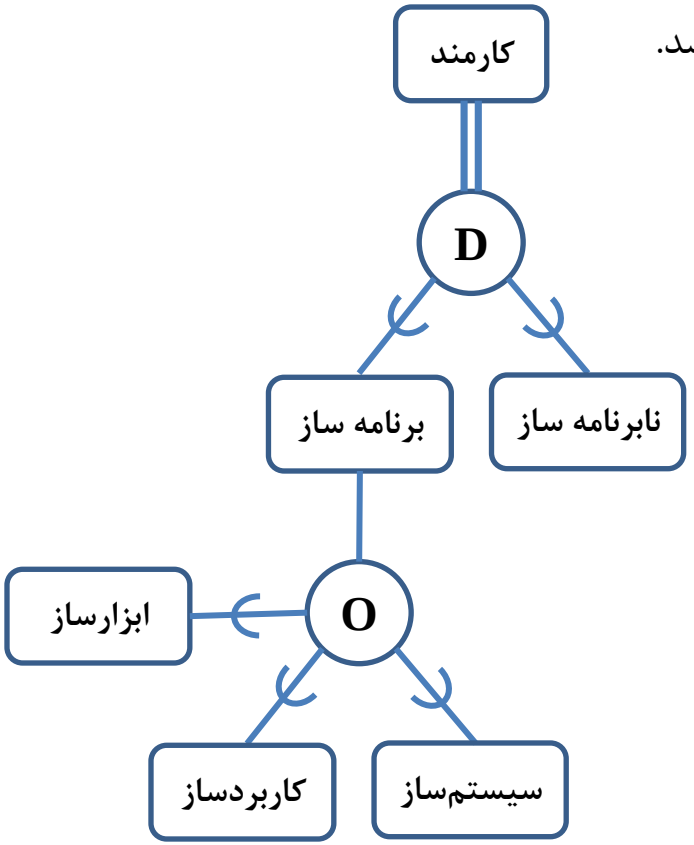
- براساس این دو ویژگی چهارگونه تخصیص داریم:



ارتباط "IS A" (ادامه)

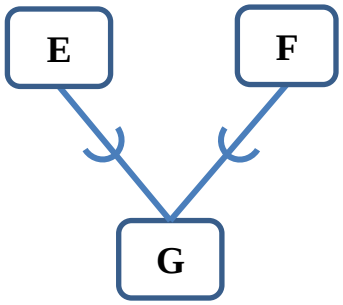
ادامه نکات:

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



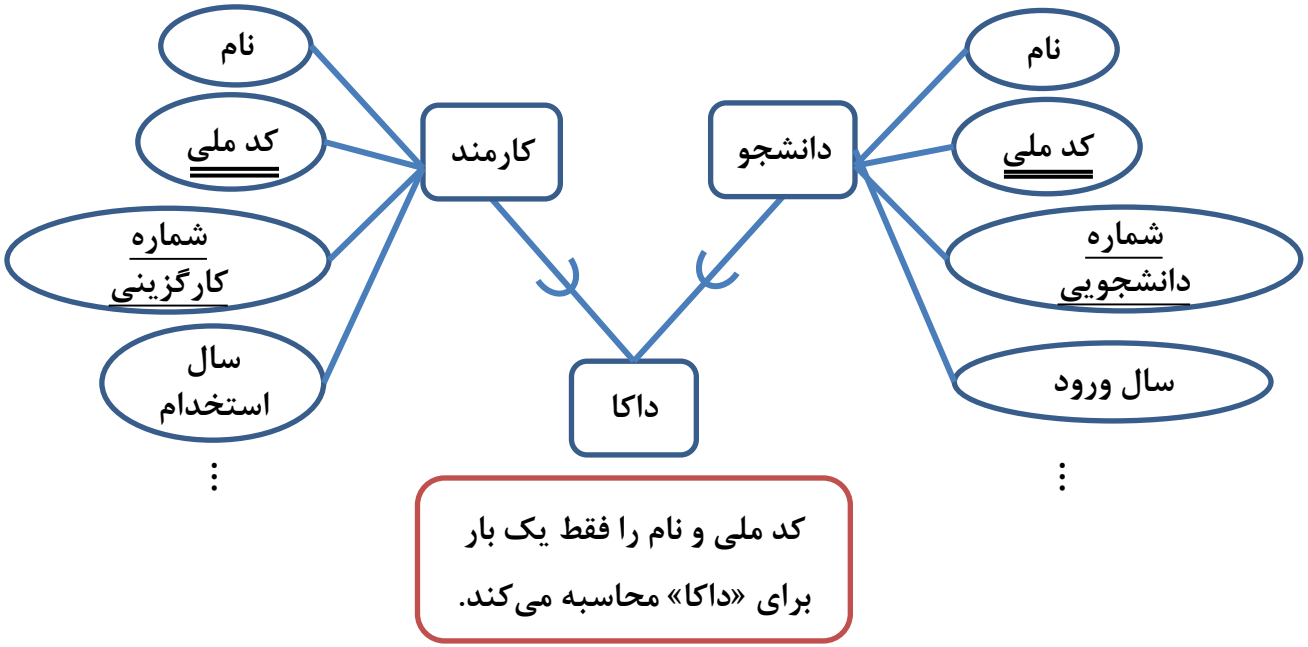
ارتباط "IS A" (ادامه)

- **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



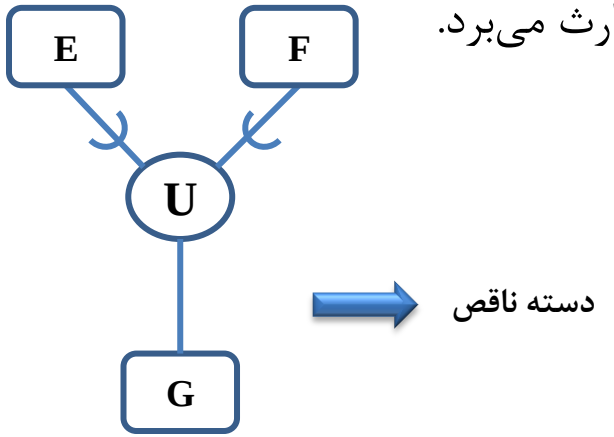
زیرنوع اجتماع (U-Type)

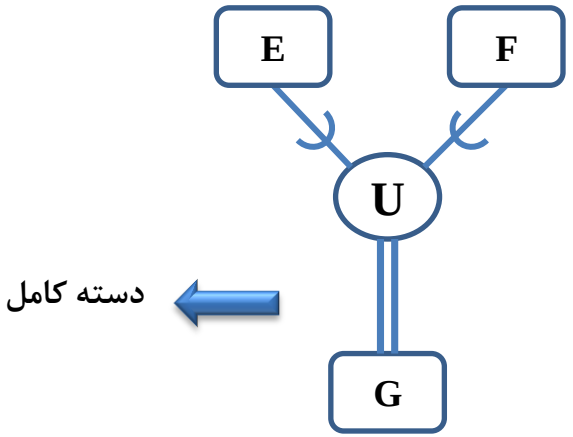
- زیرنوع اجتماع (U-Type) یا Category «دسته»
- **زیرنوع** موجودیت G را زیرنوع U-Type **زیرنوع‌های** E,F,... گوییم هرگاه در مجموعه نمونه‌های G نمونه‌هایی از E,F,... وجود داشته باشد. در واقع نمایانگر اجتماعی از نمونه‌ها از انواع مختلف است.

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.

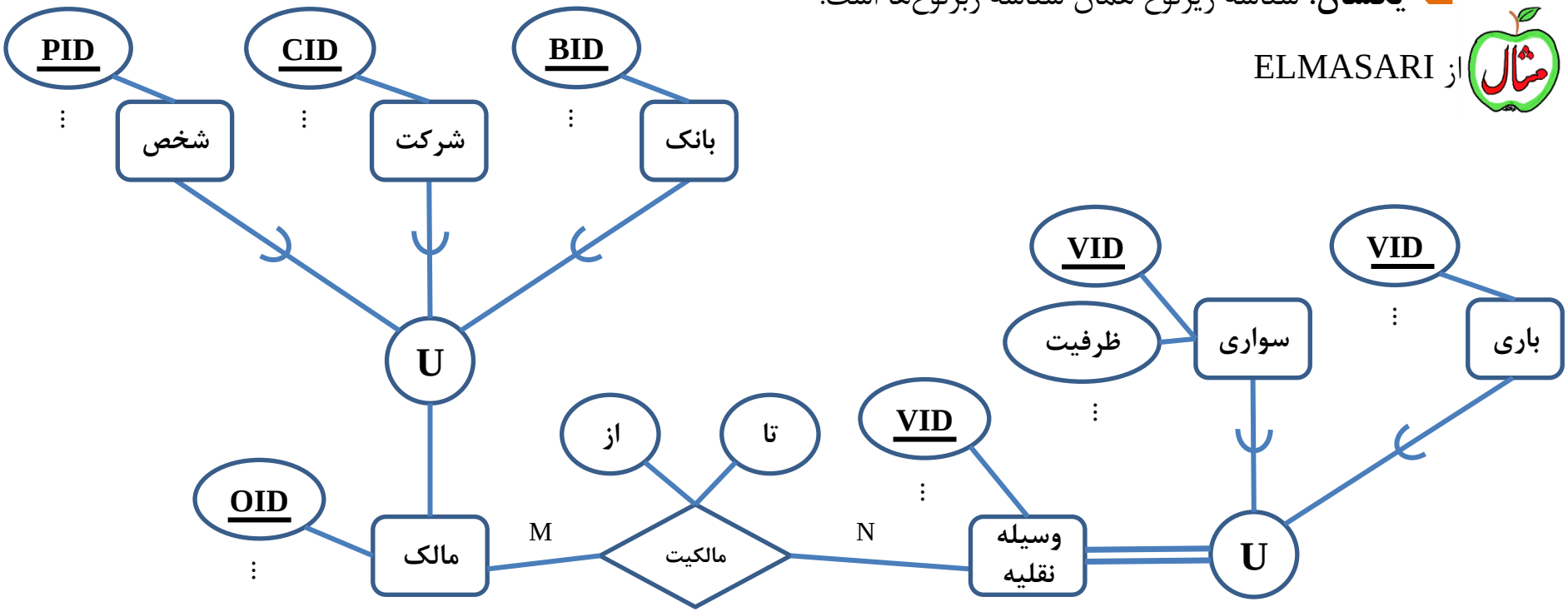




زیرنوع اجتماع (ادامه)

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

مثال از ELMASARI



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

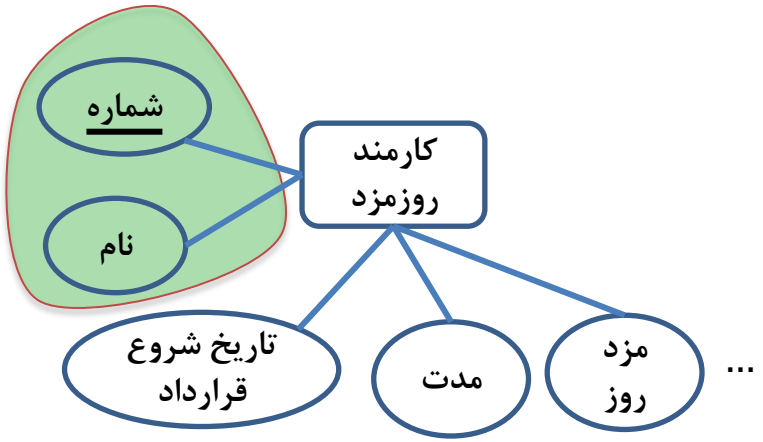
زیرنوع اجتماع (ادامه)

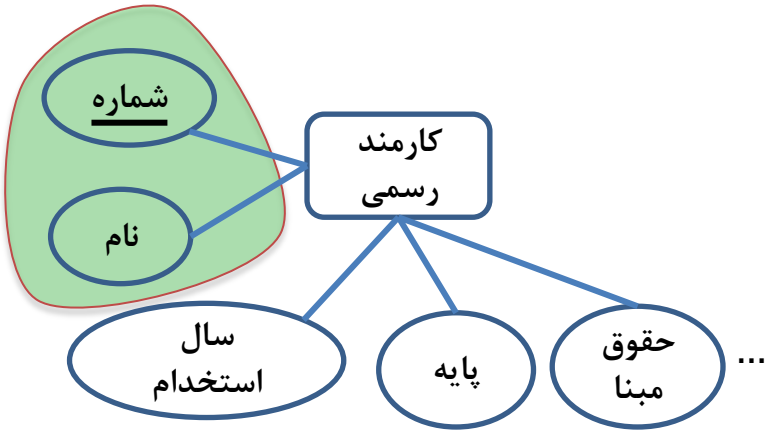
- تمرین : برای محیط با مفاهیم زیر، هم با U-Type و هم بدون U-Type یک مدلسازی ارایه دهید:
  - بانک - دانشگاه
  - شخص (دانشجو – استاد – کارمند و متفرقه)
  - حساب بانکی (کوتاه مدت – بلند مدت – قرض الحسنه و...)
  - عملیات واریز – برداشت - انتقال وجه

تعمیم (Generalization)

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

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

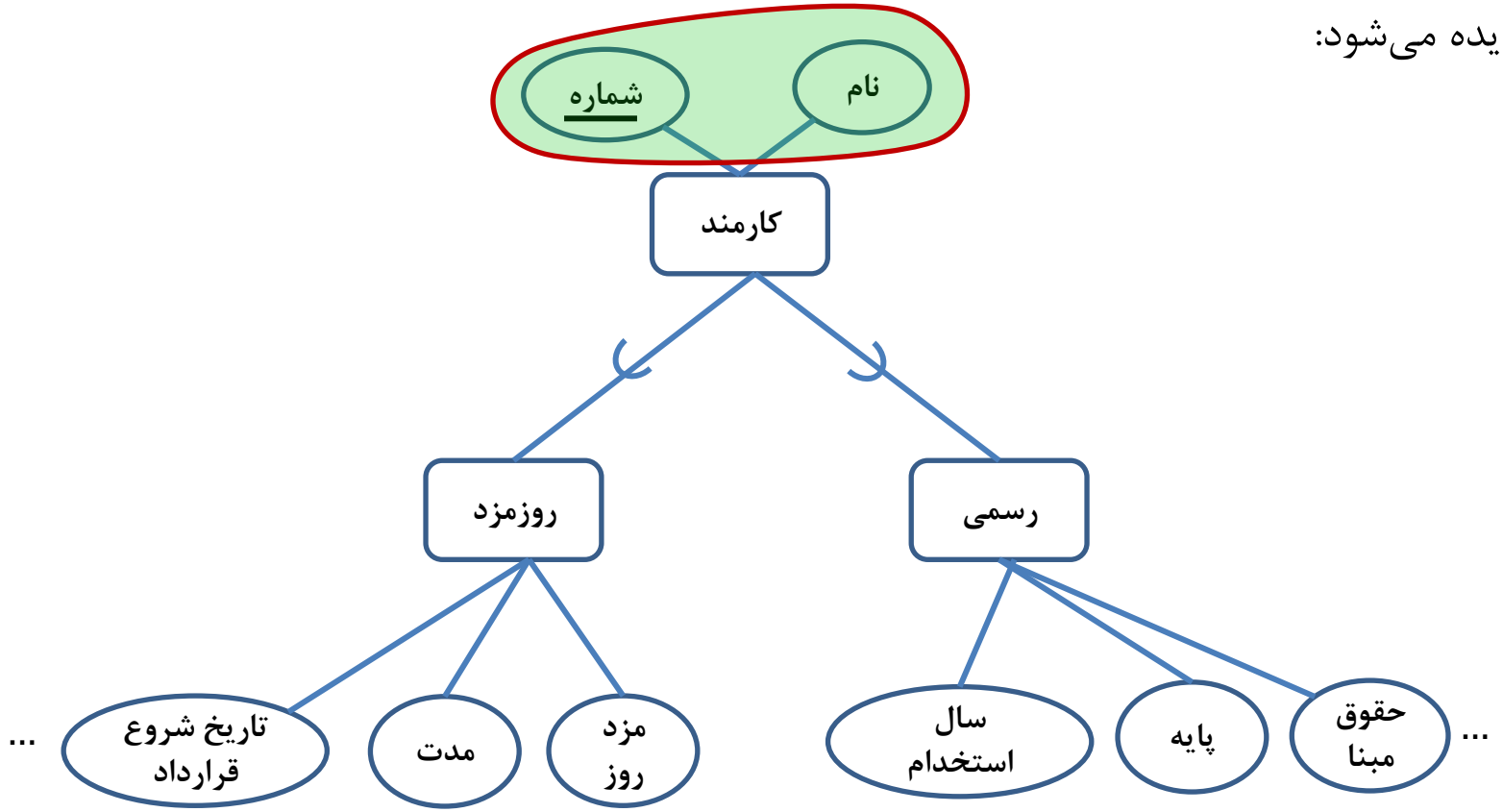




تعمیم (ادامه)

ادامه:

- یک نوع موجودیت (کارمند) در سطح انتزاعی بالاتر دیده می‌شود:



تعمیم (ادامه)

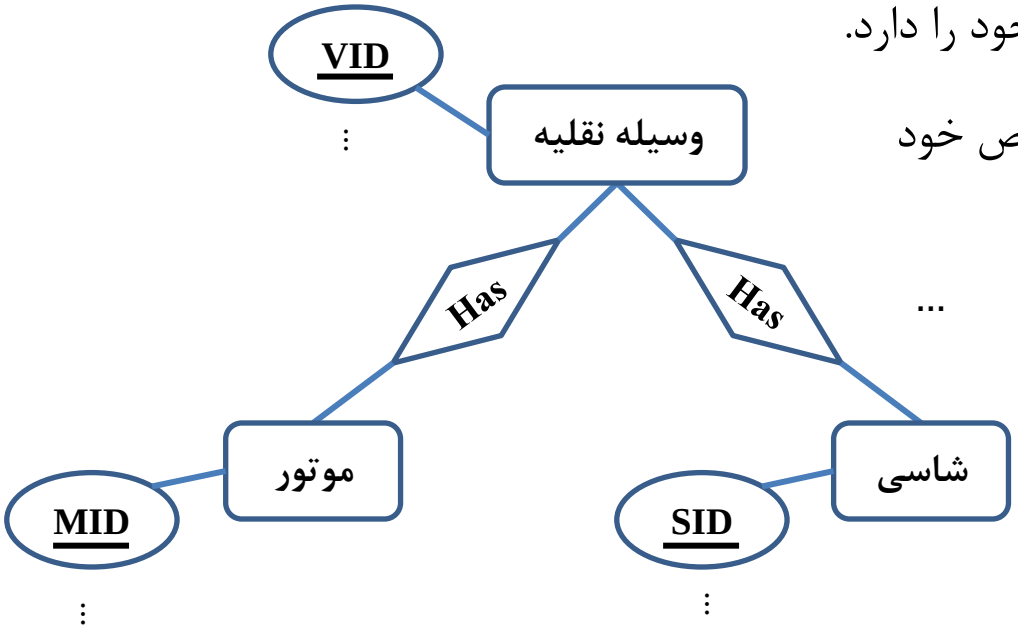
- **شرایط تعمیم:**
  - داشتن شناسه مشترک [یعنی از یک دامنه]
  - حداقل وجود دو نوع زیرنوع
  - هرچه صفات مشترک بیشتر، تعمیم توجیه پذیرتر است [شرطِ لازم نیست ولی شرطِ ارجحیت است].

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].

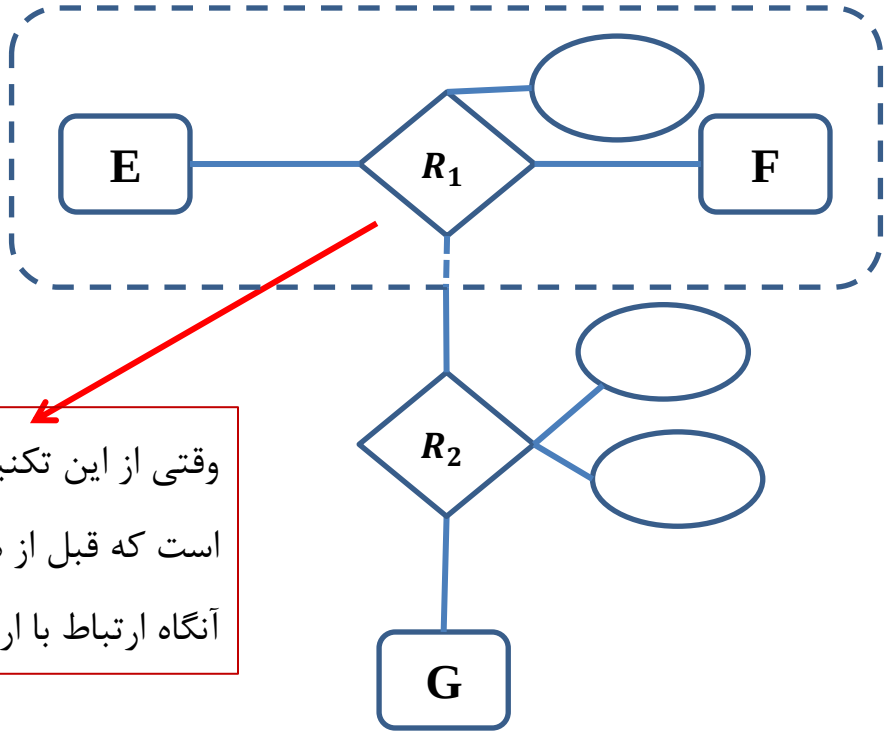
مثال ارتباط شاسی و موتور با وسیله نقلیه



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 موجودیت R صفاتی هم داشته باشد).
- ارتباط با ارتباط حیطه معنایی خاص خود را دارد.

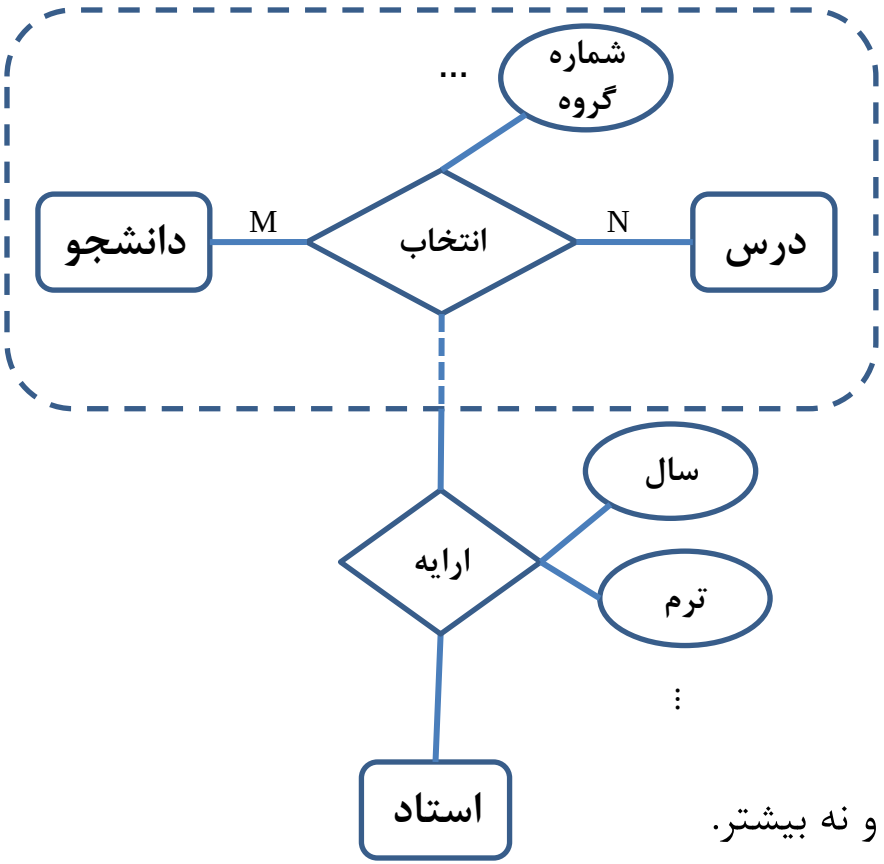


وقتی از این تکنیک استفاده می‌شود، معنایش این است که قبل از هر چیز به ارتباط R_1 نیاز است. آنگاه ارتباط با ارتباط مطرح شده است.

ارتباط با ارتباط (ادامه)

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

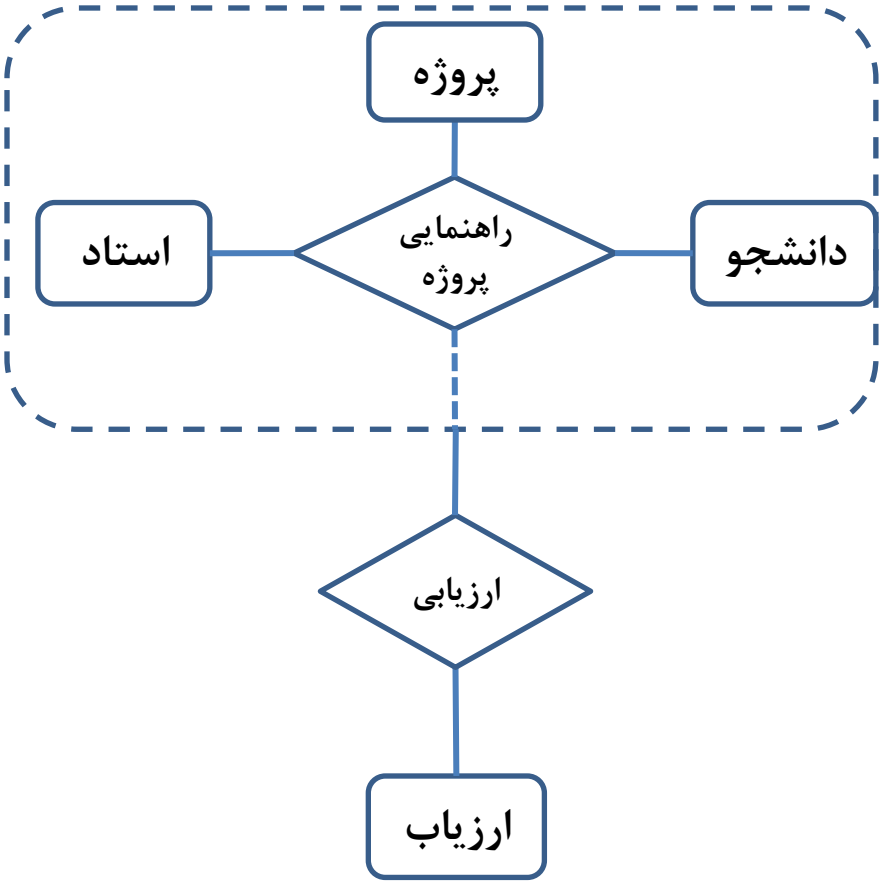
مثال طرز دیگر مدلسازی برای برای محیط دانشجو – درس – استاد:



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

ارتباط با ارتباط (ادامه)

مثال: ارزیابی راهنمایی پروژه پژوهشی دانشجو توسط استاد



- **نکات زیر بررسی شود:**
  - ویژگی های عمومی روش مدلسازی
  - کمداشت های روش [E]ER
  - تناظر بین مفاهیم روش [E]ERو روش UML [در نمودار رده Class diagram]

مراحل مدلسازی معنایی داده‌ها

۱- مطالعه، تحلیل و شناخت محیط

۲- برآورد خواسته‌ها و نیازهای اطلاعاتی و پردازشی همه کاربران ذیربط محیط (مهندسی نیازها) و تشخیص محدودیت‌های معنایی و قواعد فعالیت‌های محیط

۳- بازشناسی نوع‌موجودیت‌های مطرح و تعیین وضع هر نوع‌موجودیت

۴- تعیین مجموعه صفات هر نوع‌موجودیت، میدان و جنبه‌های هر صفت

۵- بازشناسی نوع‌ارتباط‌های بین نوع‌موجودیت‌ها، تشخیص الزامی بودن یا نبودن مشارکت در آنها و تشخیص چندی هر ارتباط

۶- رسم نمودار ER (یا EER) به صورت واضح، خوانا و حتی‌الامکان با کمترین افزونگی

۷- فهرست کردن پرسش‌هایی که پاسخ آنها از نمودار به دست می‌آید (بر حسب گزارش‌های مورد نیاز و کلا نیازهای داده‌ای کاربران)

۸- وارسی مدلسازی انجام شده، برای اطمینان از پاسخگو بودن به نیازهای کاربران.

یکپارچه‌سازی نمودارهای جزیی

- گاه به علت وسعت محیط عملیاتی و تعدد کاربران آن لازم است مدلساز به ازای هر زیرمحیط و یا حتی یک کاربر نمودار ER رسم کند.
- در این صورت نیازمند **ادغام و یکپارچه‌سازی نمودارهای ER** هستیم.
- در ادغام چند نمودار ER باید به تعارض‌های (ماهیتا معنایی) بین نمودارها توجه کرد. از جمله موارد زیر:
  - مدلهای نایکسان برای ایده واحد
  - تعارض در نامگذاری یک مفهوم (از لحاظ معنایی) واحد
  - تعارض معنایی دو مفهوم ظاهرا یکسان
  - تعارض در میدان صفت‌ها
  - تعارض در رفتارها و محدودیت‌ها
- تحلیل این تعارض‌ها قبل از تصمیم‌گیری درباره ادغام ERها باید انجام شود.

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش سوم :
طراحی منطقی

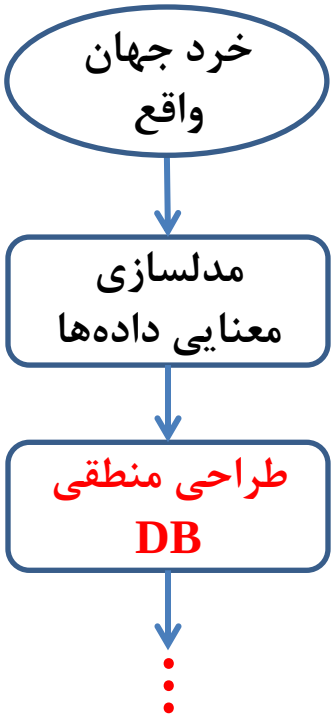
مرتضی امینی

نیمسال اول ۹۲-۹۳

(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

- **چرا DS (در معنای عام)؟**
  - برای نمایش نوع موجودیت‌ها و ارتباط بین آنها در سطح { منطقی / فیزیکی }

- **دلایل لزوم DS در حیطه پایگاهی:**

۱- تامین کننده محیط فرافایلی (محیط انتزاعی)

۲- مبنا و چارچوب طراحی منطقی DB

۳- مبنا و چارچوب طراحی زبان پایگاه داده‌ها DBL

۴- مبنا و چارچوب طراحی خود DBMS

۵- ضابطه‌ای است برای مقایسه سیستم‌ها و ارزیابی آنها

۶- مبنایی است برای ایجاد و گسترش تکنیک‌های طراحی DB

۷- . . .

ساختارهای داده (ادامه)

- DSها [در حیطه دانش و تکنولوژی DB]:
  - ۱- HDS [از Hierarchical DM] ← HDBMS (پیش رابطه‌ای (پکیج IMS و IDMS))
  - ۲- NDS [از Network DM] ← NDBMS (پیش رابطه‌ای (پکیج IMS و IDMS))
  - ۳- RDS [از Relational DM] ← RDBMS پکیج‌های جدولی
  - ۴- ODS [از Object DM] ← ODBMS
  - ۵- ORDS [از Object Relational DS] ← ORDBMS

- **TDS- ساختار داده جدولی:**
  - عنصر ساختاری اساسی در Relational Model (RM): مفهوم **رابطه**
  - رابطه [Relation]: یک مفهوم ریاضی است ...
  - اما از دید کاربر [در عمل]: نمایش جدولی دارد.
    - فعلا به جای RDS می‌گوییم TDS.

ساختار داده جدولی

- **اصطلاحات TDS:**
  - ***نوع جدول*** ← برای نمایش **نوع**: نام جدول؛ نام و نوع ستون ها؛ موجودیت و/یا ارتباط
  - ***سطر*** ← برای نمایش **نمونه**: موجودیت؛ ارتباط
  - ***ستون*** ← برای نمایش **صفت**
- **عنصر ساختاری اساسی:**
  - هر DS حداقل یک **عنصر ساختاری اساسی** دارد.

تعریف: عنصری است که به کمک آن نوع موجودیت، نوع ارتباط، و یا هردو آنها را نمایش می‌دهیم.

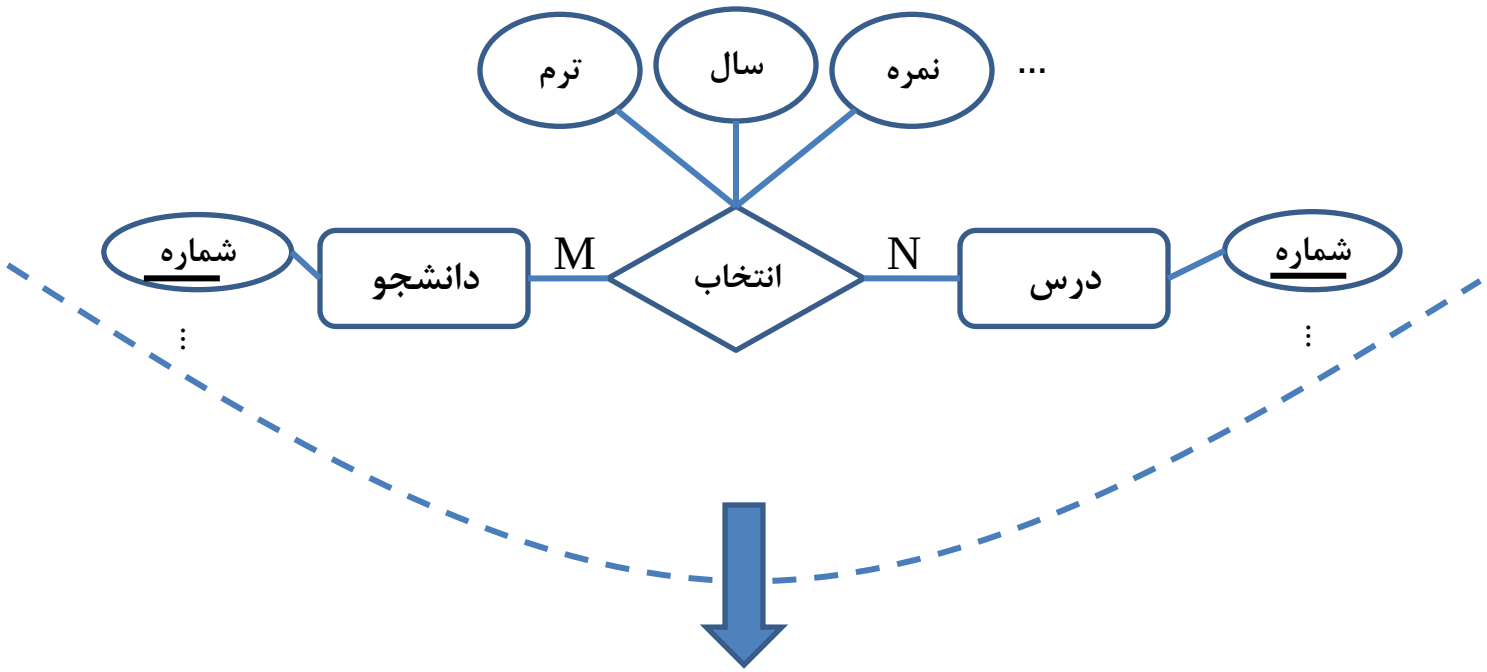
  - TDS فقط یک عنصر ساختاری اساسی دارد : همان **نوع جدول**
  - **نکته:** صفت **شناسه** در نوع موجودیت‌ها، حکم **کلید** را در جدول دارد.

TDB چیست؟

- از دید کاربر
  - طراح
  - پیاده ساز AP
  - برنامه ساز AP
- **از لحاظ نوع:** مجموعه‌ای است از تعدادی **نوع جدول** (که آنها را طراحی می‌کنیم)
- **در سطح نمونه** [از لحاظ محتوای داده‌ای]: مجموعه‌ای است از نمونه‌های متمایز یک [چند] **نوع سطر**
- نوع سطر را همان نوع جدول مشخص می‌کند.

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)

STT

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

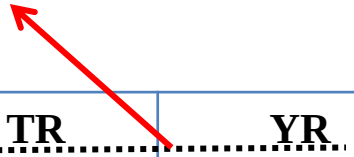
COT

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 برای رابطه باشند، جزو کلید محسوب می‌شوند.)

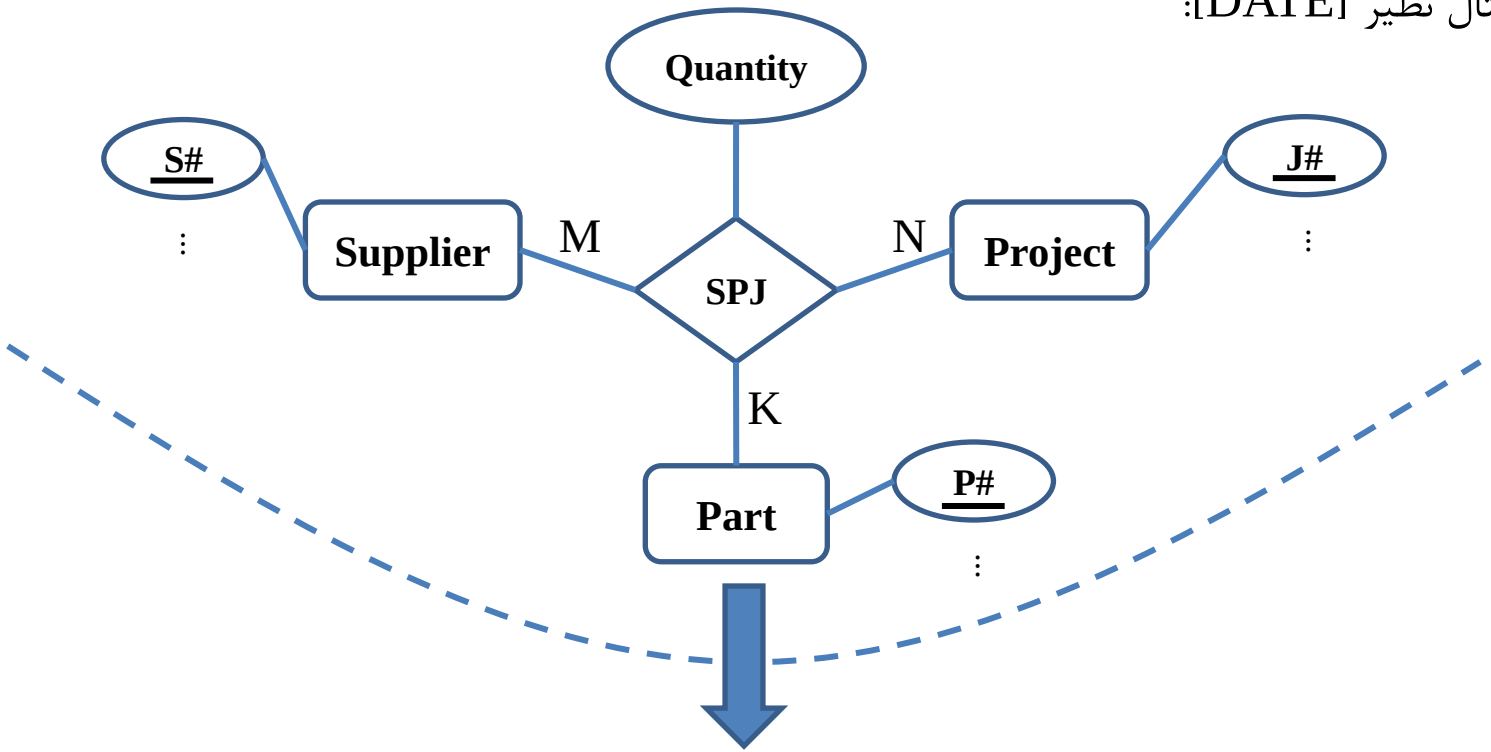


STCOT

STID	COID	TR	YR	GRADE
⋮	⋮	⋮	⋮	⋮
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13

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)

Supplier

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

Part

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

Project

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

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

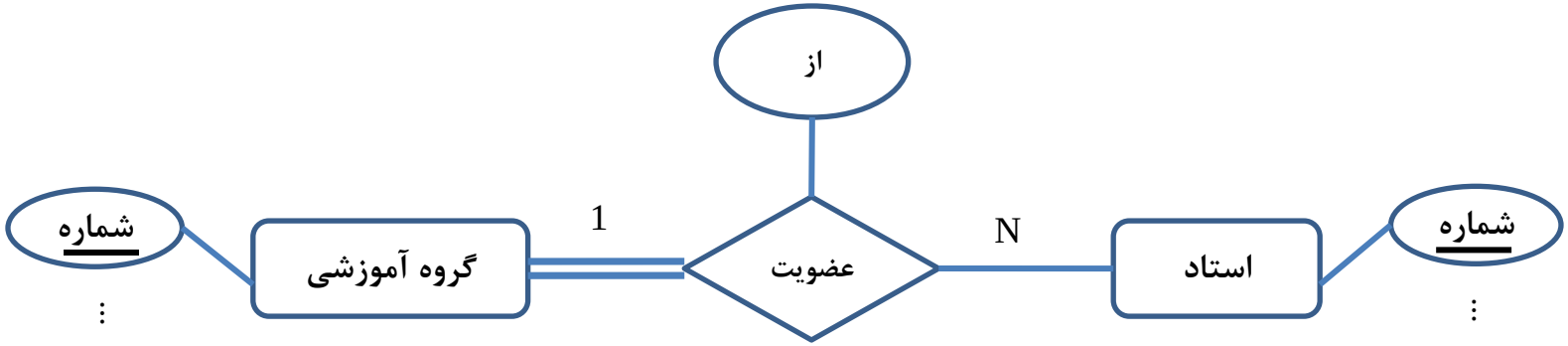
(در واقع اگر صفت چند مقداری باشد، جزو کلید محسوب می‌شود.)

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	…	PRPHONE	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	…	…	…
⋮	⋮	⋮	⋮	⋮

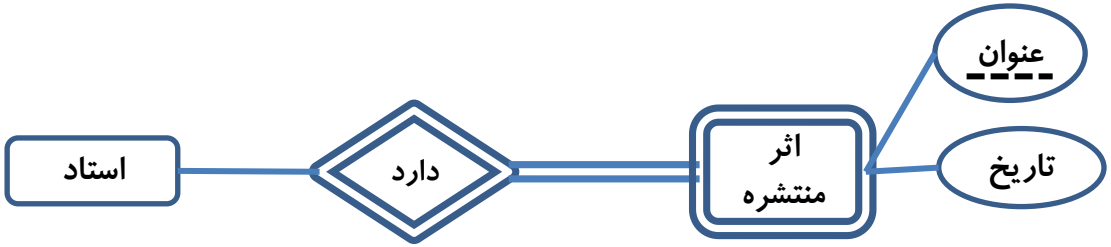
PROF

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

طرزهای دیگر طراحی؟

طراحی منطقی با TDS – رابطه شناسا

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	…	PRPHONE
Pr100	…	استاد	…	…
Pr200	…	استادیار	…	…
Pr300	…	دانشیار	…	…
⋮	⋮	⋮	⋮	⋮

PUB

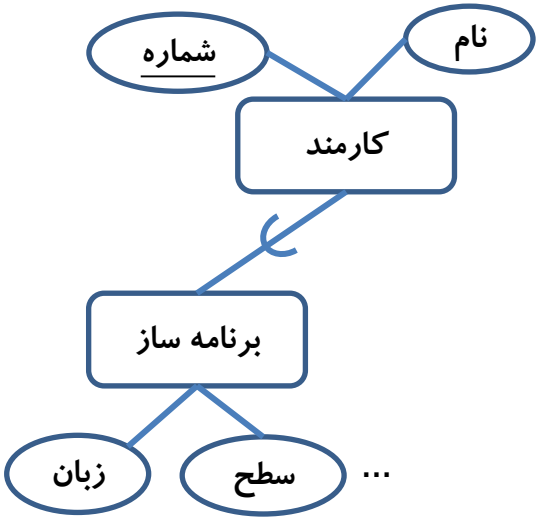
PRID	PTITLE	…	PDATE
Pr100	Data Encryption…	…	…
Pr100	Semantic Analysis of …	…	…
⋮	⋮	⋮	⋮

❊ 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.

طراحی منطقی با TDS – رابطه IS-A

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

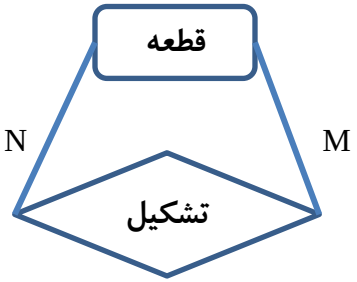
EID	LANG	…	LEVEL
E100	C++	…	…
E102	Java	…	…
⋮	⋮	⋮	⋮

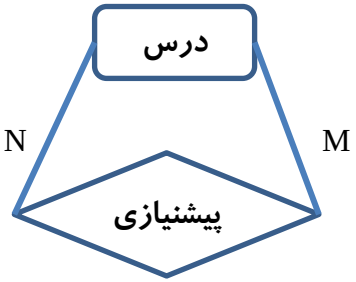
❊ EID (کلید خارجی از جدول EMP) کلید اصلی جدول PROG نیز هست.

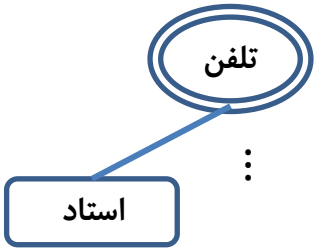
کنکاوی؟ حذف و بروزرسانی در جدول EMP چه تاثیری بر PROG باید داشته باشد (و بالعکس)؟

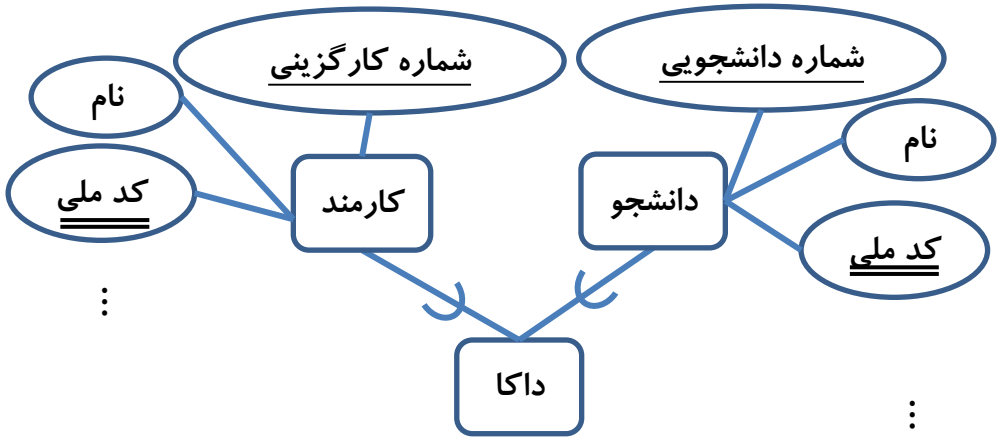
طراحی منطقی با TDS (ادامه)

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



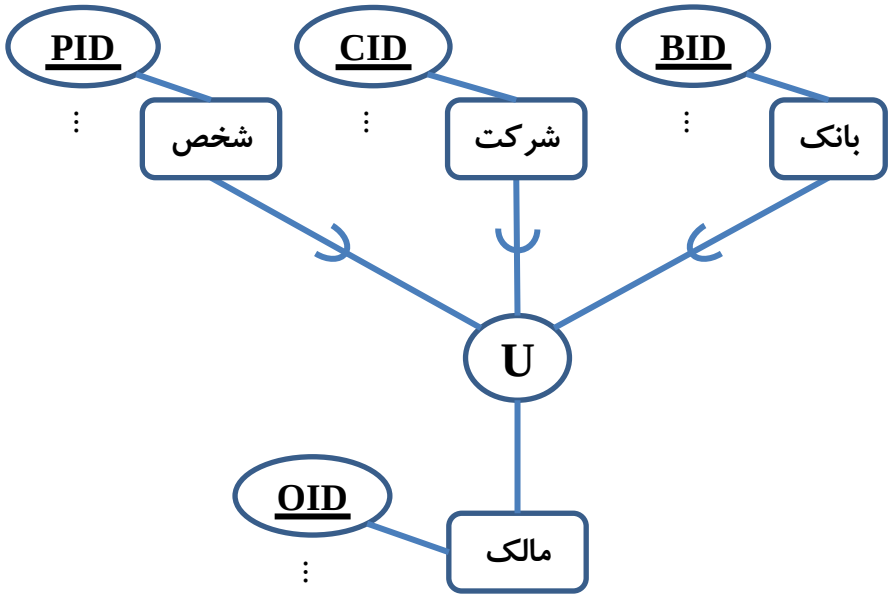


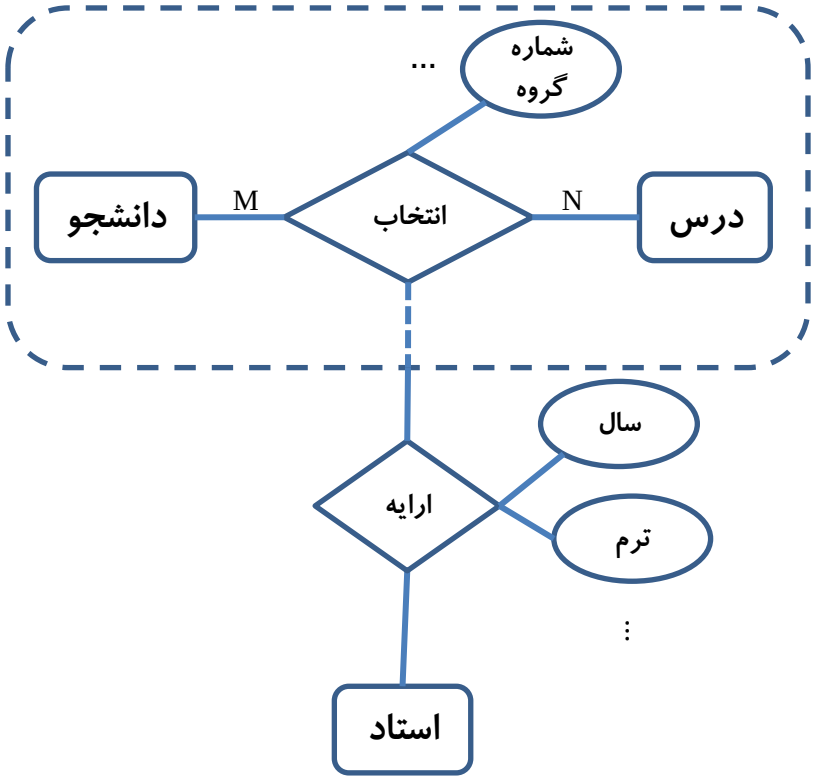




طراحی منطقی با TDS (ادامه)

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





- دستورهای Structured Query Language (SQL):
  - Date Definition Language (DDL)
  - Date Manipulation Language (DML)
  - Data Control Language (DCL)

- چند دستور از DDL:
  - ایجاد جدول CREATE TABLE
  - حذف جدول DROP TABLE
  - تغییر جدول ALTER TABLE

- **شمای پایگاه داده‌ها** عبارت است از تعریف (توصیف) ساختهای منطقی طراحی شده و نوعی برنامه است شامل تعدادی دستور برای تعریف و کنترل داده‌ها.
- **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 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, DECIMAL(p, q), 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.

مثالی از شِمای پایگاهی

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 شِمای پایگاهی (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) 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)

شِمای پایگاهی

- در شِمای پایگاهی ← دستورات
  - Data Definition (DD)
  - Data Controller (DC)

و نه دستورات
Data Manipulation (DM)

این جدایی چه مزایایی دارد؟

سیستم با شِمای پایگاهی چه می‌کند؟

- اطلاعات موجود در آن را در جایی به نحوی ذخیره می‌کند. ← **در تعدادی جدول**

حاوی فراداده‌ها و داده‌های کنترلی در مورد داده‌های کاربران ←
- **کاتالوگ سیستم**
- دیکشنری سیستم
- مِتا داده‌ها

آشنایی با کاتالوگ

مثال

مثالی از جدول‌های کاتالوگ:

SysTables

نام جدول	ایجاد کننده	تاریخ	تعداد ستون	...
STT	C1	D1	5	
COT	C1	D2	5	
SCT	C2	D2	5	
⋮	⋮	⋮	⋮	

جدولی که جدول‌ها را مدیریت می‌کند.

SysCols

نام ستون	نام جدول	نوع	طول	...
STID	STT	CHAR	8	
STNAME	STT	CHAR	25	
⋮	⋮	⋮	⋮	
GR	SCT	DEC	2,2	

جدولی که ستون‌ها را مدیریت می‌کند.

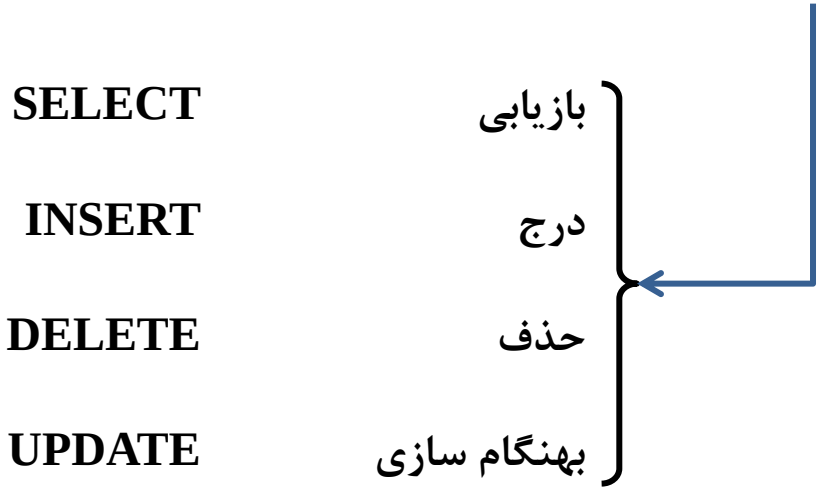
آشنایی با کاتالوگ (ادامه)

آیا برنامه ساز می تواند محتوای کاتالوگ را مستقیما تغییر دهد؟ (با دستورات INSERT, DELETE,UPDATE)

- تمرین: حداقل سه جدول دیگر برای کاتالوگ طراحی کنید.
- تمرین: چه اطلاعاتی در کاتالوگ ذخیره می شود؟

زبان جدولی TDBL

- 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.

❑ تمرین: روش دیگر؟

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


```
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.

بازیابی داده‌ها (ادامه)

NULL


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

بررسی برخورد یک package با NULL؟

عملگرهای جبر مجموعه‌ها

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

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

- 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:** حذف تکراری‌ها در نتیجه اجرای عملگرهای جبر مجموعه‌ها

عملگرهای جبر مجموعه‌ها (ادامه)

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

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

تست سازگاری پایگاه داده‌ها:

```
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)

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
  - MIN
  - MAX
  - SUM
  - COUNT و COUNT(*)

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)

گروه‌بندی

GROUP BY

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

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

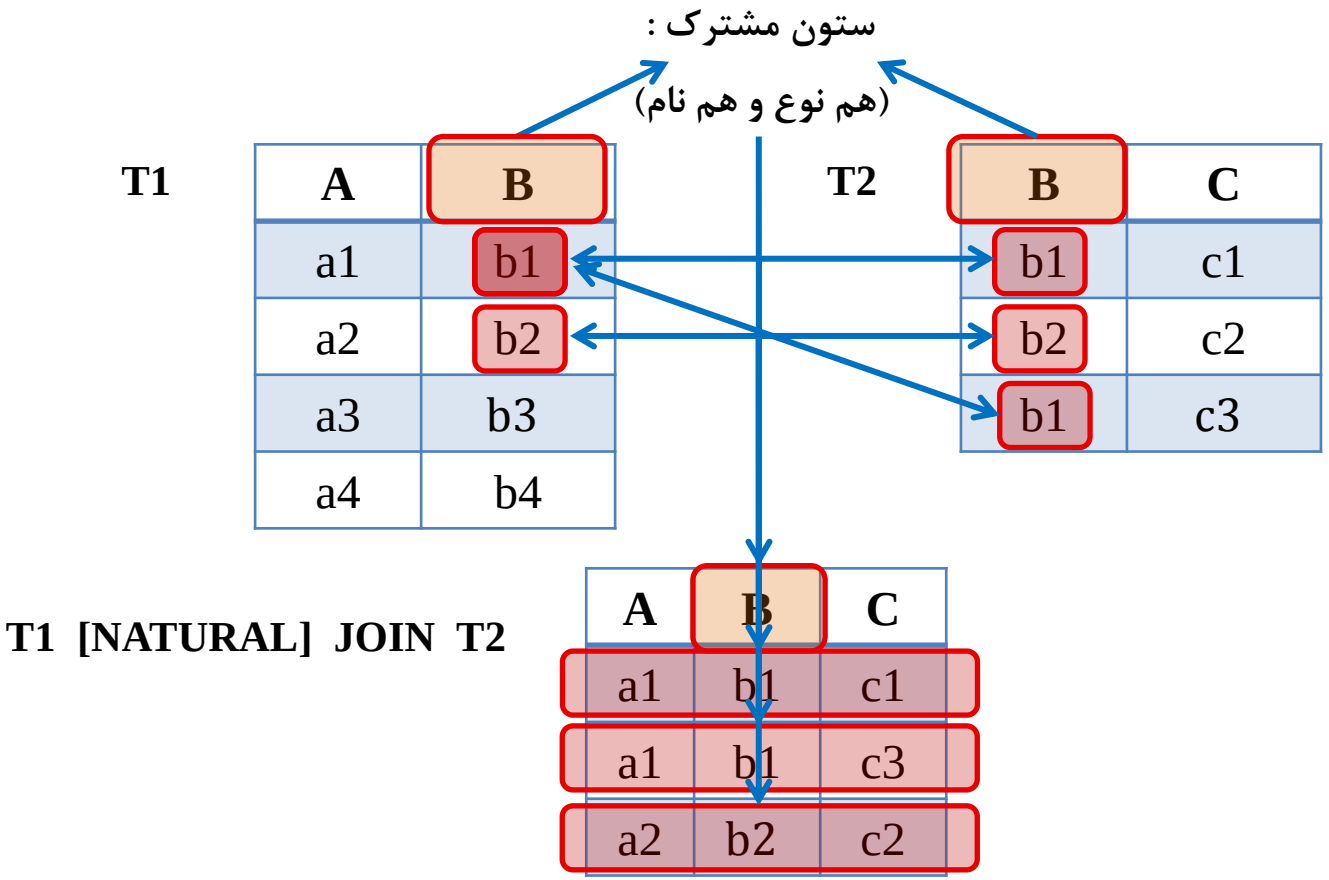
```
SELECT  T1.* , T2.*
        FROM    T1, T2
```

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
…	…	

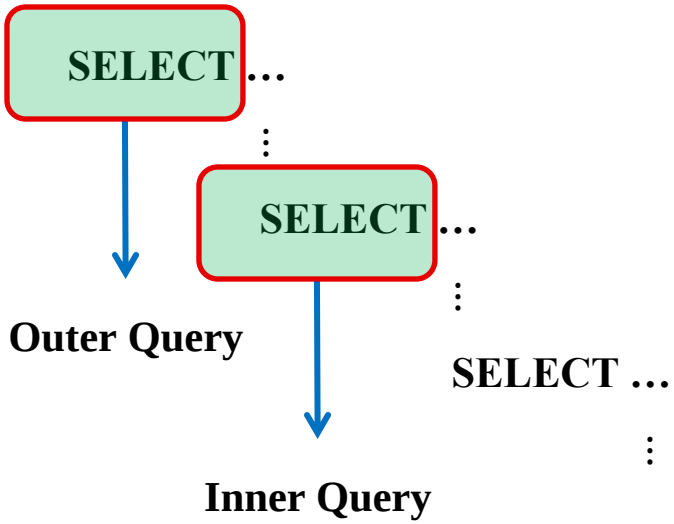
S [NATURAL] JOIN SP

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

روش پنجم

یا = ANY

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 هرگار 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

theta	ANY
theta	SOME
theta	ALL

theta ∈ { =, ≠, <, ≤, ≥, > }

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 **)**

2- SELECT S#

چون جواب SELECT تک مقداری است نیازی به ANY نیست.

FROM S

WHERE STATUS < **(SELECT MAX (**STATUS**) FROM** S**)**

بازیابی از بیش از یک جدول (ادامه)

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 ؟

دستورهای INSERT, UPDATE, DELETE

درج INSERT:

INSERT INTO *table-name*
VALUES (*one row*) | *subquery*

بهنگام‌سازی UPDATE:

UPDATE *table-name*
SET *col = value / scalar ...*
⋮
WHERE *condition(s) / subquery*

حذف DELETE:

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

درج سطری (سطر کامل – سطر ناقص):

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

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

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

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

آیا این حذف باید انتشار یابد؟

حذف از بیش از یک جدول:

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

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

دیگر امکانات SQL

- مطالعه شود :
  - پرسش بازگشتی
  - SQL ادغام شده
  - SQL پویا
  - نوشتن رویّه
  - نوشتن تابع
  - امکانات شیئ- رابطه‌ای
  - مدیریت تراکنش

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش چهارم:
معماری پایگاه داده‌ها

مرتضی امینی

نیمسال اول ۹۲-۹۳

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

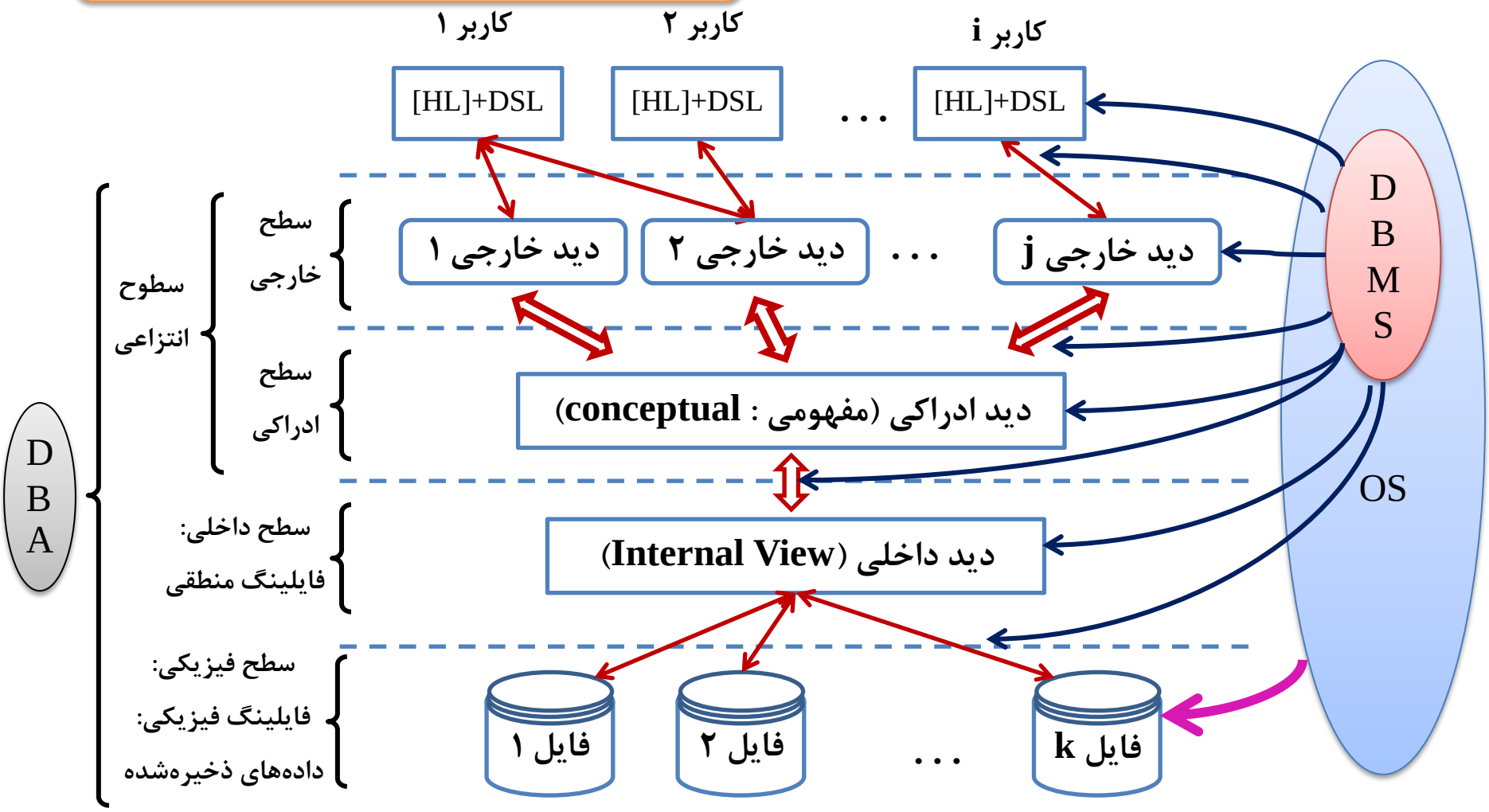
معماری پایگاه داده‌ها

- نیاز به یک معماری واحد از دیدگاه **داده شناسانه** (و نه دیدگاه عملکردی یا دیدگاه مولفه-مبنا)
- 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



اجزای معماری

اجزای معماری سه سطحی پایگاه داده‌ها:

۱- کاربر User
۲- زبان میزبان HL: مانند زبانهای جاوا، C# و دلفی
۳- زبان داده‌ای فرعی (زیرزبان داده‌ای) DSL: زبانهای داده‌ای ادغام شده در زبانهای میزبان

۴- دید خارجی (نمای خارجی) ⟵ **سطح خارجی**
۵- دید ادراکی (فرایافتی یا مفهومی) ⟵ **سطح ادراکی**
۶- دید داخلی ⟵ **سطح داخلی**

۷- فایل‌های فیزیکی
۸- سیستم مدیریت پایگاه داده‌ها (کوتاهتر: سمپاد)
۹- مدیر پایگاه داده‌ها (DBA)

دید ادراکی

دید (نمای) ادراکی (فرایافتی یا مفهومی)

تعریف: دید طراح نسبت به داده های ذخیره شدنی (ونهایتا ذخیره شده) در پایگاه داده‌ها

- ✓ دیدی جامع: دربرگیرنده نیازهای همه کاربران محیط
- ✓ مطرح در محیط انتزاعی (فرافایلی) ← مبتنی بر یک ساختار داده مشخص
- ✓ طراحی با عنصر (عناصر) ساختاری اساسی
- ✓ پس از طراحی ← توصیف شود ← **شمای ادراکی (Conceptual Schema)**

تعریف: **نوعی برنامه حاوی دستورات** { **DDL**, **DCL** } **و نه دستورات DML**

- ✓ شمای ادراکی به سیستم مدیریت داده می شود و در کاتالوگ آن نگهداری می‌شود.

دید ادراکی (ادامه)

CREATE TABLE S1 …

CREATE TABLE S2 …

CREATE TABLE S3 …

- **دید ادراکی:** جدول‌های مبنای S1 و S2 و S3
- **شمای ادراکی:** تعریف جدول‌ها است.
- به این جدول‌ها **جدول‌های مبنا** می‌گوییم.

در کجا ؟ ← کاتالوگ سیستم

چگونه ؟ ← در سیستم‌های جدولی: در تعدادی جدول که خود سیستم ایجاد می‌کند.

- اطلاعات شِمای ادراکی به سیستم مدیریت داده می‌شود و در کاتالوگ آن نگهداری می‌شود.

کاتالوگ سیستم : متا داده‌ها (Data Dictionary : Meta Data)

- حاوی :
  - ✓ تمامی اطلاعات شمای ادراکی
  - ✓ داده های کنترلی
  - ✓ Data About Data

دید ادراکی (ادامه)

جدول systables:

systables

نام جدول	ایجاد کننده	تاریخ ایجاد	تعداد ستون	کلید اصلی	...
...	...	...	...	...	...

کاربر پیاده‌ساز : **CREATE TABLE** STT **...**

سیستم : **INSERT INTO** SYSTABLES

VALUES (‘STT’ , ‘c1’ , ‘d1’ , 5 , ‘STID’ , … **)**

کاربر پیاده‌ساز : **DROP TABLE** STCOT **...**

سیستم : **DELETE FROM** SYSTABLES

WHERE TNAME = ‘STCOT’

دید ادراکی (ادامه)

- اضافه کردن ستون به یک جدول :

کاربر پیاده‌ساز : ALTER TABLE STT
ADD SADDRESS CHAR (80)

سیستم : UPDATE SYSTABLES
SET ColN = 6
WHERE TNAME = ‘STT’


سیستم برای جدولی که تعداد ستون‌های آن تغییر می‌کند در سطح فایلینگ چگونه عمل می‌کند؟


آیا با دستور DELETE جدول کاتالوگ تغییر می‌کند؟

DELETE FROM STT
WHERE STID=‘777’

دید داخلی

دید (نمای) داخلی

تعریف: دید خود DBMS [و نیز طراح پایگاه داده‌ها، در مرحله طراحی فیزیکی] ، نسبت به داده‌های ذخیره‌شده

- ✓ مطرح در سطح فایلینگ منطقی (و گاه مجازی)
- ✓ مبتنی بر یک [یا چند] **ساختار فایل** ←
  - **1:1** پیش فرض (یک جدول : یک فایل)
  - **1:N** (چند جدول : یک فایل)
  - **N:1** نادر (یک جدول : چند فایل)
- ✓ سطحی که فایل‌های منطقی پایگاه داده‌ها تعریف می‌شود.
- ✓ تناظر بین «ساخت» های سطح ادراکی و «ساخت» های سطح داخلی

مثال: تناظر 1:1 بین ساخت‌های سطح ادارکی و سطح داخلی

Table	TableFile
STT	STTFile
COT	COTFile
STCOT	STCOTFile
…	…

دید داخلی (ادامه)

- توصیف دید داخلی ← **شمای داخلی (Internal Schema)** ← دستورهای { تعریف فایل‌ها / کنترل فایل‌ها }

تعریف: **نوعی برنامه که توسط خود DBMS(و گاه براساس اطلاعاتی که طراح – پیاده‌ساز به سیستم می‌دهد) تولید می شود و شرح و وصف فایلینگ منطقی پایگاه داده هاست.**

- **توجه:** در شمای داخلی انواع رکوردها تعریف می‌شوند و دستورهای لازم جهت ایجاد فایل‌ها و کنترل آنها در این شما وجود دارد.

مثال: شمای داخلی ساده شده در یک زبان شبه پاسکال

```
TYPE    STUDENT = RECORD
        STUDENT-ID      : String ;
        STUDENT-NAME    : String ;
        STUDENT-LEV     : String ;
        STUDENT-MJR     : String ;
        STUDENT-DEPT    : String ;
```

دید داخلی (ادامه)

- اطلاعاتی که طراح-پیاده ساز به سیستم می‌دهد (مانند شاخص) در دید داخلی تاثیر می‌گذارد.

- در سیستم‌های جدولی: خود سیستم روی کلید اصلی (PK) شاخص خودکار (Automatic Index) ایجاد می‌کند. (عمدتا B-Tree)

- برای ایجاد شاخص روی دیگر ستون ها پیاده ساز باید درخواست کند.

ایجاد شاخص بر روی ستون STNAME که PK نیست:

CREATE INDEX SNX

ON STT (STNAME)

[CLUSTERED] ? **خوشه‌بندی**

دید داخلی (ادامه)

ویژگی‌های ستون شاخص؟

- ✓ تغییر ناپذیر (حتی الامکان)
- ✓ پرکاربرد در کلاز WHERE
- ✓ ... ؟

حذف شاخص: SNX **INDEX** **DROP**

با اجرای دستور **DROP TABLE** / **DROP INDEX** در سیستم چه اتفاقی می‌افتد؟

مثالی از وضعیتی بیان کنید که براساس آن طراح-پیاده ساز تصمیم به ایجاد شاخص می‌گیرد.

دید منطقی DBMS نسبت به داده‌های ذخیره شده

- DBMS
  - **می‌داند**: جنبه های فایلینگ منطقی [مجازی]
    - ✓ چه فایل‌هایی دارد
    - ✓ نگاشت سطح ادراکی به سطح داخلی
    - ✓ صفحات (Pages) فضای پایگاه داده کاربر
    - ✓ فرمت رکورد هر فایل [رکورد داخلی]
    - ✓ ساختار هر فایل
    - ✓ کلید(ها)
    - ✓ استراتژی دست‌یابی به رکوردها
    - ✓ توالی منطقی رکوردها در صفحات
    - ✓ اندازه جاری هر فایل
    - ✓ اندازه گسترش فایل
    - ✓ اطلاعات همگانی
      - BOF
      - EOF
      - R/W
      - ⋮
    - ✓ ارتباط منطقی بین فایل‌ها
    - ✓ ....
  - **نمی‌داند**: جنبه های فایلینگ فیزیکی

دید منطقی DBMS نسبت به داده‌های ذخیره شده (ادامه)

- **DBMS**
  - **می‌داند**: جنبه های فایلینگ منطقی [مجازی] ....
  - **نمی‌داند**: جنبه های فایلینگ فیزیکی
    - چگونگی نشست فایل‌ها روی دیسک
    - استراتژی دستیابی (مثلا شاخص) چگونه پیاده‌سازی شده‌اند.
    - اندازه بلاک (Block) فیزیکی
    - جزئیات تکنیک‌های Blocking
    - Localityرکوردهای فایل‌ها
    - توالی منطقی رکوردها چگونه پیاده‌سازی شده‌اند
    - ...

Locality چیست و بر کدام یک از عملیات روی فایل‌ها تاثیر می‌گذارد؟

سطح فایلینگ مجازی چیست؟

فایلینگ مجازی

- در بعضی از سیستم‌های مدیریت جدید، سیستمِ مدیریت، کل فضای پایگاه داده را به صورت **مجموعه‌ای از مجموعه صفحات** می‌بینند، یعنی نوعی **نمای مجازی** از داده‌های ذخیره شده در پایگاه داده دارد.

مثال

در سطح فایلینگ مجازی $DB = \{\{pages\}\}$

نام جدول	تعداد صفحات	شماره صفحات
STT	10	p1 … p10
COT	15	p15 … p29
STCOT	900	P101 … P1000

SELECT STT.*

FROM STT

WHERE STID = ‘444’

DBMS : **READ** P1

(فرض کنید ‘444’ در P1 است)

دید خارجی

دید (نمای) خارجی

دید کاربر (برنامه ساز) خاص است نسبت به داده‌های ذخیره شده [مثلا دید یک AP نویس]

- **دید جزئی (Partial)**: دربرگیرنده نیازهای داده‌ای یک کاربر مشخص [برای یک AP مشخص]
- مطرح در سطح انتزاعی ⟵ مبتنی بر یک ساختار داده‌ای مشخص

آیا این ساختار داده همان ساختار داده سطح دید ادراکی است؟

- روی دید ادراکی طراحی و تعریف می‌شود.
- یک کاربر ⟵ چند دید متفاوت
  چند کاربر ⟵ یک دید مشترک

دید خارجی (ادامه)

✓ توصیف دید خارجی ⟵ شِمای خارجی

نوعی «برنامه»که کاربر سطح خارجی می‌نویسد، حاوی دستوارت «تعریف داده‌ها» و معدود دستورات «کنترل داده‌ها» ( **چرا معدود؟)**

✓ شِمای خارجی ⟵ **ذخیره در کاتالوگ**

در سیستم‌های جدولی، دید خارجی خود نوعی جدول است، اما **مجازی** (Virtual Table) و نه ذخیره‌شده

دید خارجی در واقع پنجره ای است که از آن کاربر خارجی محدوده‌ی داده‌ای خود را می‌بیند و نه بیشتر.

دید خارجی (ادامه)

کاربر ۱

V1

STID	STNAME
777	st7
444	st4
⋮	⋮

چند ستون از یک جدول

V2

CONUM	COTITLE

دگرنامی ستون

STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

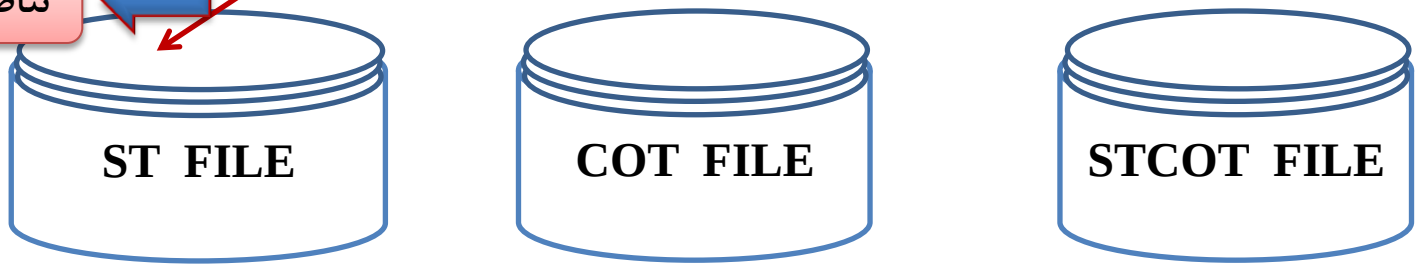
COT

COID	…

STCOT

STID	COID …

تناظر یک به یک



دید خارجی (ادامه)

کاربر ۲

V1

STID	STNAME
777	st7

دید مشترک با کاربر ۱

444	st4
⋮	⋮

STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

COT

COID	…
⋮	

STCOT

STID	COID …

دید خارجی (ادامه)

مثال

کاربر ۳

V1

STNUM	STNAME	COTITLE	TR	YR

دید روی بیش از یک جدول

STT

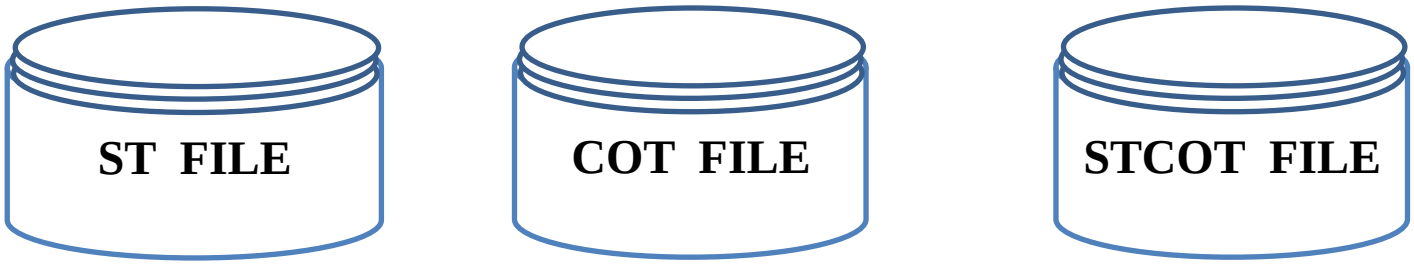
STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

COT

COID	…

STCOT

STID	COID …



دید خارجی (ادامه)

کاربر ۴

V2

STID	STNAM	TR	YR	AVG

صفت مجازی

STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

COT

COID	…

STCOT

STID	COID …

تعریف شمای خارجی

- از این مثال‌ها نتیجه می‌گیریم که تعریف، طراحی و توصیف دید خارجی در سیستم‌های جدولی از پویایی بالایی برخوردار است.
  - یعنی انواع جدول‌های مجازی را می‌توان روی لایه‌های زیرین تعریف کرد.
- **تعریف شمای خارجی کاربر ۱ (با استفاده ار مفهوم دید):**

```
CREATE  VIEW  V1 [(STID,  STNAME)]
        AS  SELECT  STT.STID,  STT.STNAE
                FROM  STT;

CREATE  VIEW  V2  [(SN, SJ, SL)]
        AS  SELECT  STID, STJ, STL
                FROM  STT
                WHERE  STJ  !=  ‘phys;
                [WITH  CHECK  OPTION]
```

شرط تعریف دید

- در شرط تعریف دید می‌توان از نام ستونی که در محدوده دید نیست استفاده کرد.

تعریف شمای خارجی (ادامه)

هر کاربر با اجازه Admin می‌تواند دید (View) خودش را داشته باشد (Sub-database).

دستور SELECT در متن دستور تعریف دید اجرایی نیست بلکه اعلانی است

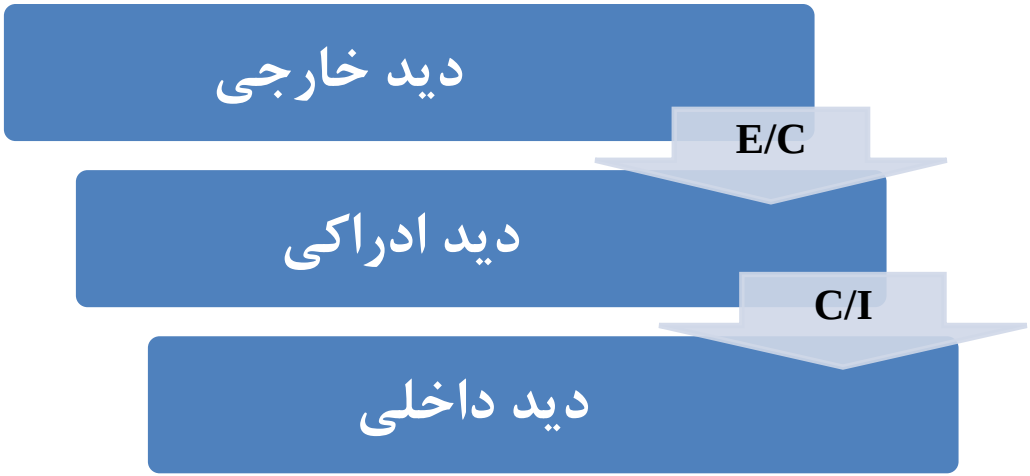
- یعنی هیچ داده‌ای بازیابی نمی‌شود و صرفا برای اعلام محدوده داده‌ای کاربران است.

تا آنجا که به تعریف دید مربوط است هر دستور SELECT معتبر با هر میزان پیچیدگی را می‌توان در CREATE VIEW نوشت.

- **تمرین:** مثال کاتالوگ پیش‌دیده را به نحوی گسترش دهید که اطلاعات (نه داده‌ها) شمای داخلی و شمای خارجی دیده شده را بتوان در آن ذخیره کرد (جدول دیگری برای کاتالوگ تعریف کنید که بتوان این شماها را در آن ذخیره کرد).

نگاشت بین سطوح

- نگاشت یا تبدیل بین سطوح (عملیات از دید خارجی در DB):
  - **E/C**: External to Conceptual Mapping
  - **C/I**: Conceptual to Internal Mapping



آیا تبدیل دیگری هم متصور است؟

عملیات در شمای خارجی

- عملیات در شمای خارجی
  - **بازیابی**: کاربر حق دارد در محدوده دید خود عمل بازیابی انجام دهد.
  - **ذخیره‌سازی**: به تشخیص Admin مجاز به انجام است.
    - درج
    - حذف
    - بروزرسانی

- هر دستور [حکم] عمل‌کننده در شمای خارجی (روی دید خارجی)،
  - تبدیل می‌شود به N>=0 دستور عمل کننده در شمای ادراکی (روی دید ادراکی)
  - و سپس به قطعه برنامه‌ای عمل کننده در شمای داخلی (روی دید داخلی)
  - و نهایتاً به عملیاتی در فایل‌های فیزیکی.

عملیات بازیابی از دید

عملیات بازیابی: چون دید خارجی در سیستم‌های جدولی، به هر حال نوعی جدول است، برای بازیابی از همان دستور SELECT استفاده می‌کنیم.

```
SELECT V2.SN
        FROM V2
        WHERE SL='ms'
```

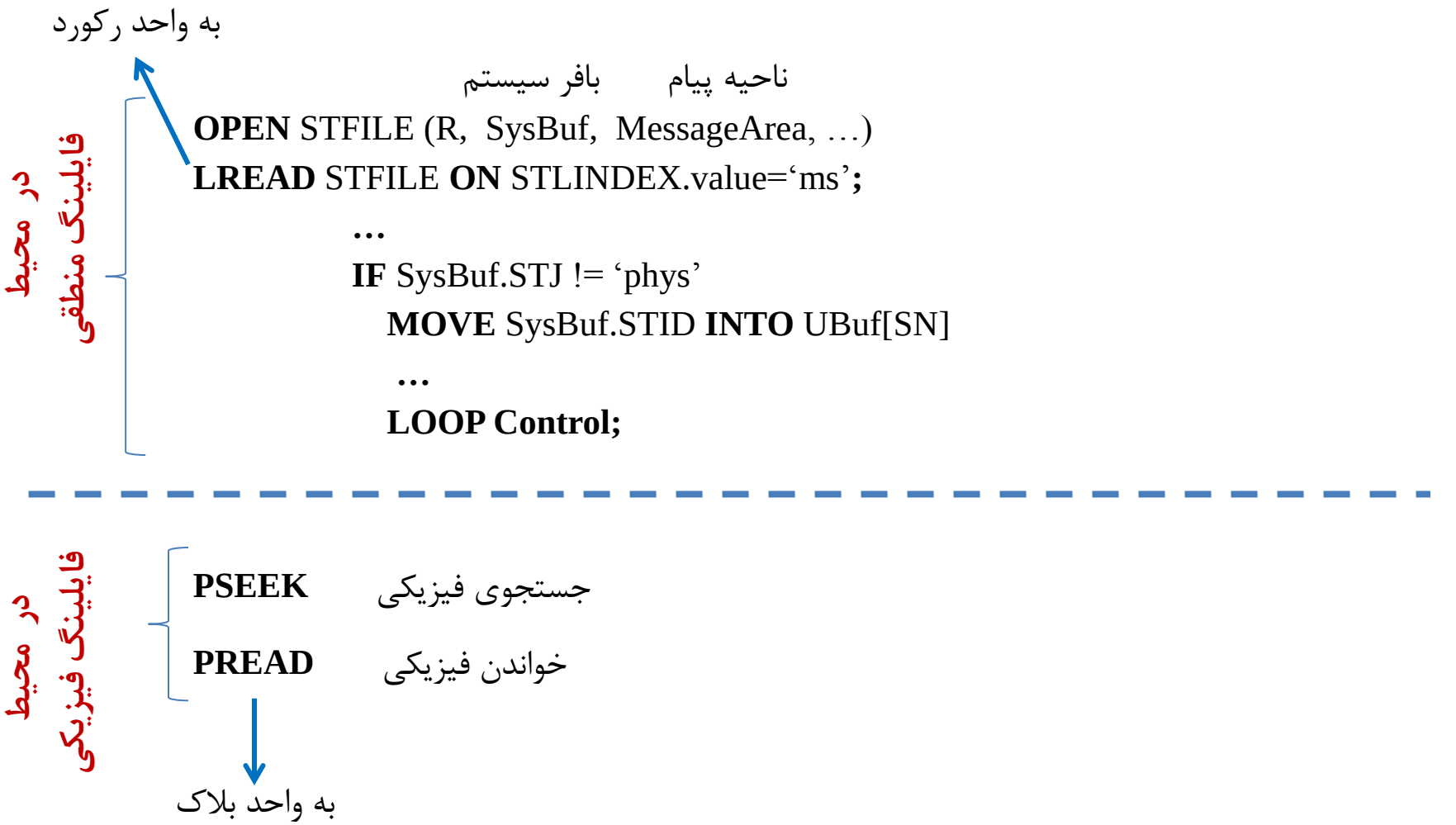
E/C

- سیستم در نگاشت E/C، شرط یا شرایط داده شده در تعریف دید را AND می‌کند با شرط یا شرایط داده شده در پرس‌وجوی روی دید. به این عمل، گاه **محاسبه دید** (View Computation) هم می‌گویند.

```
SELECT STT.STID
        FROM STT
        WHERE STL='ms'
          AND STJ != 'phys'
```

C/I

عملیات بازیابی از دید (ادامه)



عملیات ذخیره‌سازی از دید

- لزوماً از همه انواع دیدها نمی‌توان عملیات ذخیره‌سازی در DB انجام داد.
  - همه انواع دیدها قابل بروزرسانی (Updatable) نیستند.
  - محدودیتهایی هم در عمل و تاحدی در تئوری وجود دارد.

- **دید از نظر قابلیت عملیات ذخیره‌سازی** (بستگی دارد به ساختار دید و مکانیزم تعریف آن)
  - **پذیرا** (Updatable): می‌توان از آنها عملیات ذخیره‌سازی انجام داد ولی گاه مشکلاتی دارند.
  - **ناپذیرا** (Non Updatablen): تبدیل E/C انجام شدنی نیست.

- دید
  - تعریف شده روی یک جدول مبنا
  - تعریف شده روی بیش از یک جدول مبنا ← در عمل ناپذیرا، اما در تئوری بعضی‌ها پذیرا هستند.

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا

دید تعریف شده روی یک جدول مبنا

- دید دارای کلید جدول مبنا (Key Preserving) ← پذیرا (در عمل و تئوری) اما مشکلاتی هم دارد.
- دید فاقد کلید جدول مبنا (Non Key Preserving) ← ناپذیرا
- دید دارای ستون [صفت] مجازی (دیدهای آماری) ← ناپذیرا

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

- دید حافظ کلید تعریف شده روی یک جدول مبنا

V2

SN	SJ	SL
888	ms	math
444	bs	comp
⋮	⋮	

STT

STID	STNAME	STL	STJ	STD
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

```
CREATE VIEW V2 [(SN, SJ, SL)]
    AS SELECT STID, STJ, STL
        FROM STT
        WHERE STJ != 'phys'
        [WITH CHECK OPTION]
```

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

- فرض بر مجاز بودن کاربر به انجام عمل داریم و لذا صرفاً شدنی بودن را بررسی می‌کنیم.
- در **دید حافظ کلید** انجام عملیات سطری امکان‌پذیر است.
- زیرا تناظر یک به یک بین سطرهای دید و سطرهای جدول مبنا برقرار است.

حذف سطر در دید حافظ کلید

```
DELETE FROM V2
        WHERE SN='444'
```

E/C

```
DELETE FROM STT
        WHERE STID='444'  AND  STJ != 'phys'
```

- الان این سطر از جدول STT حذف می‌شود و اگر کاربر دیگری این سطر را در دیدش داشته باشد، دیگر به این سطر دسترسی ندارد.

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بروزرسانی در دید حافظ کلید

UPDATE V2
SET SJ=‘IT’
WHERE SN=‘444’

E/C

UPDATE STT
SET STJ=‘IT’
WHERE STID=‘444’ ’ **AND** STJ != ‘phys’

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

از نظر تئوریک درخواست زیر به دلیل عدم رعایت محدودیت دید باید رد شود.

```
UPDATE V2
        SET SJ='phys'
        WHERE SN='888'
```

- در عمل: اگر از عبارت [with check option] استفاده کنیم، سیستم رد می‌کند، وگرنه درخواست انجام می‌شود اما ...

E/C

```
UPDATE STT
        SET STJ='phys'
        WHERE STID='888' '  AND  STJ != 'phys'
```

- حال اگر بنویسیم:

```
SELECT V2.*  FROM V2
```

- سطر با کلید 888 دیگر در دید کاربر نمی‌آید!

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

INSERT INTO V2
VALUES (‘555’, ‘chem’, ‘bs’)

E/C

INSERT INTO STT
VALUES (‘555’, ?, ‘chem’, ‘bs’, ?)

- اگر هر کدام از ستون‌های نهان از دید کاربر، محدودیت هیچ‌مقدارناپذیری داشته باشند، درخواست رد می‌شود.
- حال اگر به جای 555 بنویسیم 777، درخواست رد می‌شود (تبدیل E/C انجام نمی‌شود) به دلیل عدم رعایت محدودیت یکتایی مقادیر کلید.
- حال اگر به جای chem بنویسیم phys، همان پیش می‌آید که در مثال UPDATE دیدیم.

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

- دلایل رد شدن درخواست عمل ذخیره‌سازی در دید تک جدولی حافظ کلید:
  - عدم رعایت محدودیت دید
  - عدم رعایت محدودیت یکتایی مقادیر کلید
  - عدم رعایت محدودیت هیچ‌مقدارناپذیری ستون‌های نهان
  - ...

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

دید تعریف شده روی یک جدول مبنا و فاقد کلید

چون این دید فاقد کلید است، امکان انجام عملیات سطری وجود ندارد.

```
CREATE VIEW V3
        AS SELECT STNAME, STJ
        FROM STT
```

- درخواست زیر انجام نمی‌شود، چون معلوم نیست کدام سطر از رابطه باید حذف شود. پس تبدیل E/C ناممکن است، مگر اینکه بپذیریم این درخواست به صورت مکانیکی انجام شود؛ یعنی تمام سطرهای حائز شرط داده شده (مجموعه‌ای از سطرها) حذف شوند.

```
DELETE FROM V3
        WHERE STNAME='ali' AND STJ='comp'
```

- اگر کاربر این پیامد را بپذیرد مشکلی نیست، اما در عمل سیستم‌ها نمی‌پذیرند!
- در دید V3 انجام INSERT نیز غیرممکن است.

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

مثال

حال اگر در تعریف V3، DISTINCT بزنیم چه پیش می‌آید؟

```
CREATE VIEW V3
        AS SELECT DISTINCT STNAME, STJ
        FROM STT
```

- فرقی نمی‌کند، باز هم همان مشکل پابرجاست:

```
DELETE FROM V3
        WHERE STNAME='a'
```

E/C

تبدیل می‌شود به حذف مجموعه‌ای از سطرها

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

- **دید تعریف شده روی یک جدول مبنا دارای ستون مجازی**
- این دیدها هم در عمل و هم در تئوری ناپذیرا هستند.

V4

PN	SQ
P1	100
P2	210
P3	80

```
CREATE VIEW V4 (PN, SQ)
        AS SELECT P#, SUM(QTY)
        FROM SP
        GROUP BY P#
```

SP

S#	P#	QTY
S1	P1	100
S1	P2	140
S2	P3	80
S2	P2	70

عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

- انجام عملیات سطری در دید V4 غیرممکن است.

DELETE FROM V4
WHERE PN='p1'

→ سطری نیست، با نوعی تفسیر می توان گفت که مجموعه‌ای از سطرها را حذف می‌کند.

- از لحاظ تئوریک هم دید V4 نباید پذیرا باشد.
- زیرا جدول V4 (که مجازی است) و جدول مبنای SP با هم تعارض معنایی (Semantic Conflict) دارند. یعنی مسند بیانگر معنای رابطه V4 اساساً با مسند بیانگر رابطه SP تفاوت دارد.

[دربحث رابطه‌ای خواهیم دید که هر رابطه (جدول) یک معنا دارد و در اینجا این دو رابطه هیچ ربطی از نظر معنایی با هم ندارند.]

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا

- **دیدهای تعریف شده روی بیش از یک جدول**
  - در عمل این دیدها ناپذیرا هستند و دخالت خود برنامه‌ساز لازم است.

V5: T1 **JOIN** T2 **دید پیوندی (پیوند طبیعی)**

V6: T1 **UNION** T2

V7: T1 **INTERSECT** T2

V8: T1 **EXCEPT** T2

- دید پیوندی
  - PK-PK: ستون پیوند در هر دو جدول PK است. پذیرا و بدون مشکل
  - PK-FK: پذیرا به شرط پذیرش پیامدها
  - FK-FK
  - NK-NK (Non-Key)

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

دید پیوندی PK-PK

V5 همان STT است اما این بار به صورت یک دید تعریف شده است.

V5

STID	STNAME	STL	STJ	STD
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

```
CREATE VIEW V5
    AS SELECT ST1.*, ST2.*
    FROM ST1  JOIN  ST2
```

ST1

STID	STNAME	STL
777	st7	bs
888	st8	ms
444	st4	bs
⋮	⋮	⋮

ST2

STID	STJ	STD
777	phys	d11
888	math	d12
444	comp	d14
⋮	⋮	⋮

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

- یک دستور اجراشونده در شمای خارجی تبدیل می‌شود به دو دستور در شمای ادراکی.

INSERT INTO V5
VALUES (‘999’, ‘St9’, ‘chem’, ‘bs’, ‘D15’)

E/C

INSERT INTO ST1
VALUES (‘999’, ‘St9’, ‘bs’)

INSERT INTO ST2
VALUES (‘999’, ‘chem’, ‘D15’)

- عمل DELETE در این دید تبدیل می‌شود به دو عمل حذف از جدول‌های مبنایی زیرین و عمل UPDATE (بسته به ستونی که می‌خواهیم بروز کنیم) به یک یا دو عمل بهنگام‌سازی در جدول‌های زیرین تبدیل می‌شود.

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

- دید پیوندی PK-FK


```
CREATE VIEW V6
        AS SELECT STT.*, STCOT.*
        FROM STT JOIN STCOT
```

- درج در این دید تبدیل می‌شود به درج یک تاپل ناقص در STT به شرط آنکه شماره دانشجویی تکراری نباشد. ولی در STCOT حتما یک تاپل درج می‌شود.

```
INSERT INTO V6
        VALUES ('9212345', 'Amir', '40638', 15)
```

E/C

```
INSERT INTO STT
        VALUES ('9212345', 'Amir', ?, ?, ?)

INSERT INTO STCOT
        VALUES ('9212345', '40638', 15)
```

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

- حذف از این دید مشکل دارد.
- اگر از هر دو جدول حذف شود، منجر به حذف داده‌های ناخواسته می‌شود
- با حذف یک سطر از جدول STT، برای حفظ جامعیت ارجاعی نیز لازم است یک تعداد سطر دیگر از STCOT حذف شود، مگر آنکه فقط از STCOT حذف کنیم و از STT سطر مربوطه را حذف نکنیم.

- عمل بهنگام‌سازی نیز مساله مشابه حذف ممکن است داشته باشد.

دید حاصل از پیوند FK-FK و دید حاصل از پیوند NK-NK چه رفتاری در عملیات ذخیره‌سازی دارند؟

عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

- **دید حاصل از اجتماع، اشتراک، و تفاضل**
- این دیدها از لحاظ تئوری مشکلی در عملیات ذخیره‌سازی ندارند، هرچند نظرات مختلفی مطرح است.

عمل / دید	درج	حذف	بهنگام‌سازی
$R_1 \cup R_2$	درج تاپل در R_1 و/یا R_2	حذف تاپل از R_1 و/یا R_2	بهنگام‌سازی تاپل در R_1 و/یا R_2
$R_1 \cap R_2$	درج تاپل در R_1 و R_2	حذف تاپل از R_1 و/یا R_2	بهنگام‌سازی تاپل در R_1 و R_2
$R_1 - R_2$	درج تاپل در R_1	حذف تاپل در R_1	بهنگام‌سازی تاپل در R_1

دیدهای پذیرا در SQL

- موضوع دیدهای پذیرا در SQL استاندارد چندان روشن نیست. در SQL 2003 دیدهایی که تمام شرایط زیر را داشته باشند، قابل بهنگام‌سازی (درج، حذف و بروزرسانی) هستند.

[توجه: ممکن است برخی دیگر از دیدها هم قابل بهنگام‌سازی باشند.]

۱- عبارت تعریف‌کننده دید، یک عبارت SELECT ساده باشد (یعنی شامل عملگرهای JOIN، UNION، INTERSECT و EXCEPT نباشد).

۲- در عبارت SELECT گزینه DISTINCT وجود نداشته باشد.

۳- در کلاز FROM عبارت SELECT، فقط یک جدول وجود داشته باشد.

۴- جدول قید شده در کلاز FROM، یک جدول مبنا یا یک دید قابل بهنگام‌سازی باشد.

دیدهای پذیرا در SQL (ادامه)

۵- در لیست نام ستون‌ها در عبارت SELECT، ستون‌های موردنظر باید در جدول مبنا متناظر داشته باشند و به یک ستون از جدول مبنا بیش از یک بار ارجاع وجود نداشته باشد. ضمناً حاوی ستون کلید باشد.

۶- در عبارت SELECT، کلاز GROUP BY و/یا کلاز HAVING وجود نداشته باشد.

۷- کلاز WHERE در عبارت SELECT حاوی کلاز FROM نباشد به گونه‌ای که در آن به همان جدولی ارجاع داده شده باشد که در کلاز FROM ذکر شده در شرط ۴.

- نتیجه اینکه عملاً دیدهایی که یک زیرمجموعه افقی-عمودی دارای کلید از یک جدول مبنا (یا از دید قابل بهنگام‌سازی) باشند، قطعاً قابل بهنگام‌سازی هستند.

[**توجه:** به شرط رعایت محدودیت‌های جامعیتی مانند یکتایی کلید و هیچمقدارناپذیری]

مزایا و معایب مفهوم دید خارجی

- **معایب مفهوم دید:**
  - محدودیت [و مشکلات] در عملیات ذخیره‌سازی
  - فزونکاری (overhead) برای انجام تبدیل E/C (محاسبه دید) . راه حل: *استفاده از تکنیک دید ذخیره شده*

مزایا و معایب مفهوم دید خارجی (ادامه)

- **مزایای مفهوم دید خارجی:**
  - فراهم‌کننده محیط انتزاعی فرافایلی برای کاربران با پویایی بالا
  - اشتراک داده‌ها (Data Sharing) ⟵ داده‌ها یک بار ذخیره می‌شوند و کاربران بسته به نیاز خود از داده‌های ذخیره شده به صورت همروند استفاده می‌کنند.
  - تامین امنیت برای داده‌های زیرین. ⟵ از طریق مفهوم داده مخفی (Hidden Data)، زیرا کاربر خارج از محدوده دید خود هیچ نمی‌بیند (داده‌های نهان امن هستند).
  - تامین‌کننده استقلال داده‌ای (مفهوم اساسی در تکنولوژی DB؛ هم مزیت و هم از اهداف مهم تکنولوژی DB).
  - امکانی است برای کوتاه‌نویسی یا ماکرونویسی پرسش‌ها.

دید ذخیره شده

- **تکنیک دید ذخیره شده [ساخته شده] (Materialized View)**
  - در این تکنیک، دید در سیستم ذخیره می‌شود؛ یعنی دیگر مجازی نیست و جدول ذخیره شده است. تا در هر بار مراجعه به دید لازم نباشد تبدیل E/C انجام شود.
  - **هدف:** برای افزایش سرعت عملیات بازیابی.
  - **شرط استفاده:** در عمل از این تکنیک وقتی استفاده می‌کنیم که داده‌های ذخیره شده در جدول‌های مبنای زیرین حتی‌الامکان تغییر نکنند. به بیان دیگر، نرخ عملیات ذخیره‌سازی در جدول‌های زیرین پایین باشد. زیرا اگر جدول‌های زیرین تغییر کنند، تغییرات متناسباً در جدول‌های دید باید اعمال شوند و این خود سربار ایجاد می‌کند.
  - **کاربرد:** در برنامه‌های آماری، گزارش‌گیری‌ها و برنامه‌های داده‌کاوی (Data Mining)
  - دید ذخیره شده (Stored View) در SQL چگونه پیاده‌سازی می‌شود؟ با CREATE SNAPSHOT.

دلایل عدم استفاده از دید

- چه زمانی از مفهوم دید استفاده نمی‌کنیم؟
  - هنگامی که سیستم تک‌کاربره باشد.
  - هنگامی که به تشخیص admin برای افزایش کارایی سیستم، برخی برنامه‌ها را مستقیماً روی شمای ادراکی (جداول مبنایی) بنویسیم.
  - هنگامی که کاربر نیازمند انجام عملیات ذخیره‌سازی باشد و از طریق دید امکان آن وجود نداشته باشد.

استقلال داده‌ای

- **مفهوم استقلال داده‌ای [DI] (جدایی برنامه‌ها از داده‌ها):**
  - مصونیت (تاثیرناپذیری) برنامه‌های کاربران [در سطح خارجی] در قبال تغییرات در سطوح زیرین معماری DB.

- استقلال داده‌ای (DI)
  - استقلال داده‌ای فیزیکی (PDI)
  - استقلال داده‌ای منطقی (LDI)

- چرا نباید برنامه‌ها تغییر کنند؟
  - چون هر تغییر در برنامه‌ها، هزینه تولید و پشتیبانی و بازتولید برنامه‌ها را بالا می‌برد.

استقلال داده‌ای فیزیکی

- **استقلال داده‌ای فیزیکی (PDI)**
  - مصونیت برنامه‌های کاربران در قبال تغییرات در شمای داخلی DB
  - تغییرات در شمای داخلی شامل تغییر در جنبه‌های فایلینگ پایگاه
    - ساختار فایل، طول رکورد، طرز ذخیره‌سازی فایل روی دیسک، گاه با دخالت طراح فیزیکی و گاه فقط توسط DBMS.
  - زیرا کاربران با مفهوم دید کار می‌کنند که اساساً در سطح فرافایلی مطرح است و برنامه‌ها درگیر جنبه‌های فایلینگ نیستند.

استقلال داده‌ای منطقی

- **استقلال داده‌ای منطقی (LDI)**
  - مصونیت برنامه‌های کاربران در قبال تغییرات در شمای ادراکی DB.
  - در سیستم‌های پایگاهی جدید تا حد زیادی این استقلال تامین است ولی نه صددرصد.
- **تغییر در شمای ادراکی**
  - رشد پایگاه داده‌ها (DB Growth)
  - تغییر سازمان پایگاه داده‌ها [سازماندهی مجدد DB] (DB Restructuring)
- **نکته:** نغییراتی که مورد بررسی قرار می‌دهیم، تغییراتی است که از داده‌ها و ساختار موجود نمی‌کاهد، چرا که تغییرات کاهشی، قطعاً بر روی برنامه‌های سطح خارجی تاثیر می‌گذارد و استقلال داده‌ای حفظ نمی‌شود.

استقلال داده‌ای منطقی (ادامه)

- **چرا رشد DB:** مطرح شدن نیازهای جدید
  - اضافه شدن ستون(های) جدید به جدول(ها)
  - ایجاد جدول‌های جدید
- استقلال داده‌ای منطقی (LDI) در قبال رشد DB، به کمک مفهوم دید تقریباً صددرصد تامین است، زیرا کاربرِ دارای یک دید، خارج از محدوده آن دید هیچ نمی‌بیند.

استقلال داده‌ای منطقی (ادامه)

مثال

دیدهای پیش‌تر تعریف شده را روی جدول STT در نظر می‌گیریم. حال نیاز جدیدی برای کاربر مطرح شده است.

کاربر ۱

V1

STID	STNAME
⋮	⋮

V2

⋮

کاربر ۲

V9

STID	STADR
⋮	⋮

STT

STID	STNAME	STL	STJ	STD	STADR
777	st7	bs	phys	d11	
888	st8	ms	math	d12	
444	st4	bs	comp	d14	
⋮	⋮	⋮	⋮	⋮	

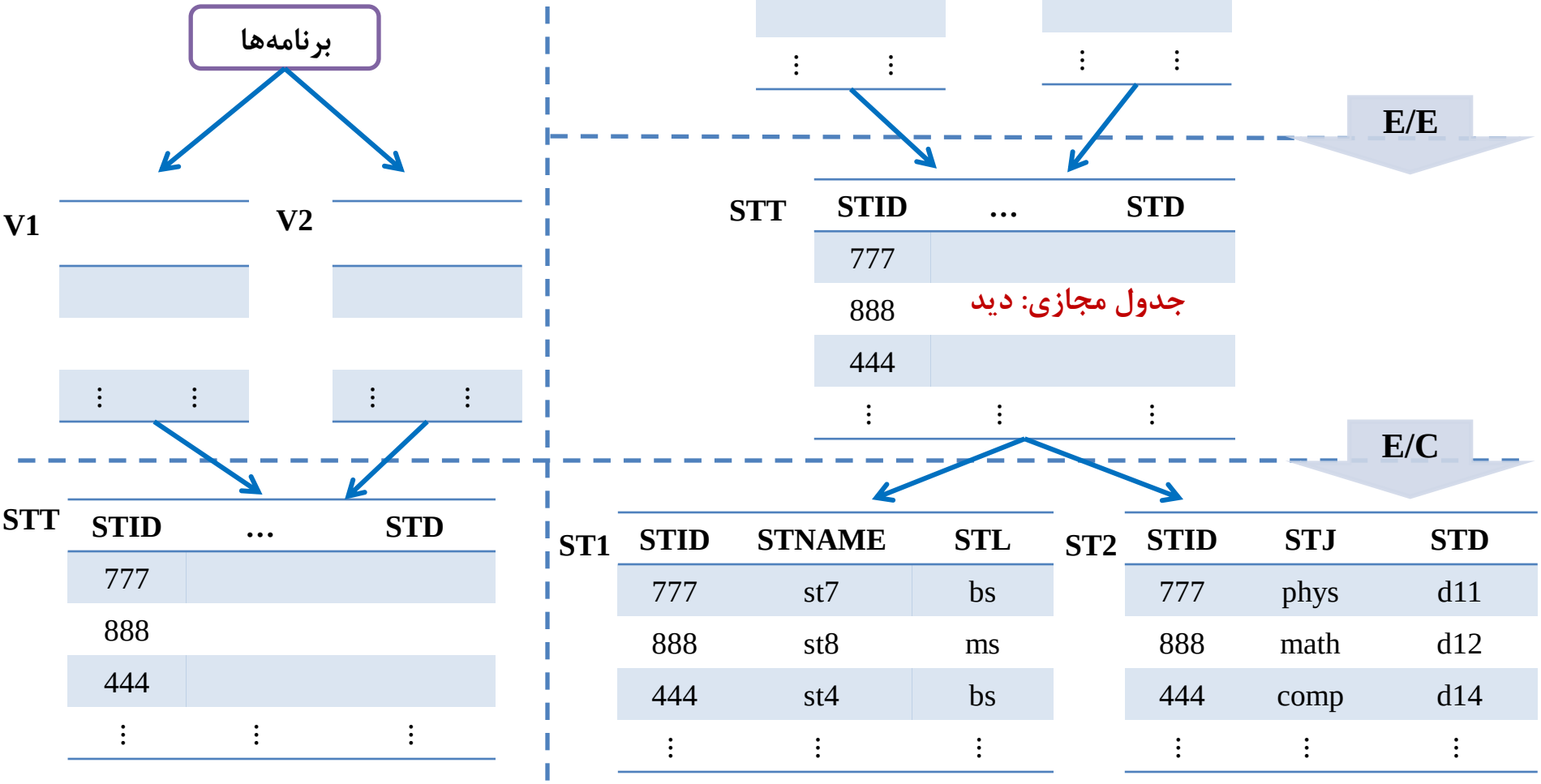
ALTER TABLE STT
ADD COLUMN STADR CHAR(70)

- این گسترش در سطح فایلینگ چگونه انجام می‌شود؟

استقلال داده‌ای منطقی (ادامه)

- آیا پیرو نیاز جدید کاربر در حد ستون، طراح همیشه جدول مبنا را گسترش می‌دهد؟
  - خیر، زیرا ممکن است آن ستون مجازی (محاسبه شدنی) باشد.
- **سازماندهی مجدد DB** یعنی طراح به هر دلیلی طراحی منطقی DB را تغییر دهد. مثلاً یک جدول مبنای موجود را به دو جدول تجزیه عمودی کند و طبعاً شمای ادراکی هم تغییر می‌کند. می‌خواهیم ببینیم LDI در قبال این تغییر تا چه حد تامین است.
- در این حالت، LDI به کمک مفهوم دید و امکان تعریف دید روی دید (View Definition on View)، تا حدی تامین است.

استقلال داده‌ای منطقی (ادامه)



استقلال داده‌ای منطقی (ادامه)

- شمای جدید:

```
CREATE TABLE ST1
        (STID …,
        …
        STL …)
        PRIMARY KEY STID

CREATE TABLE ST2
        (STID …,
        …
        STD …)
        PRIMARY KEY STID

INSERT INTO ST1
        (SELECT STID, STNAME, STL
        FROM STT)

INSERT INTO ST2
        (SELECT STID, …, STD
        FROM STT)
```

مهاجرت داده‌ها (Data Migration)

استقلال داده‌ای منطقی (ادامه)

- با حذف جدول مبنای STT، دیدهای قبلاً تعریف شده روی آن نامعتبر می‌شوند و در نتیجه برنامه‌هایی که روی آنها کار می‌کردند، دیگر اجرا نمی‌شوند و LDI دیگر تامین نیست مگر اینکه طراح و پیاده‌ساز تدبیری بیندیشد.

↓

- ✓ جدول STT را با همان نام و ساختار به شکل یک دید تعریف می‌کنیم، با مکانیزم پیوند (**دید روی دید**):

```
CREATE VIEW STT
        AS SELECT STID, …, STD
        FROM  ST1 JOIN ST2

DROP TABLE STT
```

- تعریف این دید وارد کاتالوگ سیستم می‌شود. ← دیدهای قبلاً تعریف شده معتبر می‌شوند.

استقلال داده‌ای منطقی (ادامه)

- با این تدبیر، LDI برای برنامه‌هایی که بازیابی انجام می‌دهند، صددرصد تامین می‌شود، به قیمت افزایش سربار برای انجام تبدیل E/E علاوه بر E/C و C/I. زیرا از تکنیک **دید روی دید** استفاده کرده‌ایم.

- اما LDI برای برنامه‌هایی که عملیات ذخیره‌سازی انجام می‌دادند، ممکن است تامین نباشد. زیرا این بار STT خود یک دید است و دیدها در عملیات ذخیره‌سازی عمدتاً مشکل دارند. ولی در این مثال خاص از نظر تئوریک مشکلی بروز نمی‌کند. ⟵ چون STT یک دید پیوندی PK-PK است.

استقلال داده‌ای منطقی (ادامه)

- دلایل این نوع تجزیه (که کلید در هر دو جدول باشد) چه می‌تواند باشد؟
- افزایش کارایی سیستم در رده فایلینگ با فرض 1-Table:1-File برای بعض برنامه‌ها (مثلاً برنامه‌هایی با فرکانس بالاتری نسبت به ستون‌های ST1 و با فرکانس پایین‌تری به ستون‌های ST2 ارجاع داشته باشد، فایل‌ها را جدا می‌کند).
- توزیع داده‌ها در سایت‌ها وقتی پایگاه داده توزیع شده (DDB) داشته باشیم.
- کاهش حجم Null Value
- بهینه‌سازی طراحی (رجوع شود به بحث نرمال‌سازی رابطه‌ها)
- ...

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش پنجم:
مفاهیم اساسی مدل داده رابطه‌ای

مرتضی امینی

نیمسال اول ۹۲-۹۳

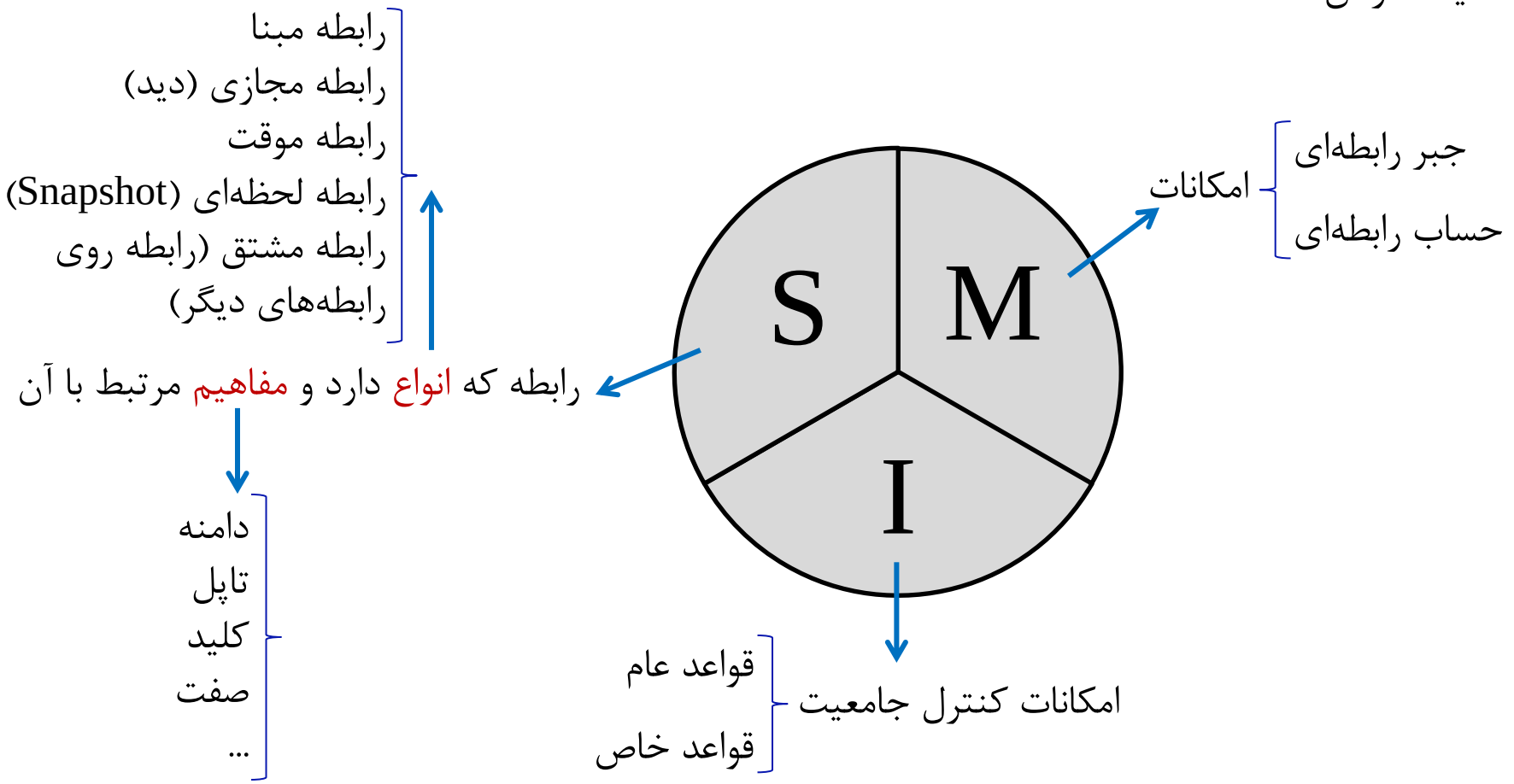
(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)

مقدمه

- RDM مبنای تئوریک RDB و RDBMS
- واضع مدل: F. Codd
- مفاهیم زیر در طی سه بخش باقیمانده از این درس مرور می‌شوند:
  - رابطه (Relation)
  - دامنه (میدان)
  - رابطه نرمال و غیرنرمال
  - کلید در مدل رابطه‌ای
  - قواعد جامعیت رابطه‌ای
  - عملیات در RDB ← جبر رابطه‌ای / حساب رابطه‌ای
  - طراحی RDB ← روش بالا به پایین / روش نرمال‌ترسازی (سنتز)

مدل داده‌ای

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



رابطه

در ریاضی: هر زیر مجموعه از ضرب کارتزین چند مجموعه

(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)

- مجموعه عنوان را با H_R یا R(H) نیز نمایش می‌دهیم. به R(H)، Intention (ذات، جوهر یا **چکیده**) رابطه هم گفته می‌شود.
  - R(H) ثابت در زمان است. یعنی اگر مجموعه صفات را عوض کنیم، از نظر ریاضی یک رابطه دیگر است.
  - همین R(H) برای تعریف رابطه در سیستم کافی است.

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

- هررابطه یک معنا دارد، بیانگر واقعیتی از یک محیط مشخص. به عنوان مثال وقتی می‌گوییم رابطه STUD را داریم، معنایش این است که در خردجهان واقع، نوع موجودیتی با نام STUD و با صفات STID و STNAME و ... و STD وجود دارد.

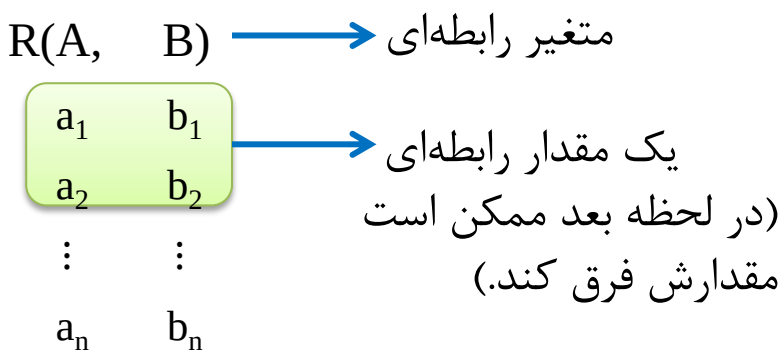
رابطه (ادامه)

- **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.

رابطه (ادامه)

تعریف (۳) [از نظر تئوری زبان‌های برنامه‌سازی] تشکیل شده است از یک متغیر رابطه‌ای و در هر لحظه از یک مقدار رابطه‌ای.

- R(H): متغیر رابطه‌ای، متغیری از جنس رابطه Relation Variable [RELVAR]
- بدنه (r): مقدار رابطه‌ای Relation Value [RELVAL]



مدل رابطه‌ای و مدل جدولی

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.)

مدل رابطه‌ای و مدل جدولی (ادامه)

ویژگی‌های رابطه:

۱- صفات در عنوان رابطه نظم (مکانی) ندارند. [چون مجموعه است] $R(A, B) = R(B, A)$

در حالی که در جدول، ستون‌ها می‌توانند نظم مکانی داشته باشند.

در مدل رابطه‌ای، تنها راه ارجاع به صفت رابطه، نام صفت است.

۲- تاپل‌ها [در بدنه] نظم ندارند (مرتب نیستند) [چون مجموعه است].

۳- رابطه، تاپل تکراری ندارد [چون مجموعه است].

۴- تمام صفات رابطه، تک مقدار هستند [رجوع شود به مفهوم رابطه نرمال] (این ویژگی دلیل تکنیکی دارد و از ذات رابطه نتیجه نمی‌شود). یعنی در هر تاپل دقیقاً یک مقدار برای هر صفت وجود دارد.

- در RM هیچ یک از مفاهیم فایلینگ مطرح نیستند (مثل نظم، فیلد، رکورد، اشاره‌گر، آدرس که در سطح طراحی و فایلینگ فیزیکی مطرح است).

مدل رابطه‌ای و مدل جدولی (ادامه)

تفاوت‌های مفهوم رابطه و اصطلاح جدول

- ۳ ویژگی اول رابطه، ۳ تفاوت

۴- در رابطه m>=0 (درجه)، یعنی از نظر تئوری رابطه می‌تواند از نظر درجه صفر باشد.

۵- رابطه می‌تواند بیش از دو بُعد داشته باشد (مثلا Data Cube).

۶- نمایش دقیق عنوان رابطه به صورت زیر است حال آنکه عنوان جدول چنین نیست.

عنوان رابطه مجموعه‌ای است از دوتایی‌ها منظم دامنه، صفت $R(H): \{\langle D_1: A_1\rangle, \langle D_2: A_2\rangle, \ldots\}$

۷- نمایش دقیق تاپل رابطه به صورت زیر است حال آنکه سطر در جدول چنین نیست.

تاپل مجموعه‌ای است از سه‌تایی‌های منظم دامنه، صفت، مقدار $TUPLE: \{\langle D_1: A_1: V_1\rangle, \langle D_2: A_2: V_2\rangle, \ldots\}$

۸- رابطه نمی‌تواند هیچ‌مقدار داشته باشد، ولی جدول می‌تواند.

دامنه [میدان]

مفهوم دامنه (میدان)

- مجموعه‌ای است نامدار از مقادیر هم نوع، که حداقل یک صفت از رابطه، از آن **معنا**، **نوع** و **مقدار** می‌گیرد.
- معادل است با مفهوم Data Type در تئوری انواع.
- دامنه‌هایی که یک رابطه روی آن‌ها تعریف می‌شود، لزوماً متمایز نیستند.

مفروض $R(H)$

if $A_i \in H,\ A_j \in H,\ A_i \neq A_j \Rightarrow (D_i \neq D_j$ که لزوما چنین نیست$)$

دامنه [میدان] (ادامه)

- **تمرین:** مثالی از یک رابطه ۵-تایی که
  - دوصفت آن از یک دامنه باشد.
  - سه صفت آن از یک دامنه باشد.
- اگر m درجه رابطه و n تعداد دامنه‌ها باشد، داریم: $n \leq m$.
- برای تعریف یک رابطه در سیستم رابطه‌ای، از لحاظ تئوریک، ابتدا باید دامنه‌هایش را تعریف کرد.

دامنه [میدان] (ادامه)

مثالی از شمای پایگاه رابطه‌ای (در مدل تئوریک)

```
CREATE DOMAIN  SN       CHAR(8)  DEFAULT '00000000'
CREATE DOMAIN  SNAME    CHAR(20) DEFAULT 'noname'
CREATE DOMAIN  SJ       CHAR(4)  DEFAULT '?...?'
CREATE DOMAIN  SL       CHAR(3)  DEFAULT '?...?'
CREATE DOMAIN SD        CHAR(4)  DEFAULT '?...?'
CREATE DOMAIN CN        CHAR(6)  DEFAULT '?...?'
CREATE DOMAIN GRADE     DEC(2, 2) DEFAULT '?...?'

CREATE RELATEION STUD
        (STID  DOMAIN  SN,
         STNAME  DOMAIN  SNAME,
         STJ  DOMAIN  SJ,
         STL  DOMAIN  STL,
         STD  DOMAIN  SD)

CREATE RELATION COUR ….

CREATE RELATION SCR …
```

دامنه [میدان] (ادامه)

دستورات زیر در SQL مطالعه شود.

- CREATE DOMAIN
- ALTER DOMAIN
- DROP DOMAIN

مزایای مفهوم دامنه از دیدگاه مهندسی نرم‌افزار بررسی شود.

رابطه نرمال و غیرنرمال

- **رابطه نرمال (بهنجار- عادی Flat Relation):**

رابطه‌ای که تمام صفات آن تک‌مقداری (حداکثر دارای یک مقدار در هر تاپل) باشند.

- **رابطه غیرنرمال (Nested Relation):**

رابطه‌ای که حداقل یک صفت آن چندمقداری باشد.

- **توجه:** تعریف زیر درست **نیست:**
  - رابطه‌ای نرمال است که مقادیر تمام صفات آن اتمیک (تجزیه نشدنی) باشند.
  - **تذکر:** ساده یا مرکّب بودن صفت نقشی در نرمال بودن و نبودن آن ندارد.

رابطه نرمال و غیرنرمال (ادامه)

مثال

صفت چندمقداری ساده

NNCOPRECO (COID , PRECOID)

COID	PRECOID
c01	{c11, c17, c08}
c02	{c03, c09}
c03	c10

یک تاپل

تبدیل به

رابطه نرمال

COPRECO (COID , PRECOID)

COID	PRECOID
c01	c11
c01	c17
c01	c08
c02	c03
c02	c09
c03	c10

یک تاپل

رابطه نرمال و غیرنرمال (ادامه)

مثال

صفت چندمقداری مرکب
P# , QTY

NNSP (S# , PQTY)

SP (S# , P# , QTY)

S#	PQTY
s1	p1 100 p2 90 p3 50
s2	p1 60 p2 90
s3	p1 150

یک تاپل

تبدیل به رابطه نرمال

S#	P#	QTY
s1	p1	100
s1	p2	90
s1	p3	50
s2	p1	60
s2	p2	90
s3	p1	150

یک تاپل

رابطه نرمال و غیرنرمال (ادامه)

- **دلیل نرمال بودن رابطه در RM:**
  - سادگی در
    - ۱- نمایش ظاهری رابطه
    - ۲- دستورات
      - DSL
        - DDL
        - DML
        - DCL
    - ۳- اجرای دستورات

مثال برای درک موارد ۲ و ۳

- درج کن
  - I_1 : $< s4 , p4 , 40 >$: در هر دو رابطه NNSP و SP منجر می شود به درج «تاپل در رابطه» با همان دستور ساده «درج کن تاپل را».
  - I_2 : $< s2 , p3 , 30 >$: با همان دستور ساده درج می شود در SP و نه NNSP .

رابطه نرمال و غیرنرمال (ادامه)

- ادامه مثال

I_1 : **INSERT INTO** $\begin{Bmatrix} \text{NNSP} \\ \text{SP} \end{Bmatrix}$

TUPLE (S4 , P4 , 40**);**

I_2 : **INSERT INTO** SP

TUPLE (S2 , P3 , 30**);**

→ **امکان پذیر**

I_2 : **INSERT INTO** NNSP

TUPLE (S2 , P3 , 30**);**

→ **امکان ناپذیر**

- دلیل: تاپلی با کلید S2 وجود دارد.

برای درج I_2 در NNSP منطقا چه باید کرد؟

✓ در رابطه غیرنرمال دستورات ساده‌ی تاپلی کار نمی‌کنند.

مزایا و معایب رابطه نرمال و غیرنرمال

نوع رابطه	مزایا	معایب
نرمال	سادگی (۱- ... ۲-... ۳-...)	طولانی شدن کلید
	تقارن صفات (پیاده‌سازی در سطح فایلینگ ساده‌تر) (نقش تمام صفات در عبارت WHERE وقتی که شرط جستجو را با theta می‌دهیم، یکسان است، زیرا همه تک‌مقداری‌اند. SELECT…. FROM …. WHERE A<(=)(>) ‘Single Value’ چنین تقارنی در رابطه غیرنرمال وجود ندارد.)	افزونگی (ادراکی یا منطقی) (این نوع افزونگی که در مرحله طراحی پیدا شده ممکن است منجر به افزونگی فیزیکی بشود یا نشود؛ بستگی دارد به نحوه پیاده‌سازی رابطه در سطح فایلینگ. اگر تناظر یک به یک باشد، که هر تاپل هم با یک رکورد پیاده‌سازی شود، افزونگی فیزیکی نیز پیش می‌آید.)
		دشواری در نمایش طبیعی ارتباط سلسله مراتبی بین اشیاء
		دشواری در نمایش مفهوم وراثت
		کاهش سرعت بازیابی در بعضی از پرسش‌ها
		سنگین و زمانگیر کردن کار طراحی منطقی پایگاه داده‌ها
غیرنرمال	[عکس معایب رابطه نرمال]	پیچیدگی (۱- ... ۲-... ۳-...)
		عدم تقارن صفات

مزایا و معایب رابطه نرمال و غیرنرمال (ادامه)

- **در عمل با کلید طولانی چه باید کرد؟**

- از یک کلید ساختگی استفاده می‌کنیم؛ یعنی یا خودمان به صورت دستی و یا خود سیستم به صورت خودکار به هر سطر یک شماره می‌دهد.

این تکنیک چه مزایا و چه معایبی دارد؟

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
STUD	STID
COUR	COID
SCR	(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 CK

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

EMPROJ (E#, J#, ENC, …)

CK CK

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 فرهم می‌نماید.
- هر زبرمجموعه 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 اصلی (ادامه)

- **دلایل لزوم انتخاب کلید اصلی:**

۱- دلیل تاریخی: PK مفهوم آشناتر برای طراحان است.

۲- ایجاد شاخص اتوماتیک روی PK.

۳- در بحث جامعیت DB: چون محدودیت هیچ‌مقدارناپذیری را اگر به همه CKها بدهیم خیلی محدود کننده است. یکی که این محدودیت را روی آن اعمال می‌کنند می‌شود PK.

- اصالت مفهومی در مدل رابطه‌ای با کلید کاندید (CK) است.

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
SCR	STID	STUD
SCR	COID	COUR
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 صریحی است، ارتباطی است که در مدل ER با لوزی مشخص شده است.

مثال

S (S#, …) — CK: S#

P (P#, …) — CK: P#

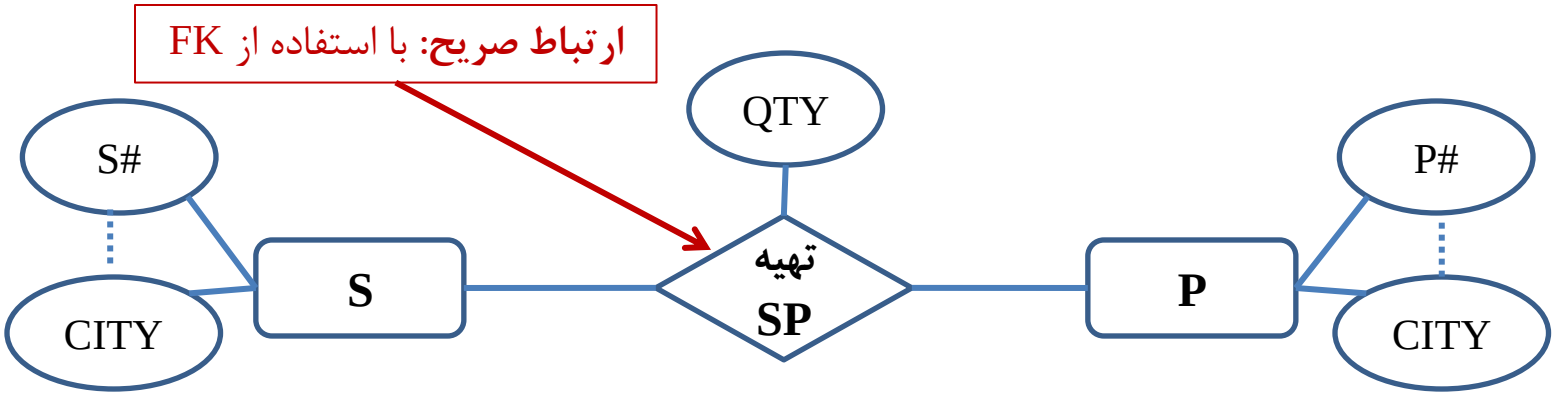
SP (S#, P#, …) — FK: S#, FK: P#; CK: (S#, P#)

SCR (STID, COID, …) — FK: STID, FK: COID; CK: (STID, COID)

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



S (S#, … , CITY)

P (P#, … , CITY)

SP (S#, P#, …)

ارجاع

ارتباط ضمنی: از طریق هر صفت مشترک؛ صفت هم‌معنا (از یک میدان) و نه لزوماً هم‌نام

بحث تکمیلی: کلید خارجی – گراف ارجاع

مفهوم گراف ارجاع

- FK امکانی است برای ارجاع از یک رابطه به رابطه‌ای دیگر
- هر مقدار معلوم FK، امکانی است برای ارجاع مقداری از تاپل(هایی) از رابطه(هایی) به تاپلی از رابطه(هایی).

تعریف **گراف ارجاع** امکانی است برای نمایش ارجاعات بین رابطه‌ها.

مثال

$$P \longleftarrow SP \longrightarrow S$$

- **شکل کلی مسیر ارجاع:**

$$R_m \rightarrow R_{m-1} \rightarrow \cdots \rightarrow R_2 \rightarrow R_1$$

با این ارجاع می شود **چرخه ارجاع**

- مسیر ارجاع می‌تواند **چرخه‌ای** باشد.

گراف ارجاع (ادامه)

- چرخه ارجاع می‌تواند تک‌رابطه‌ای باشد و این در صورتی است که یک رابطه خود ارجاع (Self-Referencing) داشته باشیم.
- هنگامی که FK تعریف می‌کنیم باید معنایش را نیز بگوییم.

مثال

چرخه ارجاع بین دو رابطه کارمند و اداره.

شماره کارمند مدیر اداره

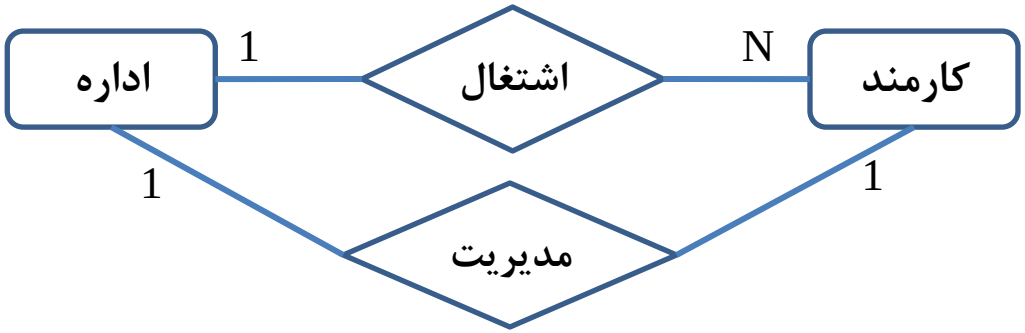
DEPT (D#, DTITLE, … , E#)

DEPT ⇄ EMPL

شماره اداره محل کار

EMPL (E#, ENAME, …, D#)

- بر اساس کدام مدلسازی این طراحی انجام شده است؟



گراف ارجاع (ادامه)

چرخه ارجاع تک‌رابطه‌ای کارمند با خودش.

شماره مدیر

EMPL (E#, ENAME, ENC, …, EPHONE, EMANAGER#) **EMPL**

- **نکته‌های مثال اخیر:**
  - مثالی است از حالتی که در آن R1 و R2 در تعریف FK، لزوماً متمایز نیستند.
  - رابطه EMPL به خود رجوع کننده (خود ارجاع) است.
  - اگر m درجه EMPL باشد و n تعداد دامنه‌هایش باشد، داریم: $n \leq m\text{-}1$
  - لزوم دگر نامی شماره کارمندی مدیر، چون عنوان رابطه (Heading)، مجموعه‌ای از نام صفات است.
  - **تمرین:** این طراحی بر اساس کدام مدلسازی انجام شده است؟

گراف ارجاع (ادامه)

مثال چرخه ارجاع سه رابطه‌ای

دانشکده استاد

PROF (PRID, PRNAME, …, DEID)

DEPT(DEID, DTITLE, …., UNID)

UNIV(UNID, UNAME, …, UNPRESNUM)

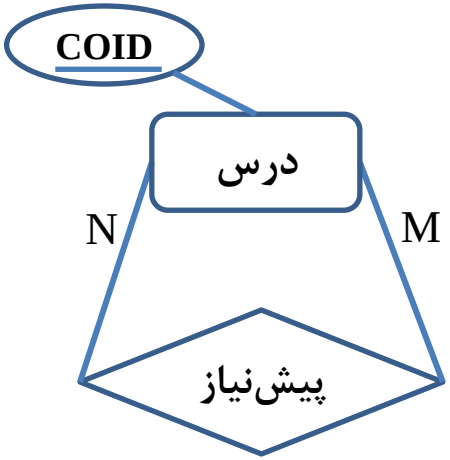
شماره استادی رئیس دانشگاه

تمرین: این طراحی بر اساس کدام مدلسازی انجام شده است؟

گراف ارجاع (ادامه)

نکته صرف وجود دور در ERD چرخه ارجاع ایجاد نمی‌شود.

مثال



COUR (COID, …)

COPRECO(COID, PRECO)

- **در چه وضعی چرخه ارجاع پدید می‌آید؟**
- باید به چندی ارتباط‌ها توجه شود.

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.

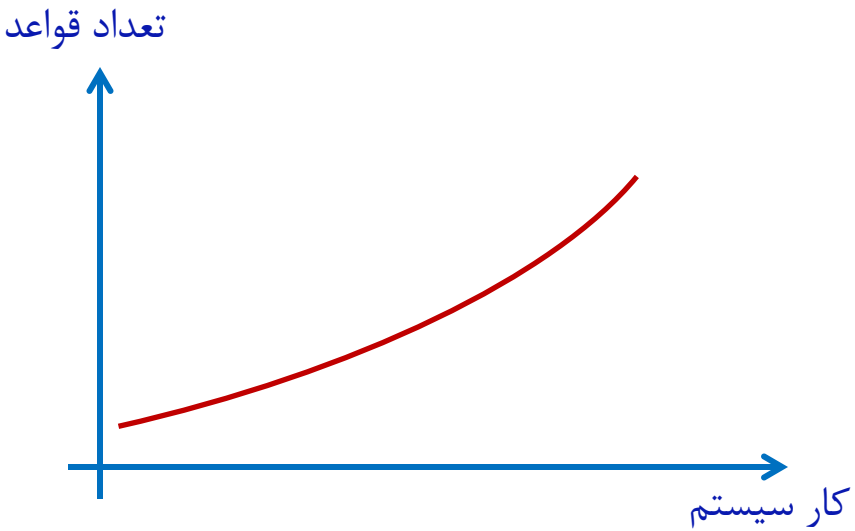
← **اجرایی:** قواعد در یک رویه به سیستم داده می‌شوند.

جامعیت در مدل رابطه‌ای (ادامه)

- هر DBMSای باید بتواند جامعیت پایگاه داده‌ها را کنترل و تضمین کند.
- **دلیل:** زیرا همیشه ممکن است عواملی سبب نقض جامعیت شوند. از جمله:
  - اشتباه در برنامه‌های کاربردی (به ویژه اشتباهات معنایی)
  - اشتباه در وارد کردن داده‌ها
  - وجود افزونگی کنترل نشده
  - اجرای همروند تراکنشها به گونه‌ای که داده نامعتبر ایجاد شود.
  - خرابی‌های سخت‌افزاری و نرم‌افزاری

جامعیت در مدل رابطه‌ای (ادامه)

- اِعمال IRها برای سیستم سربار دارد.
- **منشأ سربار (دلایل بروز سربار) در DBMS**
  - انجام نگاشت‌ها (ناشی از معماری)
  - قواعد جامعیتی
  - اِعمال ضوابط امنیت داده‌ها در سطح DBMS



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، A_i 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)

STUD (STID, …)

777
888
444

SCR (STID, COID, …)

777	CO1
…	…
444	CO4

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

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

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

- برای اِعمال قاعده C2 در مدل رابطه‌ای لازم است:

۱- معرفی FKها به سیستم

۲- دادن گراف ارجاع

۳- مشخص کردن روش اِعمال در عملیات حذف و به‌هنگام‌سازی مقدار کلید اصلی

(در درج روش خاصی لازم نیست و در صورت عدم وجود تاپل مرجع، درخواست رد می‌شود.)

```
CREATE TABLE SCR
        (STID      CHAR(6) NOT NULL
         COID      CHAR(6) NOT NULL
         TR        CHAR(1)
         YR        CHAR(5)
         GR        DEC(2, 2))
        CHECK (0 <= GR <= 20)
        PRIMARY KEY (STID, COID)
        FOREIGN KEY STID REFERENCES STUD
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY COID REFERENCES COUR
                ON DELETE CASCADE
                ON UPDATE CASCADE
```

گراف ارجاع

روش اِعمال (انتشار عمل)

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

- روش‌های اِعمال C2 در حذف (بعضاً در به‌هنگام‌سازی):

۱- روش CASCADE: انتشاری یا تسلسلی

در این روش با حذف تاپل مرجع، تمام تاپل‌های رجوع کننده به آن حذف می‌شوند.

هر چه گراف ارجاع سنگین‌تر باشد، کار سیستم در اینجا بیشتر است.

DELETE FROM STUD
WHERE STID=‘444’

DELETE FROM SCR
WHERE STID=‘444’

۲- روش RESTRICTED: روش منوط به ... (یا مشروط به ...) یا روش تعویقی

در این روش اگر بخواهیم تاپل مرجع را حذف کنیم، ابتدا باید تاپل‌های ارجاع کننده به آن حذف شوند.

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

- روش‌های اِعمال C2 در حذف (و بعضاً در به‌هنگام‌سازی):

۳- روش SET TO NULL: روش هیچ‌مقدارگذاری یا Nullifying

در این روش باحذف تاپل مرجع، FK در تاپل‌های رجوع کننده Null می‌شود به شرط آنکه FK جزء سازنده PK نباشد.

۴- روش SET TO DEFAULT: روش درج پیش‌فرض

در این روش، با حذف تاپل مرجع، FK با مقدار پیش‌فرض جاگذاری می‌شود به شرط آنکه FK جزء سازنده PK نباشد.

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

- **روش‌های اِعمال C2 در حذف (و بعضاً در به‌هنگام‌سازی):**

۵- روش NO ACTION: عدم اقدام

برای این روش دو پیشنهاد داده شده است:

۵-۱- عدم اقدام مطلق: مثلاً مجاز نبودن عمل حذف تاپل مرجع و نمایش خطا

۵-۲- انجام عمل خواسته شده و نه اقدام دیگر: تاپل مرجع حذف بشود ولی اقدام دیگری انجام نشود. در این مورد طراح-پیاده‌ساز می‌پذیرد که موقتاً محدودیت C2 نقض شود.

- در حالت وجود چرخه ارجاع کدام روش انجام شدنی است؟
- نمی‌توان روش RESTRICTED را در حالت کلی اِعمال کرد. با روش CASCADE هم ممکن است تاپل‌های ناخواسته حذف شود.
- در این مواقع NO ACTION را انتخاب می‌کنیم.

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 [ستونی]

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

محدودیت‌های ناظر به صفت:

۱- صفت y تابع صفت x است (وابستگی تابعی دارد).

۲- مقادیر صفت B زیرمجموعه‌ای از مقادیر صفت A است. وابستگی شمولی $B\{values\} \subseteq A\{values\}$

۳- صفت سن کاهش نمی‌یابد (محدودیت پردازشی).

محدودیت ۱ و ۲، **محدودیت‌های وضعیتی** هستند ولی محدودیت ۳، **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.

تعداد واحد درس‌های عملی حداکثر ۲ واحد است.

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، COUR 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 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)

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

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

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 COUR  JOIN  STCOT
                                GROUP BY  (STID, TR, YR)
                                HAVING SUM(CREDIT) > 20) )
```

امکانات بیان محدودیت‌ها – رهانا

رهانا [راه‌انداز] – TRIGGER

- امکانی است اجرایی برای اِعمال محدودیت‌های [صفتی،] رابطه‌ای و پایگاهی.

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)

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’)
        )
```

- **مطالعه یادداشت‌های تکمیلی در خصوص رهانا**

پرسش و پاسخ . . .

amini@sharif.edu

به نام آنکه جان را فکرت آموخت

بخش هفتم:
طراحی پایگاه داده رابطه‌ای

مرتضی امینی

نیمسال اول ۹۲-۹۳

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

طراحی پایگاه داده رابطه‌ای

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

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

طراحی پایگاه داده رابطه‌ای (ادامه)

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

ویژگی‌های طراحی خوب

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

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

طراحی بالا به پایین

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

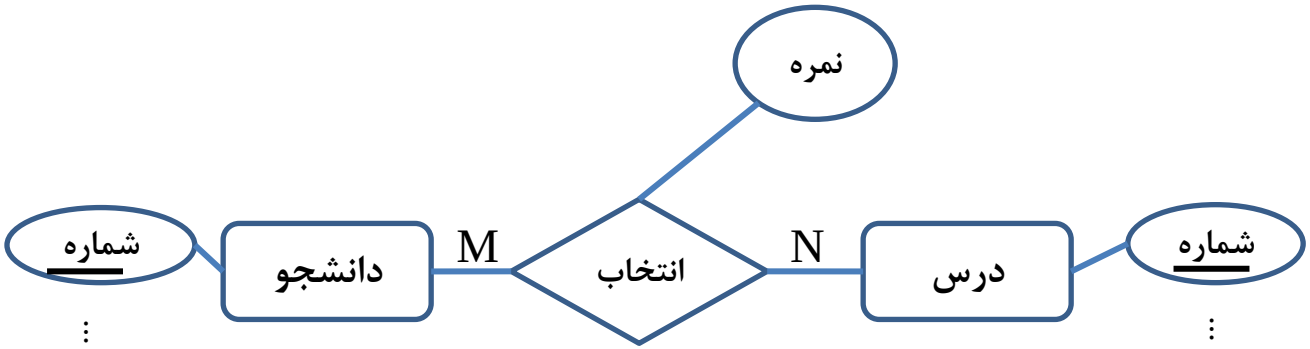
حالت ۱: طراحی ارتباط چند به چند

حالت ۱

- درجه ارتباط: n=2
- چندی ارتباط: M:N

سه رابطه لازم است.

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



STUD (STID, ….)

COR (COID, ….)

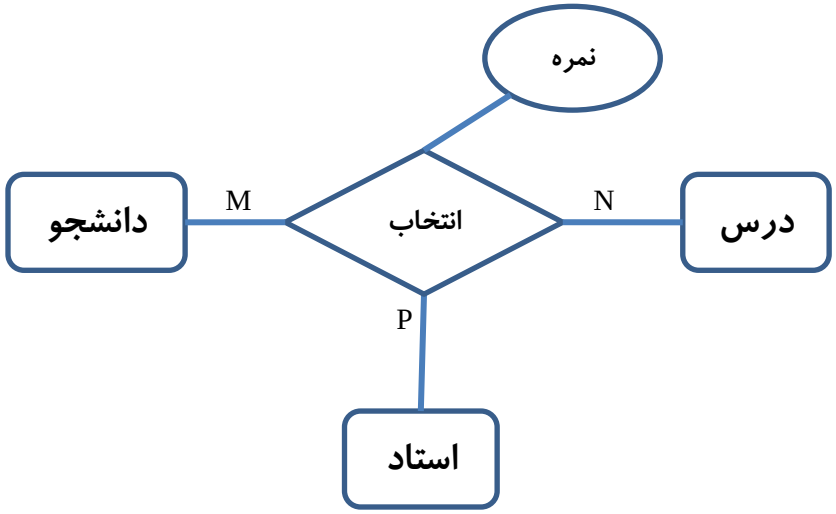
SCR (STID, COID, GR)

حالت ۱: طراحی ارتباط چند به چند (ادامه)

تعمیم حالت ۱

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

حالت ۱: طراحی ارتباط چند به چند (ادامه)



STUD (STID, ….)

COR (COID, ….)

PROF (PRID, ….)

SCP (STID, COID, PRID, GR)

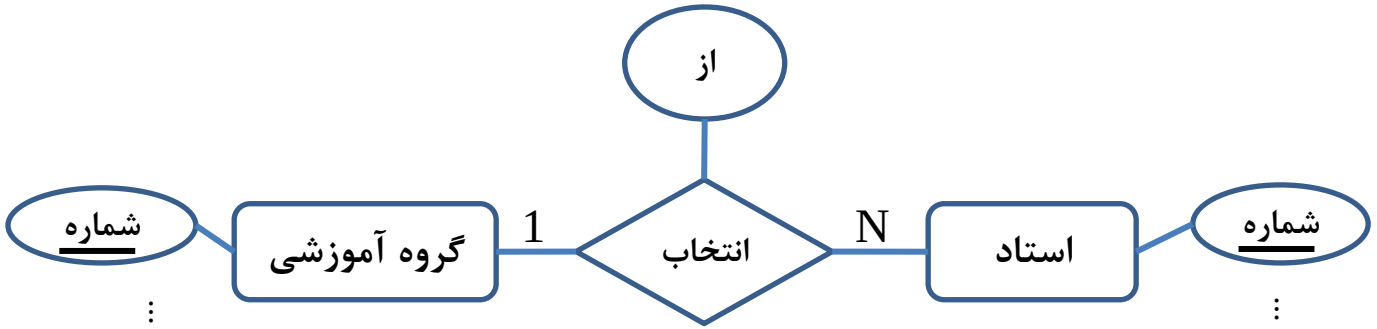
- فرض برای محدودیت: یک استاد فقط یک درس را تدریس می‌کند (البته در این مورد چندی رابطه دقیق مدل نشده که این محدودیت لحاظ نشده است).
  - در این صورت باید رابطه SCP را به دو رابطه (یا بیشتر) تجزیه عمودی کنیم.
- این محدودیت را در مرحله دوم طراحی (در مباحث آتی) دخالت می‌دهیم.

حالت ۲: طراحی ارتباط یک به چند

حالت ۲

- درجه ارتباط: n=2
- چندی ارتباط: 1:N

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



DEPT (DEID, DTID, …., DPHONE)

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

حالت ۲: طراحی ارتباط یک به چند (ادامه)

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

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

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

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

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

حالت ۳: طراحی ارتباط یک به یک

حالت ۳

- درجه ارتباط: n=2
- چندی ارتباط: 1:1

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

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

COUR (COID, ….., BKID)

BOOK (BKID, ….., BKPRICE)

حالت ۳: طراحی ارتباط یک به یک (ادامه)

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

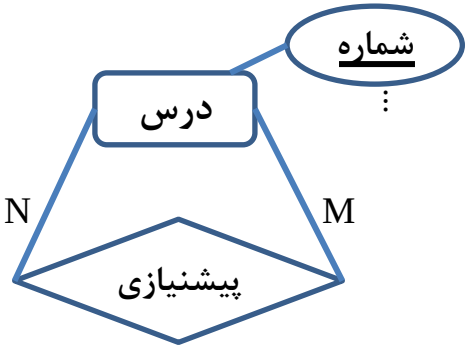
- وقتی با **یک** رابطه توجیه دارد که تعداد صفات موجودیت‌ها کم باشد، مشارکت طرفین الزامی باشد و فرکانس ارجاع به ارتباط کم باشد.

حالت ۴: طراحی ارتباط خود ارجاع چند به چند

حالت ۴

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

دو رابطه لازم است.



COUR (COID, ….)

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

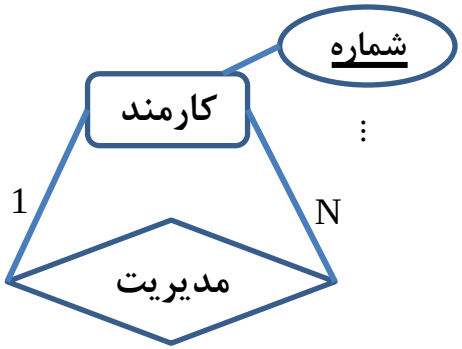
- **گراف ارجاع:** COPRECO ⟵ COUR
- **نتیجه:** صرف وجود ارتباط با خود، چرخه ارجاع ایجاد نمی‌شود. باید به چندی ارتباط توجه کنیم.

حالت ۵: طراحی ارتباط خود ارجاع یک به چند

حالت ۵

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

یک رابطه لازم است.



- در این رابطه چه نکاتی وجود دارد؟

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

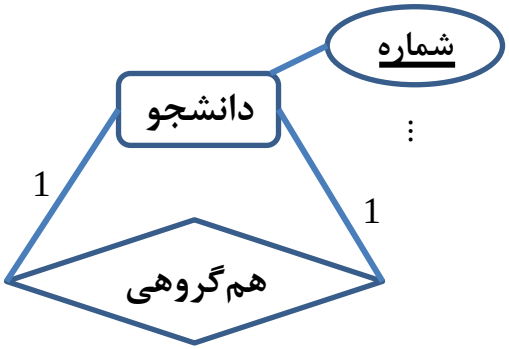
- **گراف ارجاع:**

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

حالت ۶: طراحی ارتباط خود ارجاع یک به یک

حالت ۶

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



با یک یا دو رابطه طراحی می‌کنیم.

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

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

(II) **STUD** (STID, STNAME, ….)

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

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

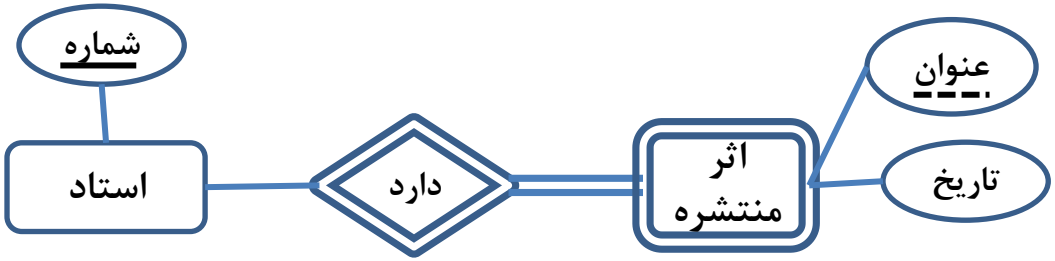
حالت ۷: طراحی موجودیت ضعیف

حالت ۷

- موجودیت ضعیف داریم.

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

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

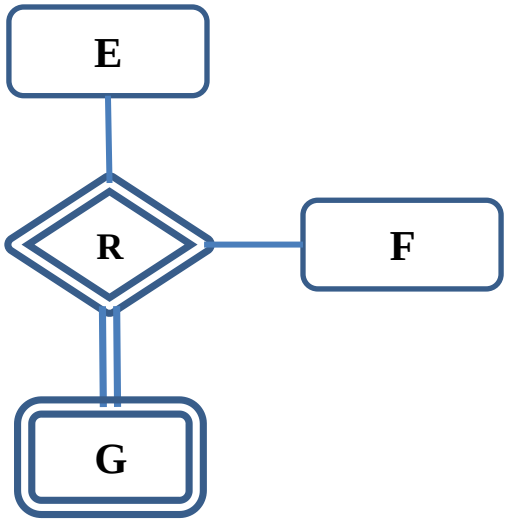


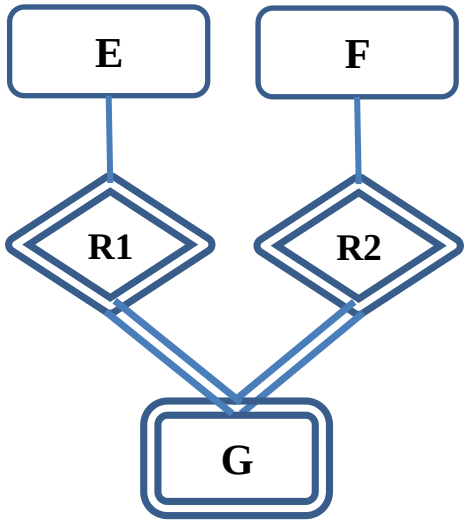
PROF (PRID, PRNAME, ….)

PRPUB (PRID, PTITLE, PTYPE, ….)

حالت ۷: طراحی موجودیت ضعیف (ادامه)

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





در این حالت ممکن است نسبت به یک موجودیت قوی، یک صفت ممیزه داشته باشد و نسبت به موجودیت قوی دیگر، صفت ممیزه دیگری داشته باشد.

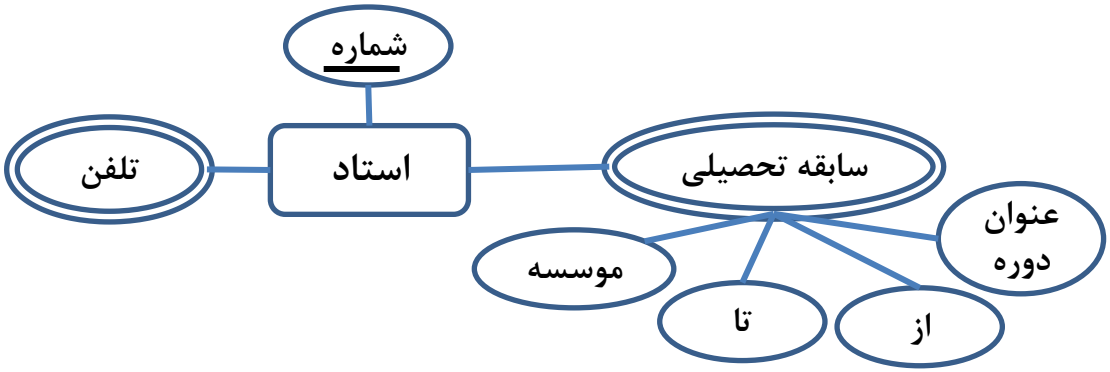
حالت ۸: طراحی صفت چندمقداری

حالت ۸

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

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

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



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

PROF (PRID, PRNAME, ….)

PRTEL (PRID, PHONE)

حالت ۸: طراحی صفت چندمقداری (ادامه)

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

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

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

حالت ۸: طراحی صفت چندمقداری (ادامه)

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

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

مثال فرض: هر استاد حداکثر سه شماره تلفن دارد.

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

- مزیت این تکنیک: JOIN لازم ندارد و لزومی ندارد که هر استاد حتما یک شماره تلفن داشته باشد.
- عیب این تکنیک: هیچمقدار (Null) در آن زیاد است، اگر تعداد کمی از استادان، سه شماره تلفن داشته باشند.

حالت ۸: طراحی صفت چندمقداری (ادامه)

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

دیگر صفات خود نوع موجودیت

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

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

حالت ۹: طراحی ارتباط IS-A

حالت ۹

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

۱- فرض: نوع موجودیت E، n زیرنوع دارد.

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

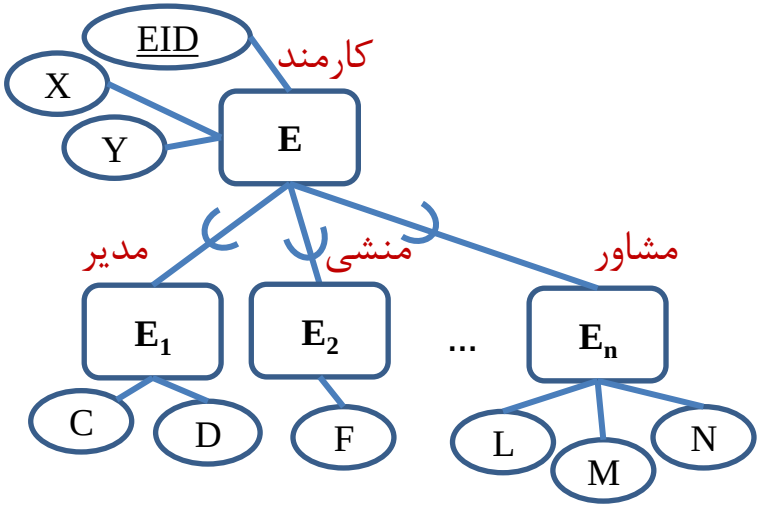
E (EID, X, Y)

E1 (EID, A, B)

E2 (EID, F)

…

En (EID, L, M, N)



حالت ۹: طراحی ارتباط IS-A (ادامه)

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

حالت ۹: طراحی ارتباط IS-A (ادامه)

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

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

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

E2 (EID, X, Y, F)

…

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

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

حالت ۹: طراحی ارتباط IS-A (ادامه)

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

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

E (EID,	X,	Y,	A,	B,	F,	L,	M,	N,	TYPE)
100	x1	y1	a1	b1	?	?	?	?	مدیر
200	x2	y2	?	?	?	l2	m2	n2	مشاور

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

حالت ۹: طراحی ارتباط IS-A (ادامه)

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

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

آرایه بیتی

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

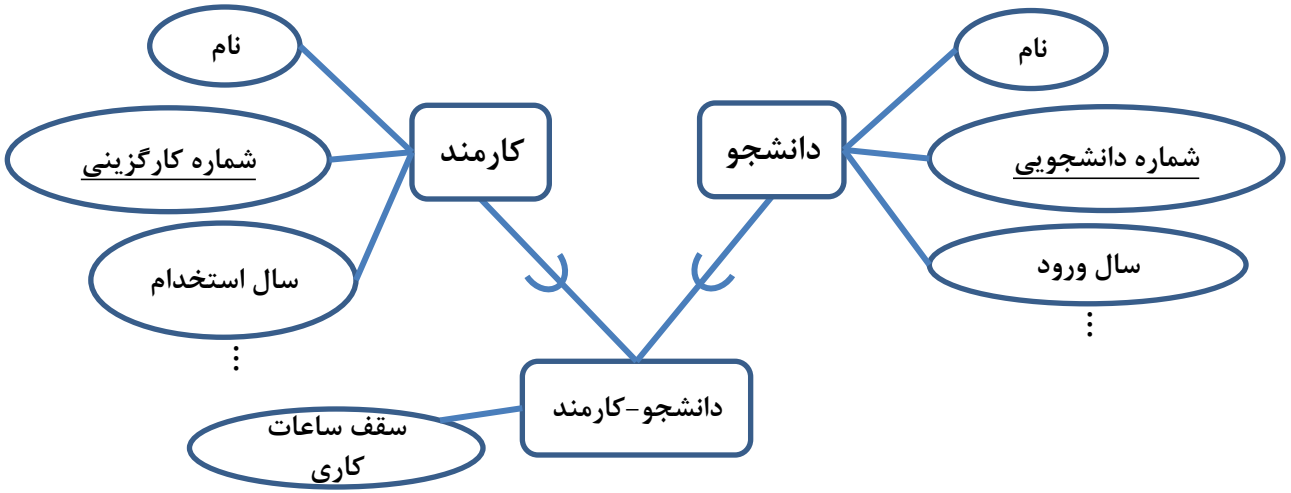
مدیر منشی مشاور

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

حالت ۱۰: طراحی ارث‌بری چندگانه

حالت ۱۰

- وجود ارث‌بری چندگانه بین یک زیرنوع با چندزبرنوع
- اگر زیرنوع، n زبرنوع داشته باشد، رابطه نمایشگر زیر حداقل n کلید کاندید دارد.



STUD (STID, STNAME, …)

EMPL (EID, ENAME, …)

STEM (STID, EID, MAXW)

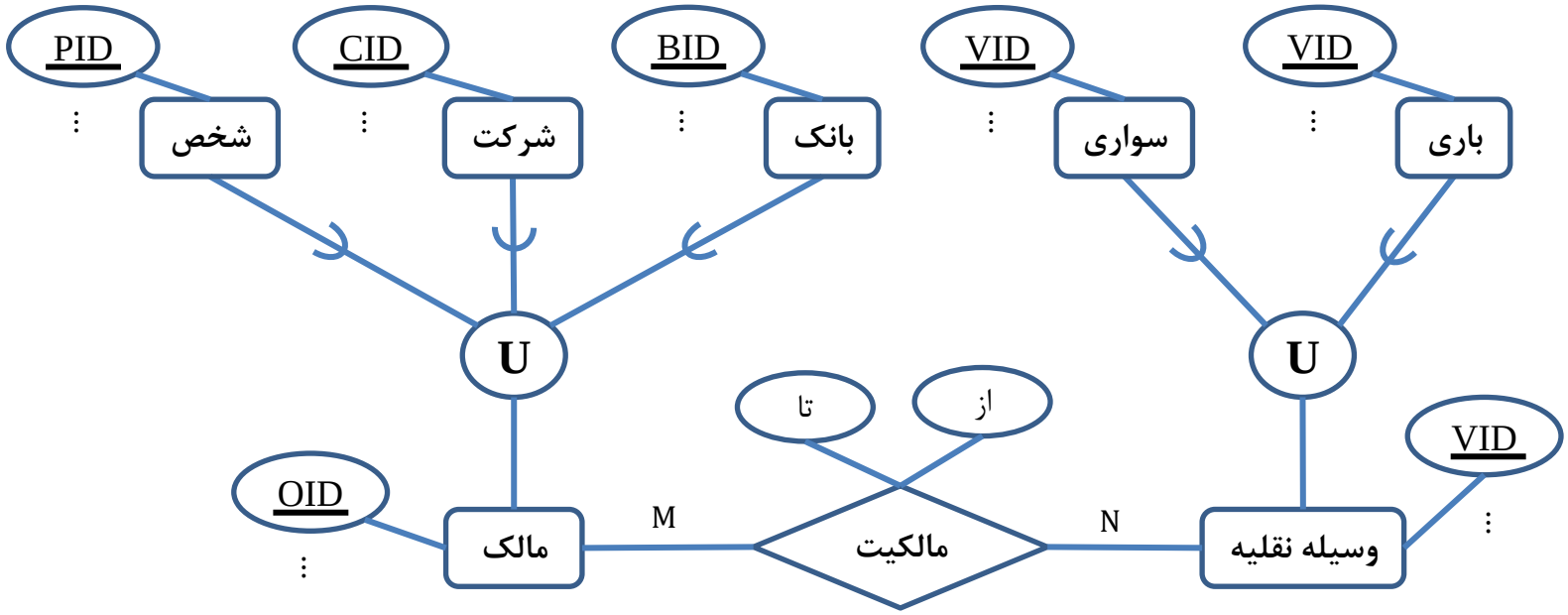
آیا ممکن است برای زیرنوع اصلاً رابطه طراحی نکنیم؟

حالت ۱۱: طراحی زیرنوع اجتماع (U-Type)

حالت ۱۱

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

n+1 رابطه طراحی می‌کنیم.



حالت ۱۱: طراحی زیرنوع اجتماع (ادامه)

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

(برای IS-A هم همین نکته مطرح است.)

PERS (PID, …., OID)

COMP (CID, …., OID)

BANK (BID, …., OID)

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

VEHIC (VID, ….)

OWNS (OID, VID, F, T, ….)

SAVARY (VID, N, ….)

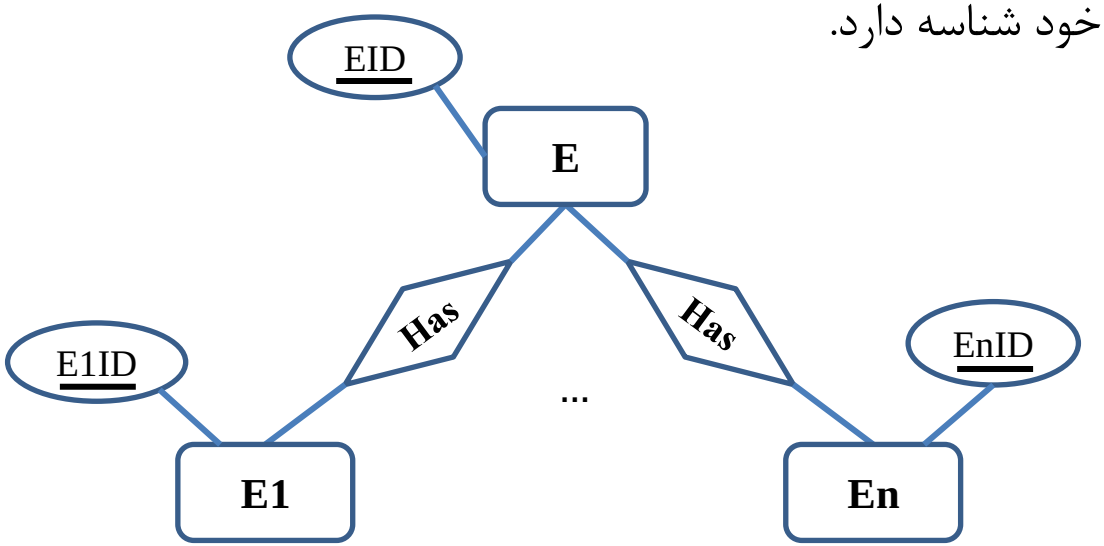
BARY (VID, T, ….)

آیا طرز طراحی دیگری وجود دارد.

حالت ۱۲: طراحی ارتباط IS-A-PART-OF

حالت ۱۲

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



E (EID, ….)

E1 (E1ID, EID, ….)

….

En (EnID, EID, ….)

آیا طرز طراحی دیگری وجود دارد؟ در چه شرایطی؟

حالت ۱۳: طراحی تکنیک Aggregation

حالت ۱۳

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

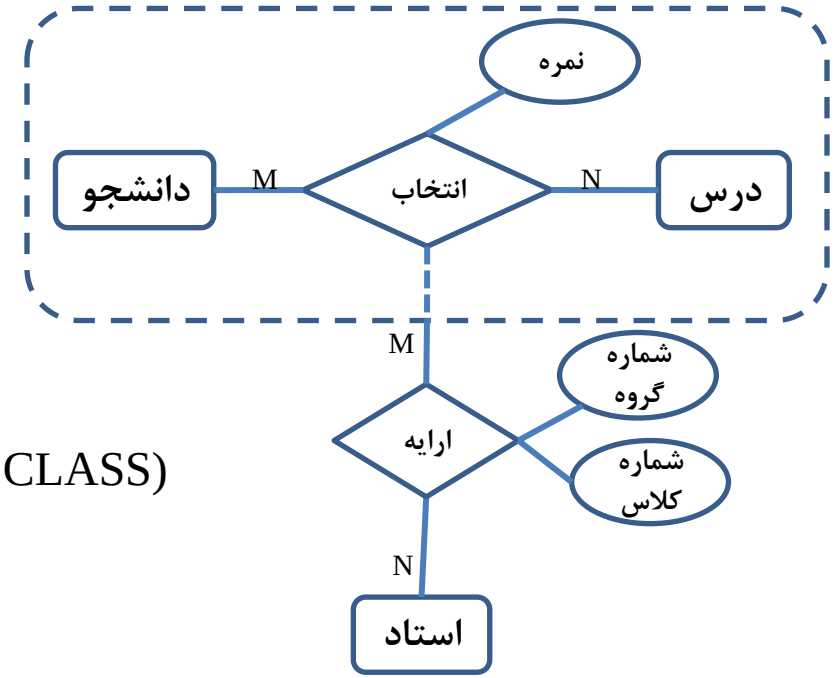
STUD (STID, ….)

COUR (COID, ….)

SCR (STID, COID, GR)

PROF (PRID, ….)

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



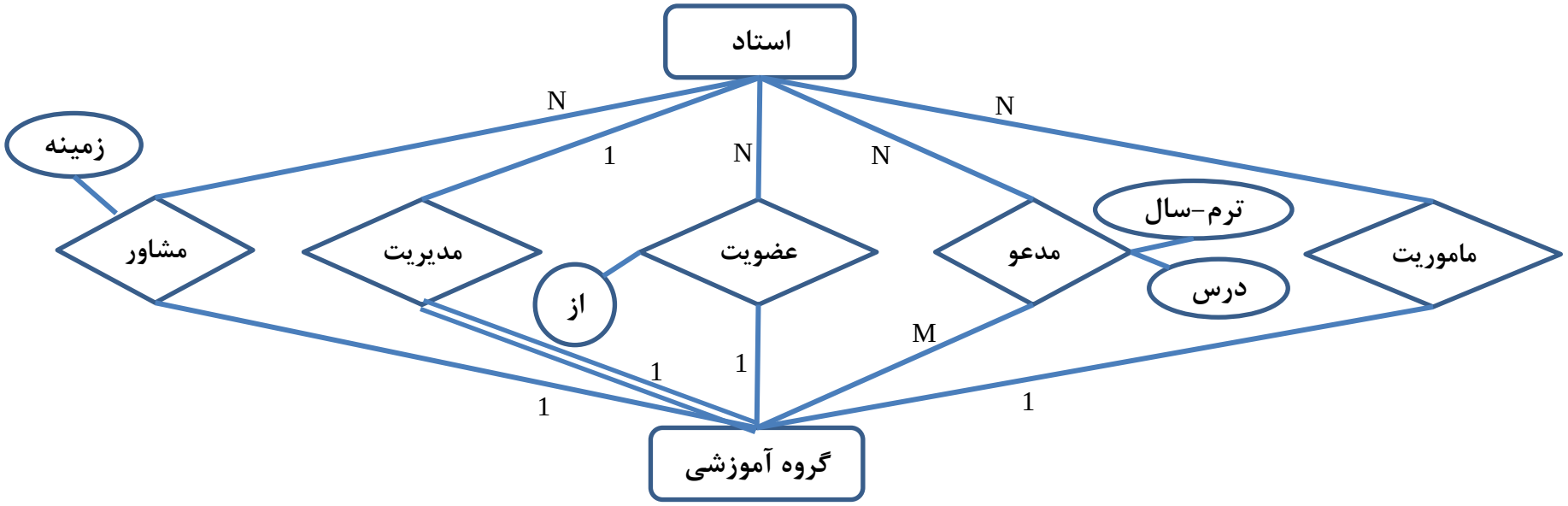
حالت ۱۳: طراحی تکنیک Aggregation (ادامه)

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

حالت ۱۴: طراحی با وجود چند ارتباط

حالت ۱۴

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



حالت ۱۴: طراحی با وجود چند ارتباط (ادامه)

DEPT (DEID, …., DPHONE, PRID)

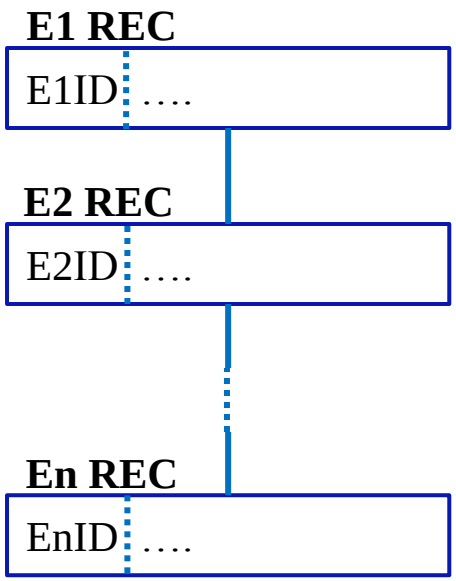
ماموریت موضوع عضویت از مشاور زمینه

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

INVITED (DEID, PRID, YR, TR)

سه کلید خارجی از یک دامنه

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



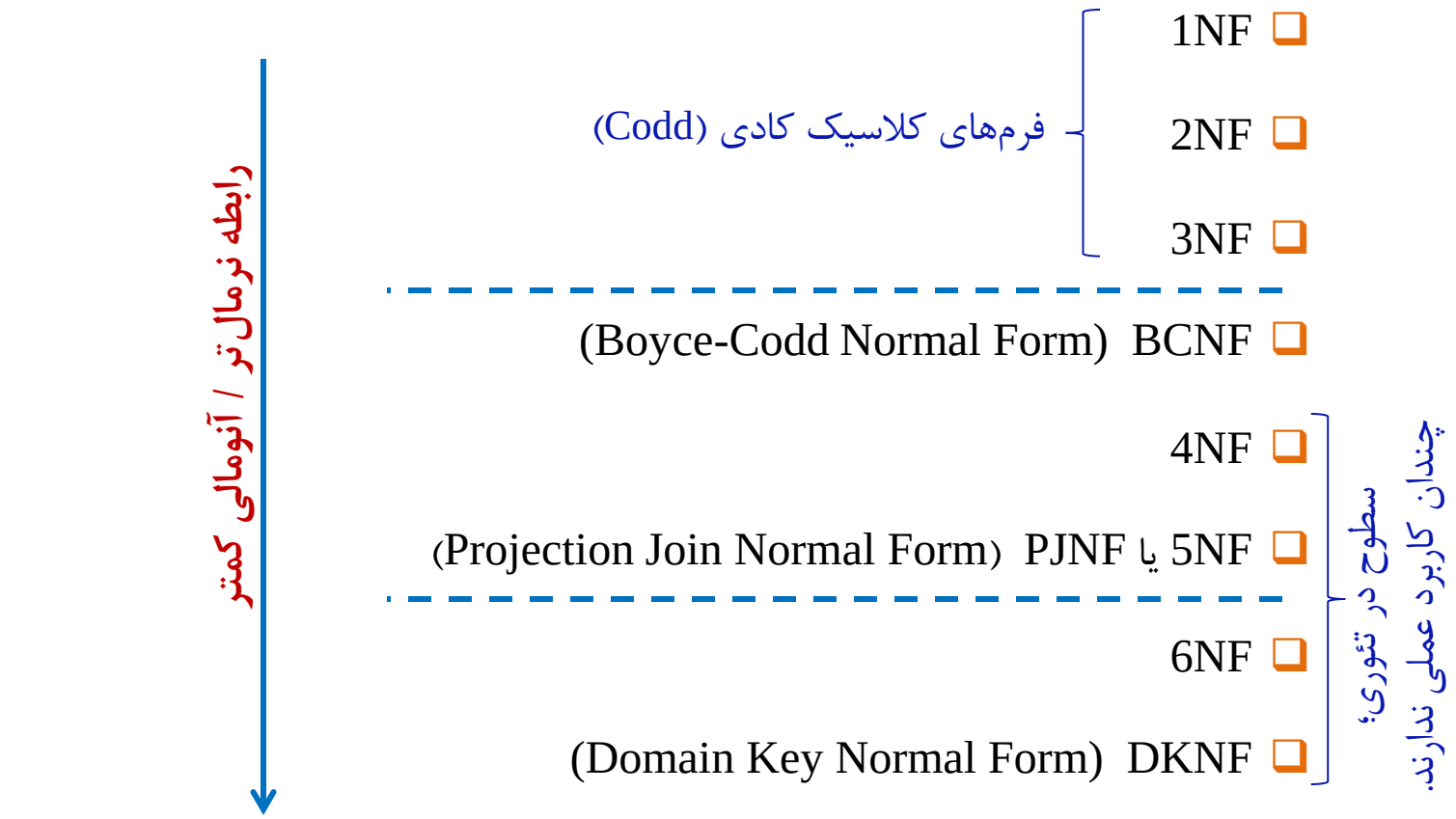
- **تمرین:** فرض می‌کنیم بین n نوع موجودیت، هر یک نمایش داده شده با یک نوع رکورد، ارتباط سلسله‌مراتبی وجود داشته باشد (بر اساس ساختار سلسله‌مراتبی HDS). مطلوب است طراحی این محیط در مدل رابطه‌ای.

طراحی RDB– روش سنتز یا نرمال‌تر سازی رابطه‌ها

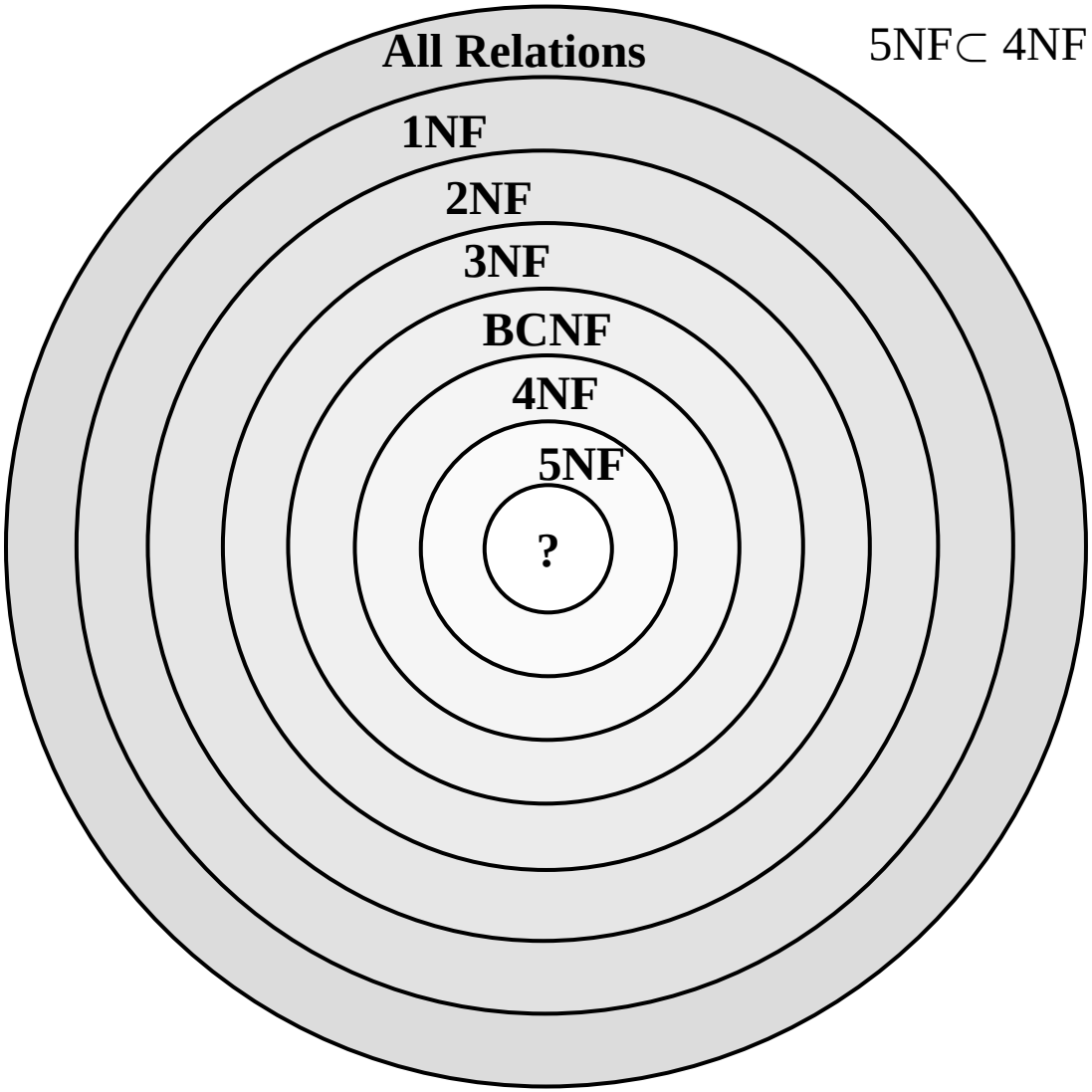
- **ایده اصلی:** یک رابطه، هر چند نرمال (با تعریفی که قبلاً دیدیم) ممکن است آنومالی (مشکل) داشته باشد در عملیات ذخیره‌سازی (در درج، حذف یا بهنگام‌سازی).
  - **آنومالی در درج:** عدم امکان درج یک فقره اطلاع که منطقاً باید قابل درج باشد.
  - **آنومالی در حذف:** حذف یک اطلاع ناخواسته در پی حذف اطلاع خواسته.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



رابطه بین فرم‌های نرمال



- 5NF⊂ 4NF ⊂ BCNF ⊂ 3NF ⊂2NF ⊂ 1NF
- یعنی رابطه‌ای که BCNF باشد،

  مثلا 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: صفت R.B با 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:

$$IF \quad t_1.A = t_2.A \quad THEN \quad t_1.B = t_2.B$$

مثال: آیا داریم:

$A \rightarrow B$

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

$a_1 \rightarrow b_1$

$a_1 \rightarrow c_1, c_2$

$A \rightarrow B$؟ بله

$A \rightarrow C$؟ خیر

$B \rightarrow A$؟ خیر

$B \rightarrow C$؟ خیر

وابستگی تابعی (ادامه)

نکات:

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

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

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

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

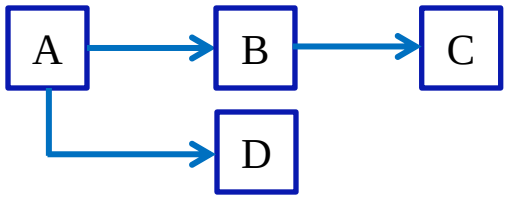
وابستگی تابعی (ادامه)

(۵) نمایش FDهای رابطه R

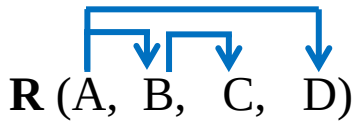
- به صورت یک مجموعه:

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

- 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)))

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

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 انعکاس)

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

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

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

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

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

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

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

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

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

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

R (A,	B,	C)
a_1	…	c_1
.	.	.
a_1	…	c_2

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

حالت اول

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

حالت دوم

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

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

کاربردهای قواعد آرمسترانگ

۱- محاسبه بستار صفت A: A^+

مجموعه تمام صفاتی که با A، وابستگی تابعی دارند.

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

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

مجموعه تمام FDهایی که از F منطقاً استنتاج می‌شوند:

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

وابستگی تابعی (ادامه)

کاربردهای مهم F^+:

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

- شرط معادل بودن: $F^+=G^+$

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

۲- تشخیص FD افزونه

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

وابستگی تابعی– قواعد آرمسترانگ (ادامه)

۳- محاسبه مجموعه کاهش‌ناپذیر FDهای یک رابطه

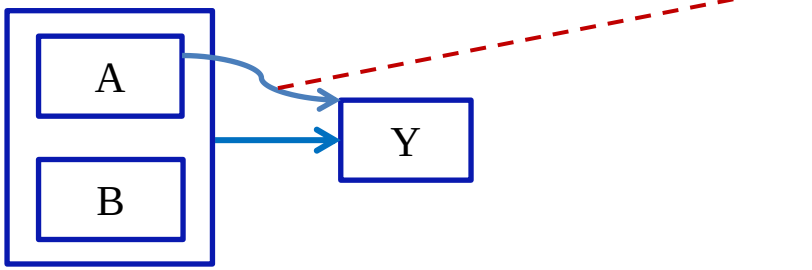
سه شرط دارد:

۱- هیچ FD در آن افزونه نباشد.

۲- سمت راست هر FD، صفت ساده باشد.

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

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



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

وابستگی تابعی (ادامه)

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

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

کجای کار؟ مجموعه کاهش‌ناپذیر چه کاربردی دارد؟

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

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

فرم‌های نرمال کلاسیک کادی

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

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 نباشد و جزء PK 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:

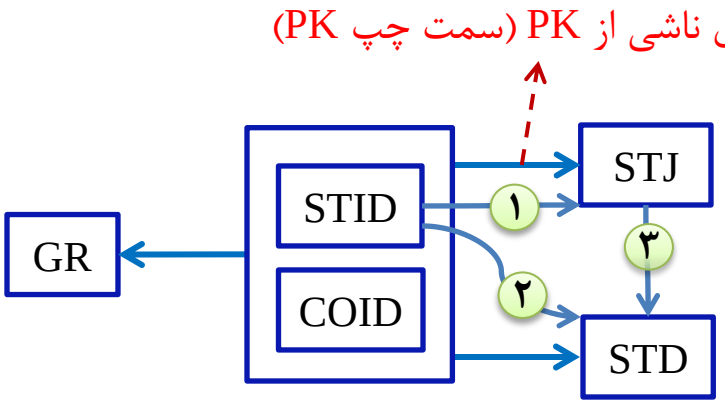
۱- یک دانشجو در یک رشته تحصیل می‌کند.

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

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

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

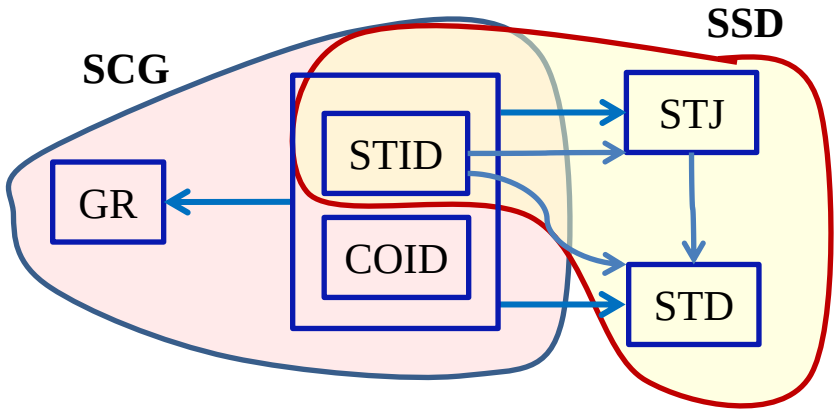
فرم‌های نرمال کلاسیک کادی (ادامه)

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

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.

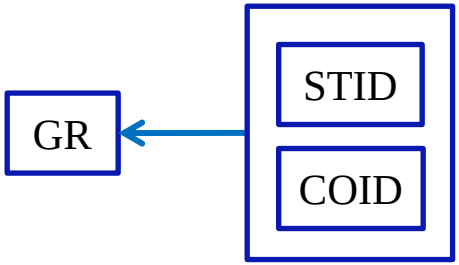
۲- حذف کن: ⟨’444’, ‘CO1’, 13⟩

بدون مشکل از SCG حذف می‌شود.

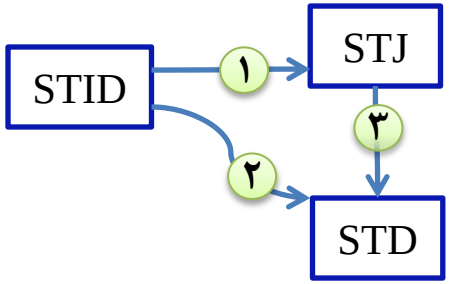
۳- بهنگام‌سازی کن: تغییر رشته دانشجوی 777 را به Chem

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. $R = SCG \bowtie SSD$

فرم‌های نرمال کلاسیک کادی (ادامه)

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

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

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

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

  - تاپل‌هایی در پیوند از دست می‌روند. به این تاپل‌ها، تاپل‌های آونگان [معلق] (Dangling) گوییم.
  - در مباحث نرمالترسازی معمولا فرض بر این است که **صفت (صفات) پیوند هیچمقدار ندارند.**

فرم‌های نرمال کلاسیک کادی (ادامه)

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

۱- در درج:

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

۲- در حذف:

حذف کن ⟨’666’, ‘Chem’⟩

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

۳- در بهنگام‌سازی:

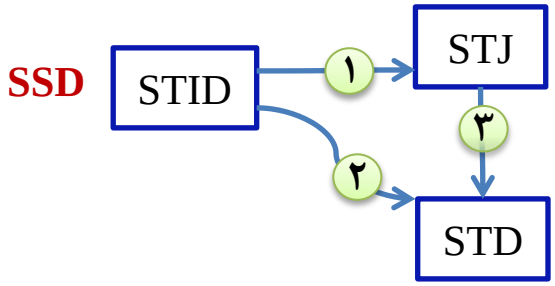
«شماره دانشکده رشته فیزیک را عوض کنید». به تعداد تمام دانشجویان این رشته باید بهنگام‌سازی شود.

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:

SJ (STID, STJ) و **SD (STJ, STD)**

STID	STJ
777	Phys
888	Math
444	Math
555	Phys

STJ	STD
Phys	D11
Math	D12

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

فرم‌های نرمال کلاسیک کادی (ادامه)

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

SJ STID → STJ

SD STJ → STD

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

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

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

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

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

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

- در بیشتر متون کلاسیک، بحث تجزیه خوب، تحت عنوان تجزیه بی‌کاست یا بی‌گمشدگی (Nonloss/Lossless Decomposition) مطرح شده است، که چندان مناسب به نظر نمی‌رسد، مگر آنکه فرض کنیم که منظور همان بی‌حشو و حافظ وابستگی‌های تابعی بودن است (و دو ویژگی دیگر تجزیه خوب را پیش‌فرض تجزیه خوب بدانیم).

- **قضیه ریسانِن (Rissanen):**
  - رابطه R به دو پرتوش (R_1 و R_2) تجزیه خوب می‌شود، اگر R_1 و R_2 از یکدیگر مستقل باشند.
  - R_1 و R_2 مستقل از یکدیگرند اگر و فقط اگر:
    - صفت مشترک، حداقل در یکی از آنها CK باشد ⇐ بی‌حشو بودن
    - تمام FDهای رابطه اصلی یا در مجموعه FDهای R_1 و R_2 وجود داشته باشند یا از آنها منطقاً استنتاج شوند ⇐ حافظ FDها

- **نکته:** بر اساس ضوابط ریسانن، اگر در رابطه R(A, B, C)، وابستگی‌های A→B، B→C و A→C برقرار باشد، در اینصورت تجزیه خوب چنین است: R_1(A, B) و R_2(B, C).
- در اینجا B در رابطه دوم کلید کاندید است، چون همه صفات به آن وابستگی تابعی دارند و کاهش‌پذیر هم نیست.

[بحث تکمیلی] تجزیه خوب (ادامه)

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

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

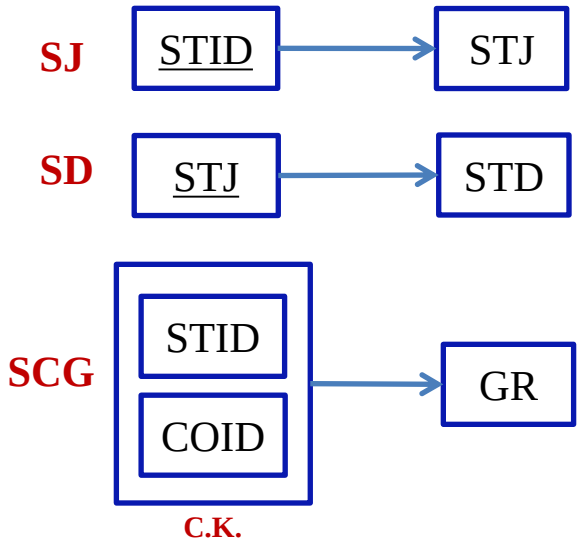
- **اصطلاح:** در وابستگی تابعی A→B (A Determines B) به A دترمینان گویند.

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)

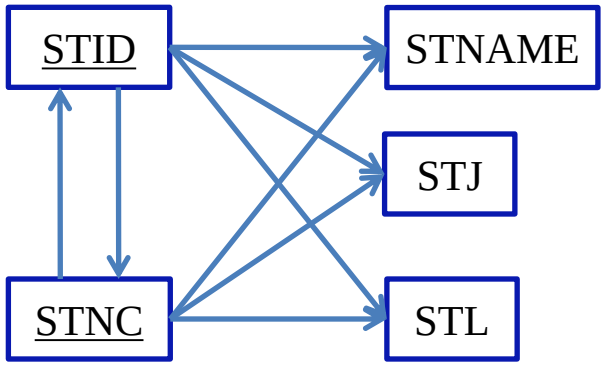
- BCNF از 3NF قوی‌تر است. ⇐ رابطه می‌تواند در 3NF باشد، اما در BCNF نباشد.
- **حالت I:** رابطه R فقط یک CK داشته باشد. ⇐ اگر R در 3NF باشد، در BCNF هم هست (مثال دیده شده).
- **حالت II:** رابطه R بیش از یک CK داشته باشد.
  - **II-1)** CKها مجزا باشند (صفت مشترک نداشته باشند). ⇐ اگر R در 3NF باشد، در BCNF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

فرم نرمال BCNF (ادامه)

مثال برای حالت II-1

ST (STID, STNAME, STNC, STJ, STL, …)
C.K. C.K.

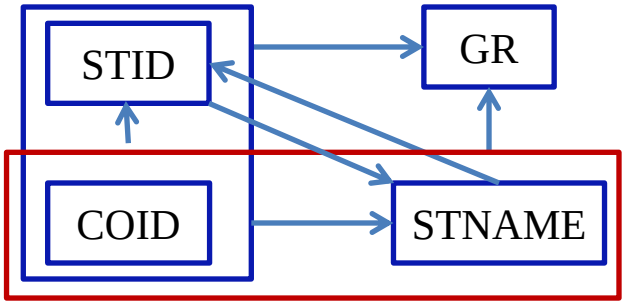
دو دترمینان، هر دو هم CK هستند.



مثال برای حالت II-2

SCNG (STID, COID, STNAME, GR)
C.K. C.K.

(فرض: هیچ دو دانشجویی نام یکسان ندارند.)



فرم نرمال BCNF (ادامه)

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

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

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

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

⇒ هر دو BCNF هستند.

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

فرم نرمال BCNF (ادامه)

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

SCNG (STID, COID, STNAME, GR)

C.K. C.K.

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

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

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

۲- یک استاد فقط یک درس تدریس می‌کند.

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

فرم نرمال BCNF (ادامه)

- فرض می‌کنیم طراح رابطه زیر را طراحی کرده است.

C.K.

SCNG (ST#, CO#, PR#)

C.K.

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

قضیه هیث (Heath): در رابطه R(A, B, C)، که در آن A، B و C سه مجموعه از صفات هستند، اگر A→B، آنگاه تجزیه R به دو پرتو R_1(A, B) و R_2(A, C)، تجزیه بی‌کاست(Nonloss) است.

فرم نرمال 4NF

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

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

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

R (A, B, C)

a_1	b_1	c_1
	b_2	
	b_3	
a_1	b_1	c_2
	b_2	
	b_3	
a_2	b_1	c_i
	b_7	

A →→B

فرم نرمال 4NF (ادامه)

نکات:

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

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

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

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

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

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

فرم نرمال 4NF (ادامه)

مثال

استاد از دانشجو گزارش آزمایشگاه می‌گیرد.

رابطه غیرنرمال با صفت چندمقداری

NNPSR (PR#, ST#, RE#)

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

NNPSR (PR#, ST#, RE#)

PR_1	777 888 444	R_1 R_2
PR_2	777 666	R_3
…	…	…

فرم نرمال 4NF (ادامه)

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

- پدیده MVD بیان فرمال صفت چندمقداری است.

مثال: فرم نرمال شده این مثال، افزونگی زیادی دارد.

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

زیرا یک دترمینان دارد که آن هم CK است.

رابطه غیرنرمال با صفت چندمقداری

NNCTX (C#, T#, B#)

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

CTX (C#, T#, B#)

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

فرم نرمال 4NF (ادامه)

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

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

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

$\langle c_1, t_2, b_8 \rangle$

$\langle c_1, t_3, b_8 \rangle$

یعنی عمل منطقاً تاپلی تبدیل شده به عمل مجموعه‌ای

- رابطه CTB باید تجزیه شود تا رابطه‌های حاصل 4NF شود.

فرم نرمال 4NF (ادامه)

- دلیل آنومالی این رابطه، وجود پدیده MVD است.

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

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

CT (C#, T#)

c_1	t_1
c_1	t_2
c_1	t_3
c_2	t_4
c_2	t_2

CB (C#, B#)

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

درج به صورت عملاً تاپلی و نه مجموعه‌ای

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

فرم نرمال 4NF (ادامه)

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

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

فرم نرمال 4NF (ادامه)

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

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

فرم نرمال 5NF

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

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

مثال

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

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

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

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

فرم نرمال 5NF (ادامه)

STUD (STID, STNAME, STJ, STL)

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

STN (STID, STNAME)

SJL (STID, STJ, STL) $\Rightarrow$ STUD= [JD]*(STN, SJL) JD به دو پرتو

STN (STID, STNAME)

SJ (STID, STJ) $\Rightarrow$ STUD= [JD]*(STN, SJ, SL) JD به سه پرتو

SL (STID, STL)

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

فرم نرمال 5NF (ادامه)

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

مثال

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

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

PCD (PR#, CO#, D#)

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

مثال

رابطه SPJ تمام کلید است. ⇐ حداقل BCNF

MVD ندارد. ⇐ 4NF است.

SPJ (S#, P#, J#)

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

فرم نرمال 5NF (ادامه)

- فرض می‌کنیم بخواهیم این رابطه را تجزیه کنیم:

SP (S#, P#)

S_1	P_1
S_1	P_2
S_2	P_1

PJ (P#, J#)

P_1	J_1
P_1	J_2
P_2	J_1

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

SJ (S#, J#)

S_1	J_2
S_1	J_1
S_2	J_1

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

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

تاپل حشو

SPJ (S#, P#, J#)

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

فرم نرمال 5NF (ادامه)

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

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

- در این مثال از سه فقره اطلاع دو موجودیتی، باید یک اطلاع سه موجودیتی را استنتاج کنیم، چرا که این یک محدودیت جامعیتی حاکم بر محیط است (وجود وابستگی پیوندی).
- توجه داشته باشید که در حالت کلی چنین استنتاجی درست نیست و پدیده دام پیوندی حلقه‌ای بروز می‌کند، ولی در اینجا به دلیل وجود وابستگی پیوندی، چنین مشکلی بروز نمی‌کند.

فرم نرمال 5NF (ادامه)

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

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

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

۱- تعداد تاپل‌ها: n+1

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

فرم نرمال 5NF (ادامه)

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

- سه دترمینان دارد که هر سه CK هستند. ⇐ BCNF است.
- MVD ندارد. ⇐ 4NF است.
- CC ندارد. ⇐ 5NF است.

R (A,	B,	C)
a_1	b_1	c_2
a_1	b_1	c_1
a_2	b_1	c_1

فرم نرمال 6NF

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

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

DKNF چیست؟

جمع‌بندی

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

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

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

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

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

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

جمع‌بندی (ادامه)

معایب تئوری نرمال‌ترسازی:

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

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

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

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

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

جمع‌بندی (ادامه)

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

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

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

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

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

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

$$\dots$$

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

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

جمع‌بندی (ادامه)

- به رابطه‌های ناشی از تجزیه افقی می‌گوییم:

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

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

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

پرسش و پاسخ . . .

amini@sharif.edu