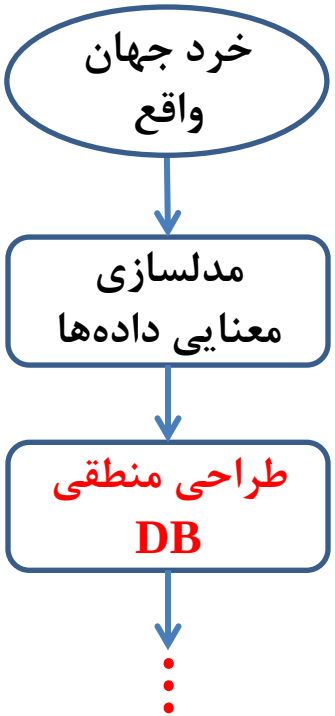
طراحی منطقی DB

- 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 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 Hierarchal 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 پکیج های 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**

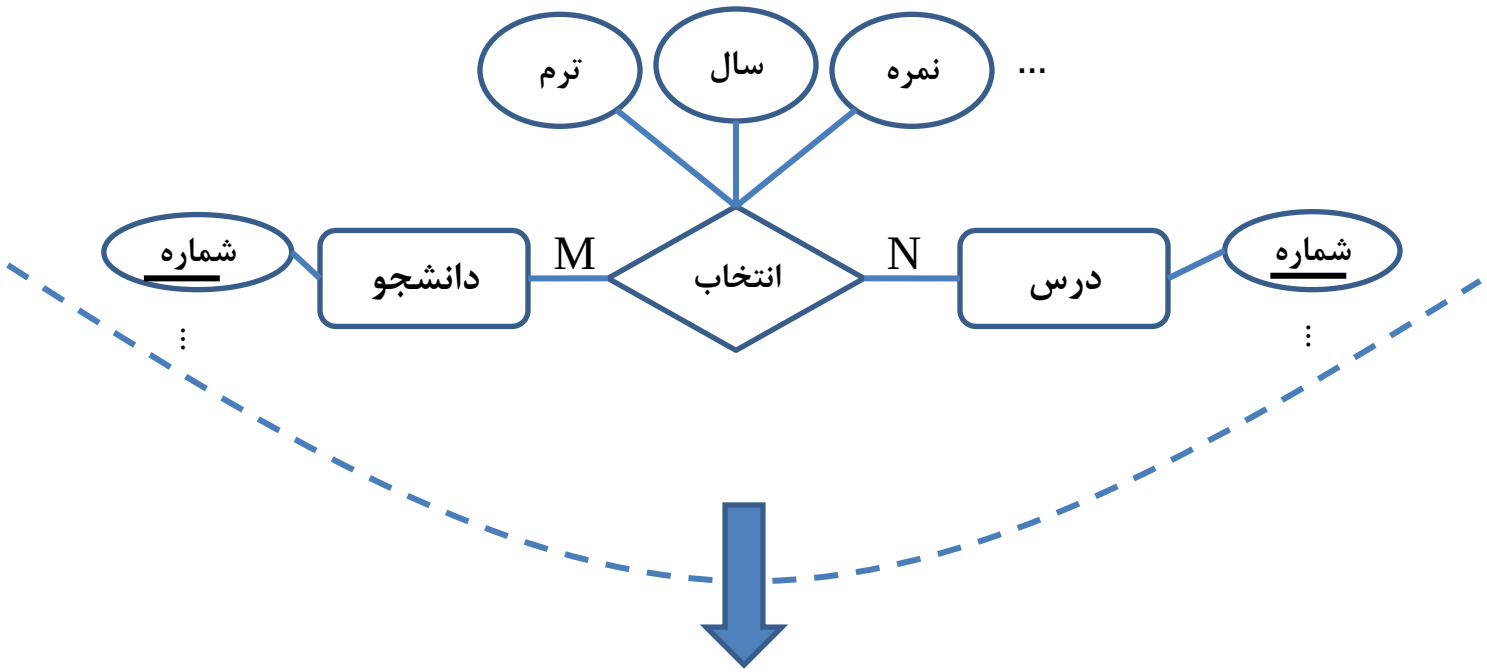
کنجکاوی؟ در HDS و NDS؟

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
⋮	⋮	⋮	⋮	⋮

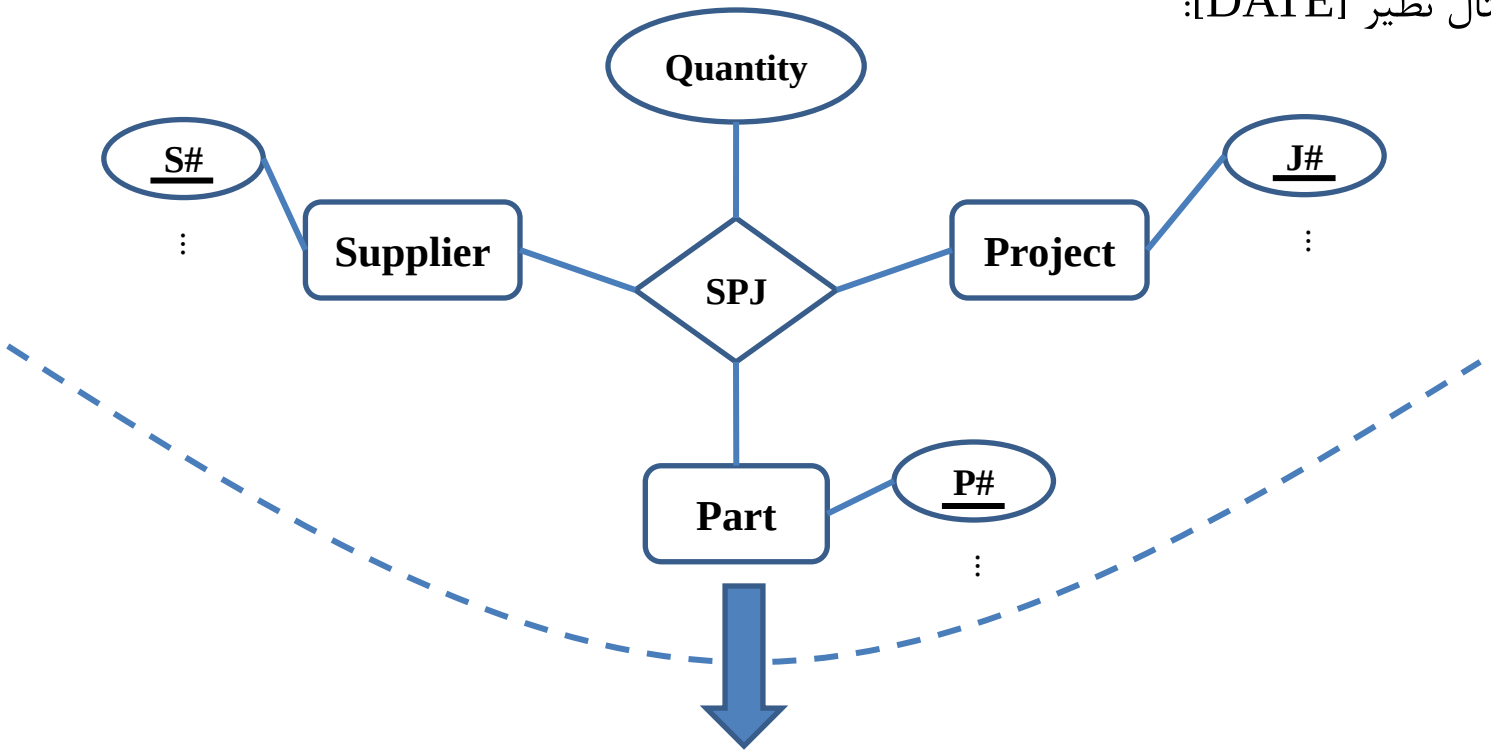
طراحی منطقی با TDS – رابطه چند به چند (ادامه)

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	…
⋮	⋮	⋮	⋮

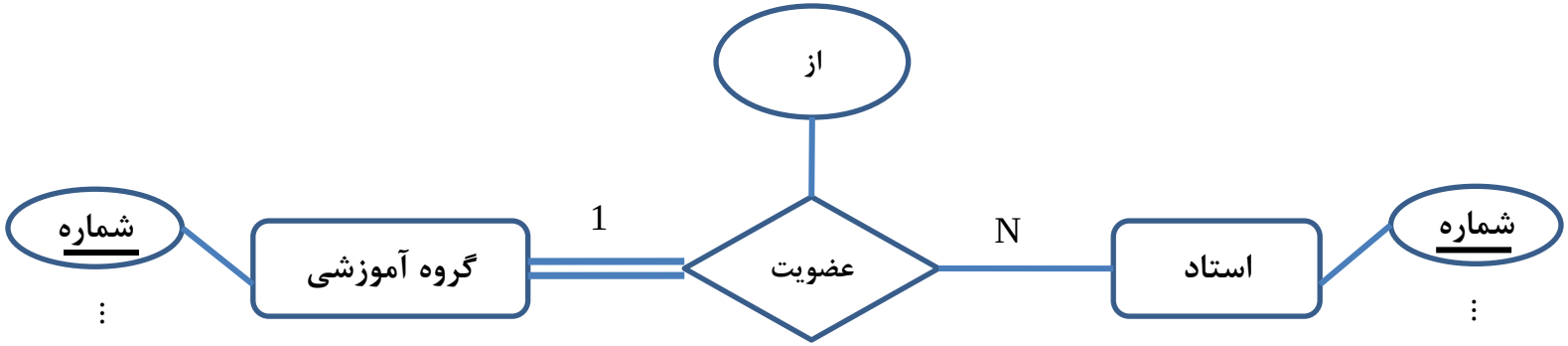
طبق قواعد معنایی محیط ممکن است تاریخ هم جزو کلید بشود.

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	…	…	…
⋮	⋮	⋮	⋮	⋮

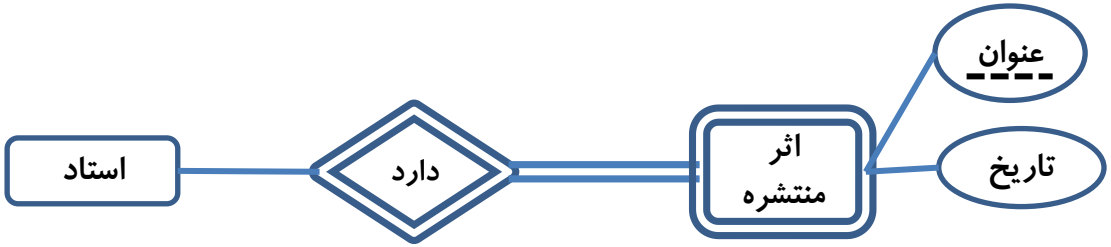
PROF

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)

PUB

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

PROF

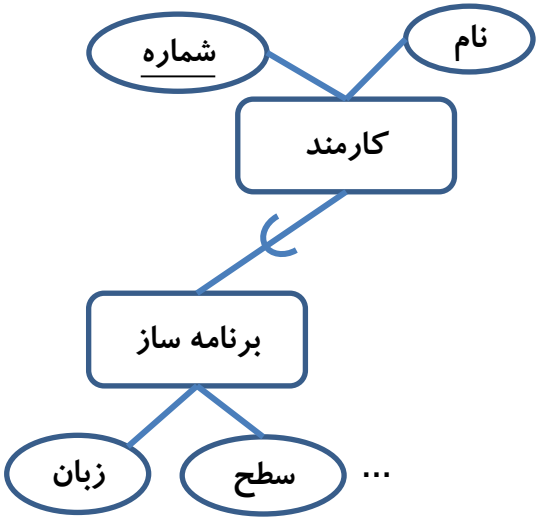
PRID	PRNAME	RANK	…	PRPHONE
Pr100	…	استاد	…	…
Pr200	…	استادیار	…	…
Pr300	…	دانشیار	…	…
⋮	⋮	⋮	⋮	⋮

- 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

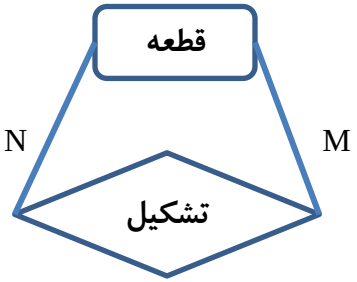
EID	LANG	…	LEVEL
E100	C++	…	…
E102	Java	…	…
⋮	⋮	⋮	⋮

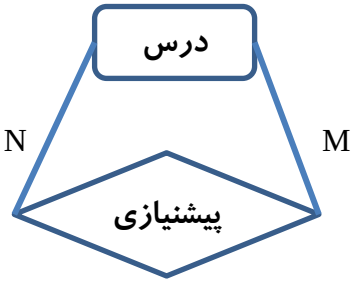
❊ EID (کلید خارجی از جدول EMP) کلید اصلی جدول PROG نیز می‌باشد.

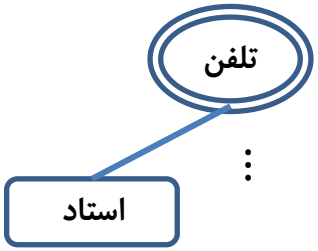
کنکاوی؟ حذف و بروزرسانی در جدول EMP چه تاثیری بر PROG باید داشته باشد (و بالعکس)؟

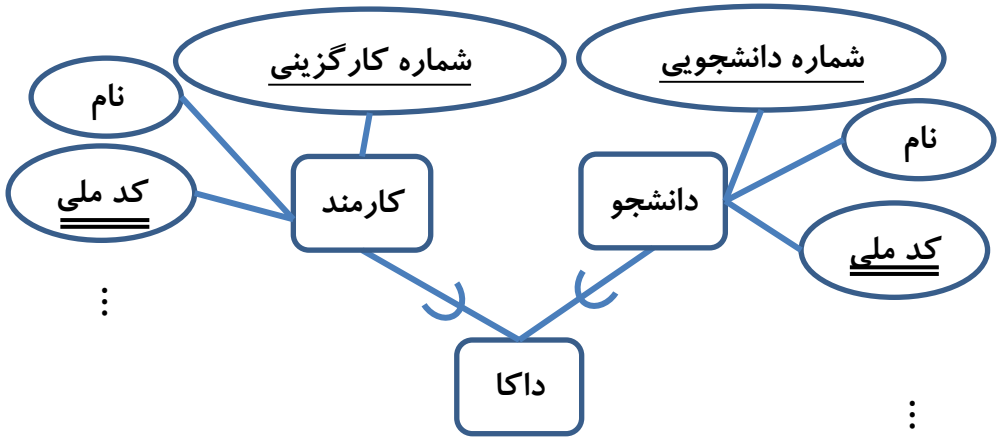
طراحی منطقی با TDS (ادامه)

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



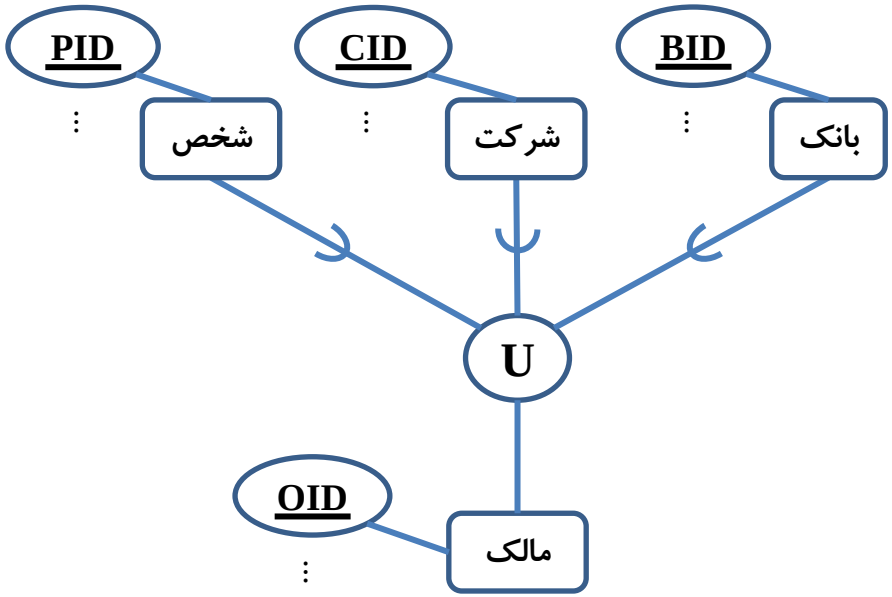


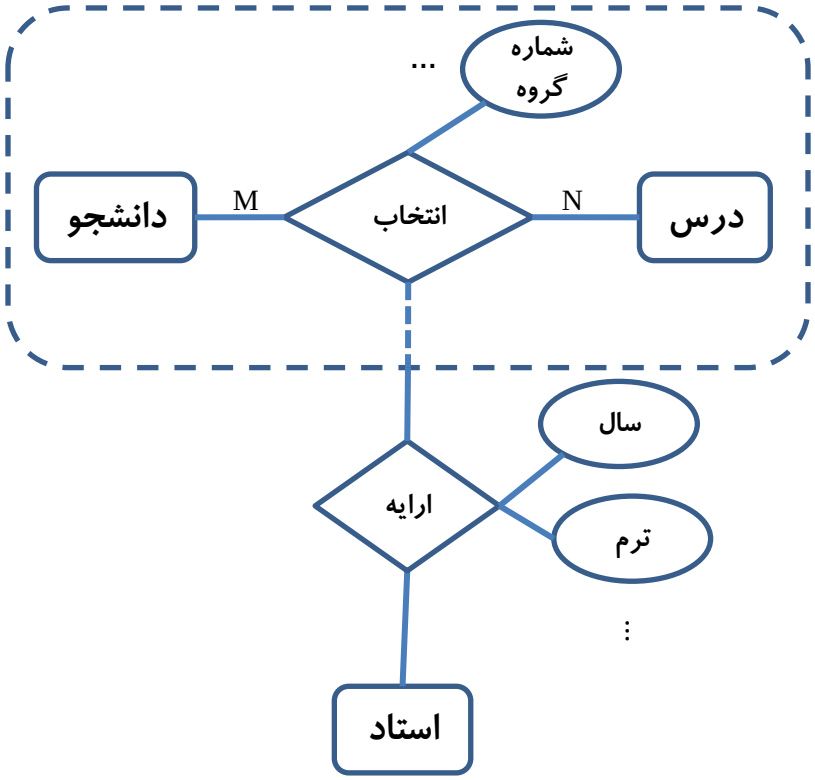




طراحی منطقی با TDS (ادامه)

- **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]**

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 داده های کنترلی در مورد داده های 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.

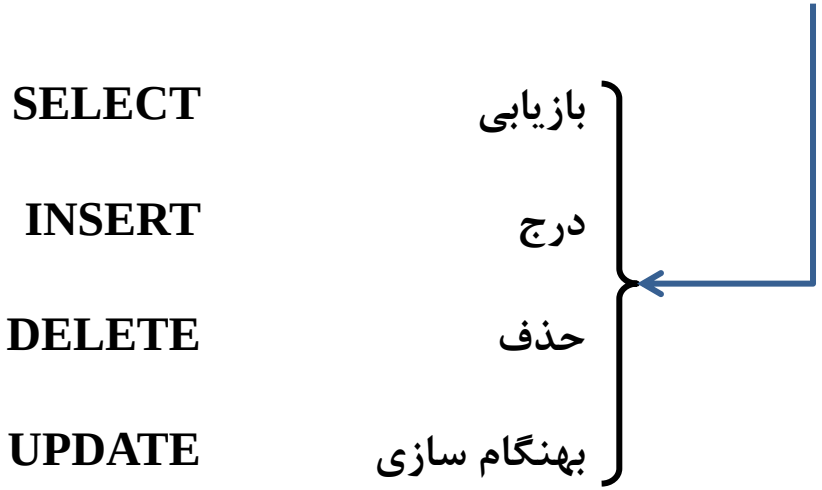
آشنایی با کاتالوگ (ادامه)

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.

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 ستون هایی که با توابع جمعی به دست آمده اند.

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’
```

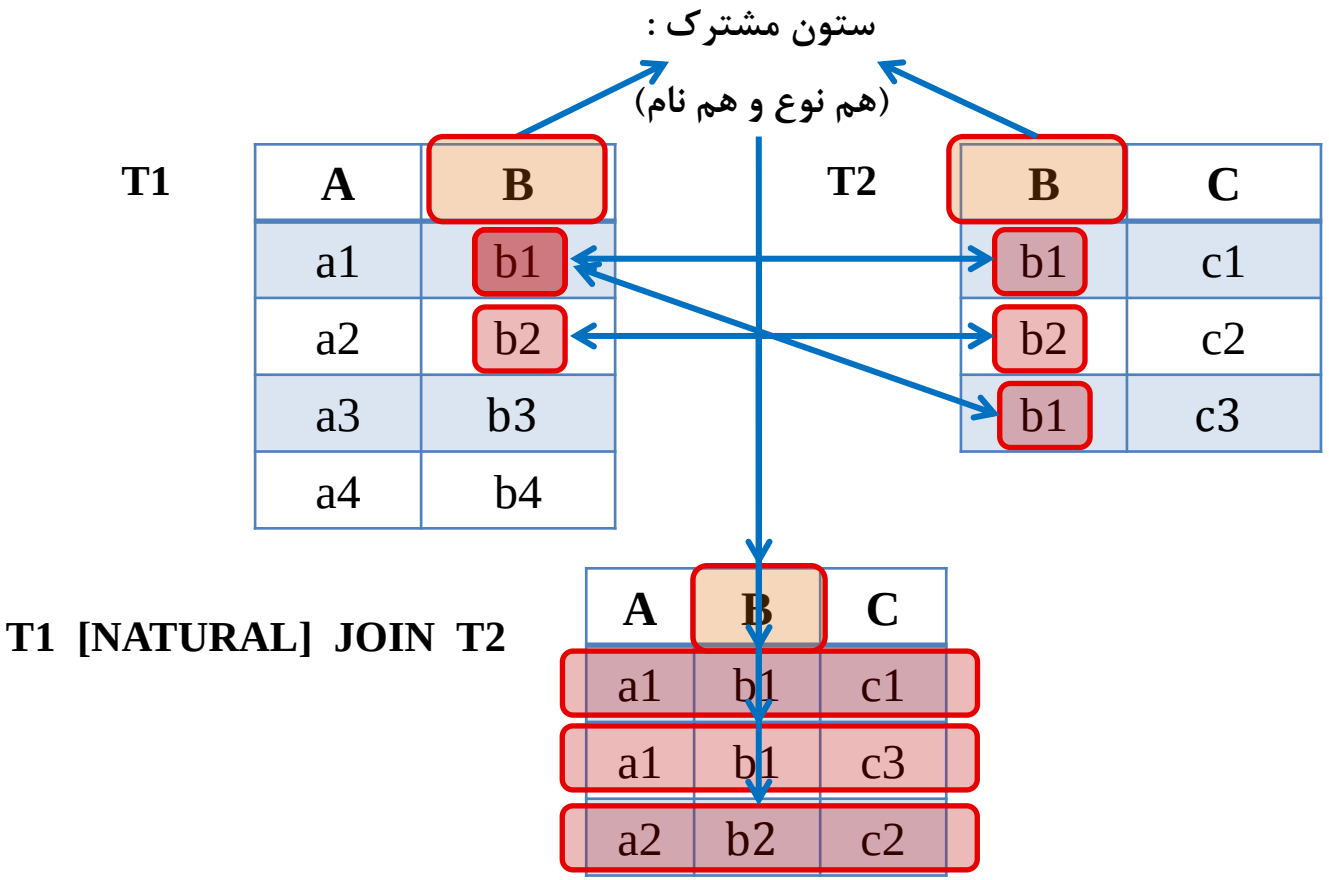
شبیه سازی عملگر پیوند

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 در 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
…	…	…		

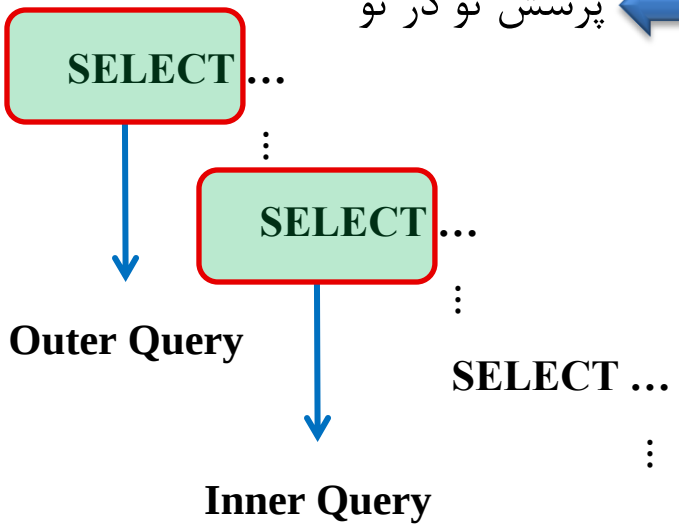
بازیابی از بیش از یک جدول – زیرپرسش

ضرب دکارتی در SQL

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

❑ زیر پرسش یا SubQuary

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



بازیابی از بیش از یک جدول – عملگر تعلق

IN و NOT IN : عملگر تعلق

مثال

روش سوم

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

روش چهارم

یا

= ANY

روش پنجم

یا

= SOME

عملگر تعلق

مکانیزم اجرا:

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

بازیابی از بیش از یک جدول – پرسش های بهم بسته

تعریف دو پرسش درونی و بیرونی (در یک پرسش تو در تو) را **بهم بسته (Correlated)** گوییم 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)

روش دهم

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

روش های 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:

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

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

بهنگام‌سازی چند سطر:

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

- تعداد واحد تمام درس های عملی گروه آموزشی D11 را برابر یک کن.

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    STDEID = 'Null'
         WHERE   STDEID = 'D333'
```