به نام آنکه جان را فکرت آموخت

بخش چهارم:
مقدمات پیاده‌سازی و SQL

دکتر عیسی زارع پور

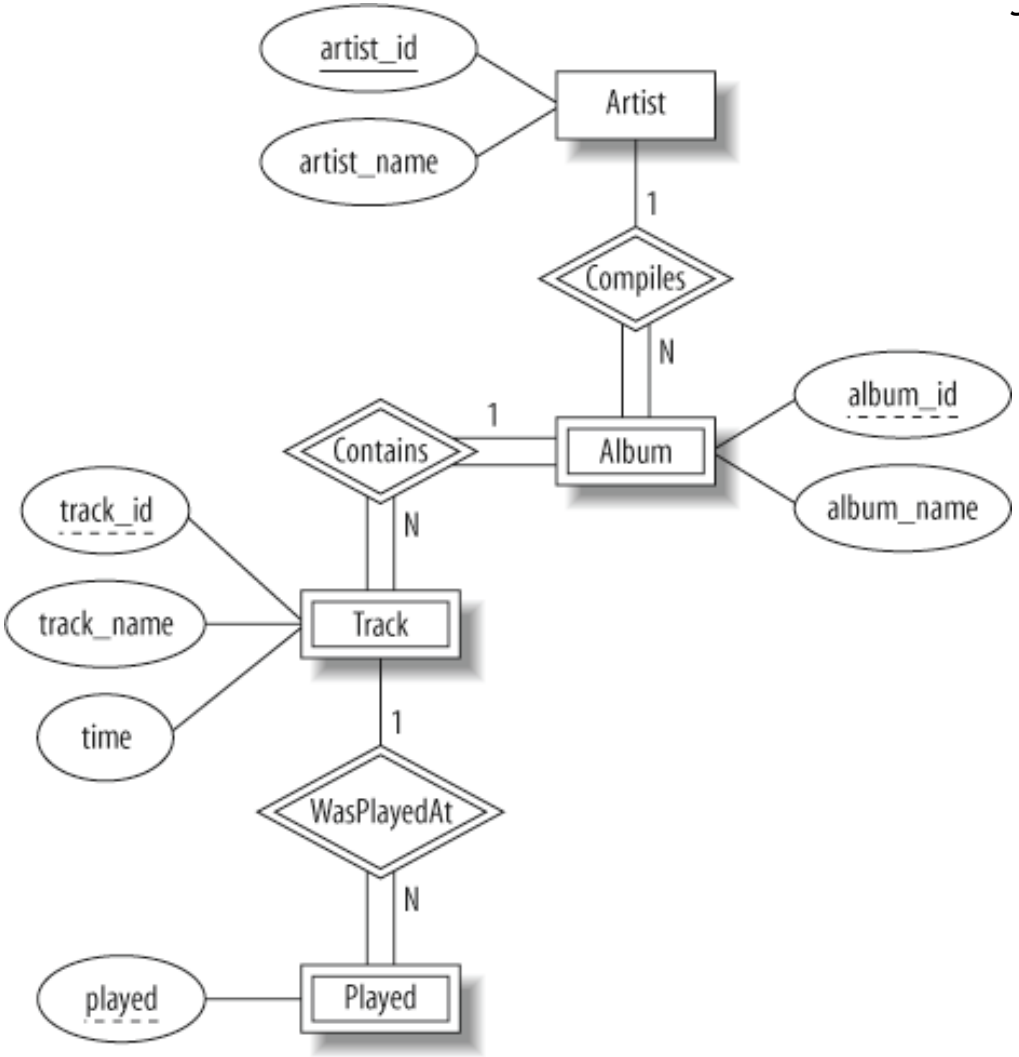
دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت

نیمسال اول ۹۹-۹۸

محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است. اسلایدها توسط آقای دکتر مرتضی امینی(دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف) تهیه شده است.

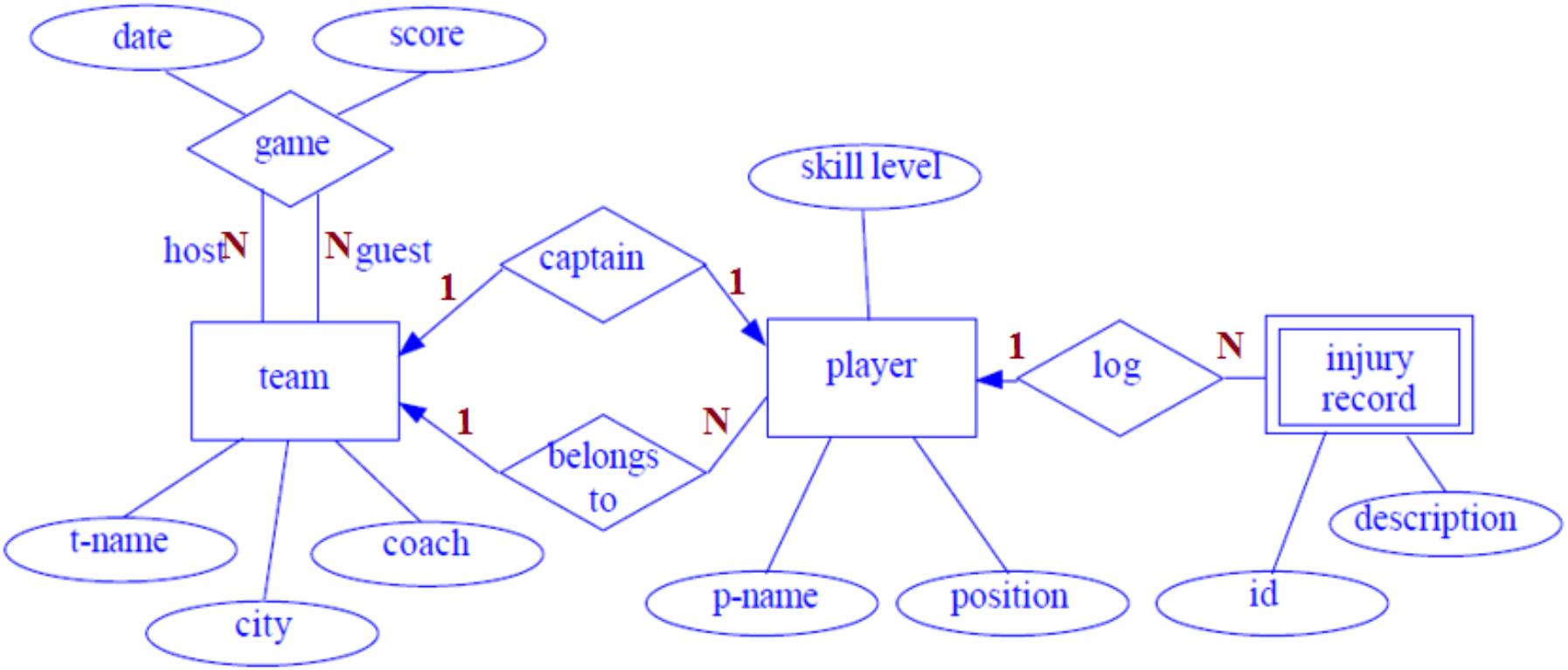
تمرین سر کلاسی

جداول لازم برای مدل ER زیر را استخراج کنید



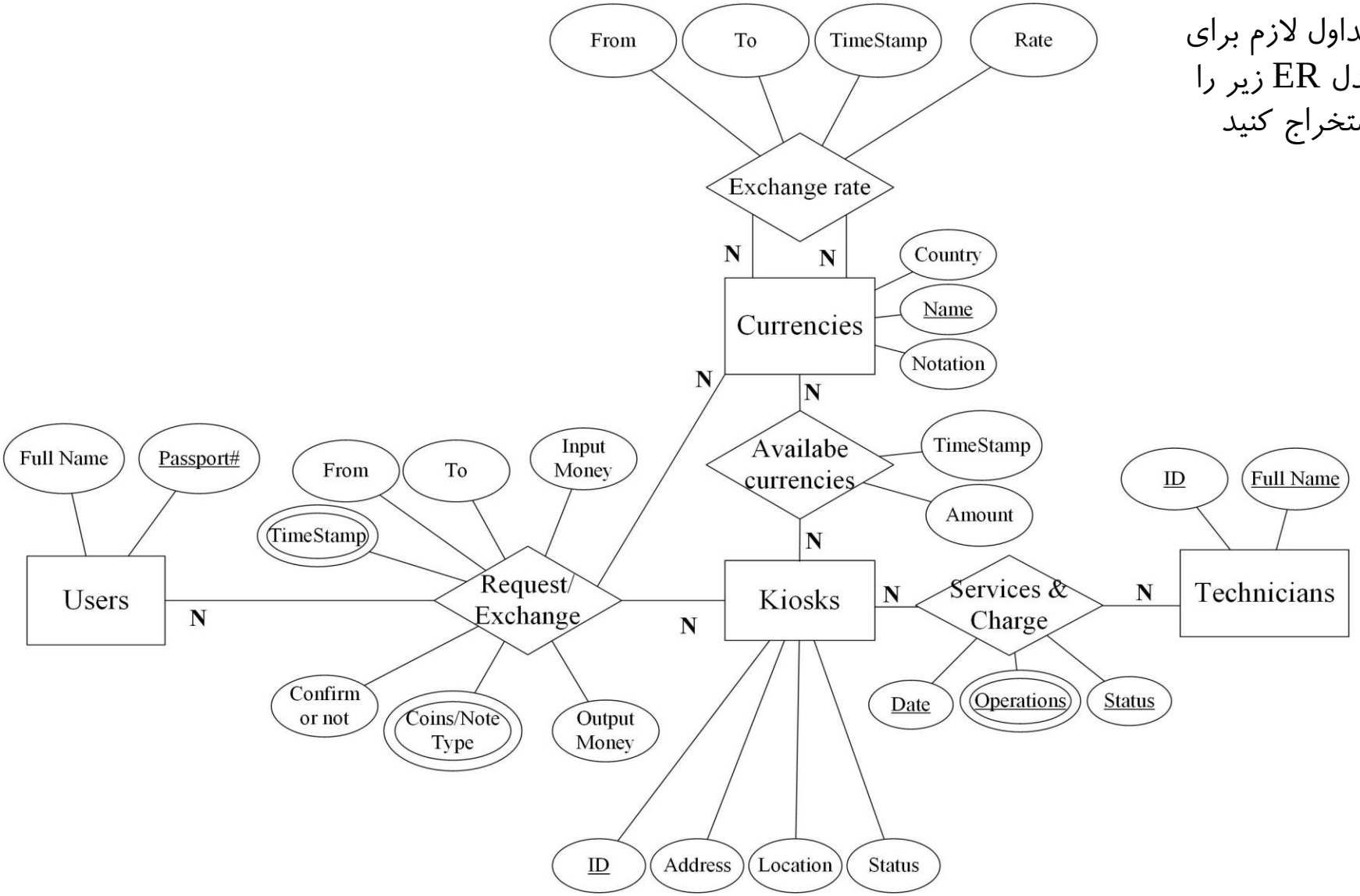
تمرین سر کلاسی

جداول لازم برای مدل ER زیر را استخراج کنید



کوییز

جداول لازم برای مدل ER زیر را استخراج کنید



مقدمات پیاده‌سازی

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

تعریف و حذف پایگاه داده و شِما

- **دستور تعریف پایگاه داده**

CREATE DATABASE *DatabaseName*

- **دستور حذف پایگاه داده**

DROP DATABASE *DatabaseName*

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

CREATE SCHEMA *SchemaName*

DROP SCHEMA *SchemaName*

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

تعریف جدول

دستور تعریف جدول CREATE TABLE

CREATE TABLE *TableName*

{ **(** *columnName* *dataType* [**NOT NULL** | **UNIQUE**]

[**DEFAUTL** *defaultOption*][***CHECK*** (*searchCondition*)] **[, …])**}

[**PRIMARY KEY** (*listOfColumns*),]

{[**UNIQUE** (*listOfColumns*),][, …]}

{[**FOREIGN KEY** (*listOfForeignKeyColumns*)

REFERENCES *ParentTableName* [(*listOfCandidateKeyColumns*)],

[**ON UPDATE** *referentialAction*]

[**ON DELETE** *referentialAction*]][, …]}

{[**CHECK** (*searchCondition*)][, …]}

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

انواع داده‌ای

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

محدودیتهای صفتی (ستونی)

- **Default**: تعیین مقدار پیش‌فرض یک ستون
- **Not Null**: ستون ناهیچ‌مقدار
- **Unique**: یکتایی مقادیر ستون(ها)
- **Primary Key**: کلید اصلی (می‌توان تعدادی از ستونها را با یکدیگر به عنوان کلید اصلی تعریف کرد)
- **Foreign Key .... References ....**: کلید خارجی (می‌توان تعدادی از ستونها را با یکدیگر به عنوان کلید خارجی تعریف کرد)
- در برخی از DBMS ها نیازی به کلمه کلیدی foreign key نیست و تنها لازم است که جدول والد را با کلمه کلیدی references مشخص کرد.
- **Check**: تعیین محدودیت مقداری برای مقادیر ستون

شیوه های مختلف ایجاد و مدیریت یک محدودیت روی صفت

- بعد از تعریف هر ستون

```
CREATE TABLE Persons
(P_Id  int  NOT NULL  PRIMARY KEY,
FirstName varchar(255),
LastName varchar(255) NOT NULL,
Address varchar(255),
City varchar(255) )
```

- تعریف در انتهای تعریف همه ستونها

```
CREATE TABLE Persons
(P_Id int NOT NULL,
FirstName varchar(255),
LastName varchar(255) NOT NULL,
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id))
```

- تعریف بعد از همه ستونها با نام

```
CREATE TABLE Persons
(P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName))
```

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
```

مثالی از تعریف جدول

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  (STLEV in ( 'bs' , 'ms' , 'doc' ))
)
```

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

محدودیت صفتی (ستونی) [کلاز کنترلی]

مثالی از تعریف جدول (ادامه)

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

محدودیت صفتی (ستونی) [کلاز کنترلی]

حذف جدول

دستور حذف جدول DROP TABLE

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

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

DROP TABLE SCT

تغییر جدول

دستور تغییر جدول ALTER TABLE

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

ALTER TABLE ***tableName***

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

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

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

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

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

[ALTER [COLUMN] SET DEFAULT ***defaultOption*****]**

[ALTER [COLUMN] DROP DEFAULT]

. . .

اضافه کردن ستون «وضعیت» به جدول اطلاعات دانشجو

ALTER TABLE STT

ADD COLUMN STATE CHAR(10)

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

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.

مثال) بخشی از مدل جدولی برای یک سیستم مدیریت کتابخانه

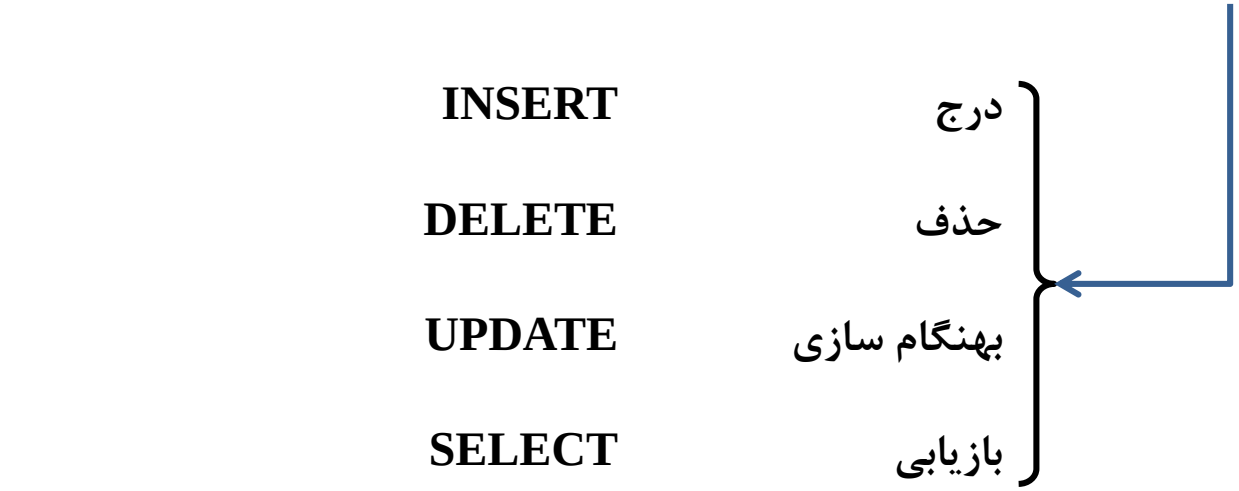
- **Libraries** (LibID, LibName, LibAddress, LibPhone)
- **Books** (BookID, BookTitle , Authors, ISBN)
- **Members** (MemberID, NatioanlID, FullName , Address, Age, JoinDate, ExpireDate)
- **Publishers** (PUBID, PUBName, Address, Phone)
- **Subjects** (SID, STitle)
- **Borrowing** (BookID, MemberID, LibID , ReturnDate date)
- **PublishedBooks** (BookID, PUBID,Location, Date)
- **PublisherSubjects**(PUBID, SID) موضوع صفت چند مقداری است
- **Book_Subject** (BookID, SubjectID)

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؟

- عملیات در TDB : **دستور های DML**



عملیات ذخیره‌سازی

دستورهای INSERT, UPDATE, DELETE

درج INSERT:

INSERT INTO *table-name* ***[****(col1,col2, ...)****]***

VALUES (***one row*) |** *subquery*

بهنگام‌سازی UPDATE:

UPDATE *table-name*

SET *col = value / experession [, col = value / experission] ...*

⋮

WHERE *condition(s) / subquery*

حذف DELETE:

DELETE FROM *table-name*

WHERE *condition(s) / subquery*

درج

درج سطری (سطر کامل – سطر ناقص):

- **INSERT INTO** STT **VALUES** **(** ‘333’ , ‘st2’ , ‘IT’ , ‘bs’ , ‘D17’ **)**
- **INSERT INTO** STT (STID, STNAME) **VALUES** **(** ‘333’ , ‘st2’ , Null , ‘bs’ , Null **)**
- **INSERT INTO** STT (STID, STNAME) **VALUES** **(** ‘333’ , ‘st2’**)**

درج گروهی:

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

```
CREATE TEMPORARY 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؟

بهنگام‌سازی (ادامه)

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'
```

دستور بازیابی SELCT

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.

بازیابی داده‌ها (ادامه)


```
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)

نمایش تعداد مشخصی از سطرهای حاصل از پرس و جو در خروجی

دانشجویان را بر اساس نام به صورت نزولی مرتب کن و پنج نفر اول را نمایش بده

```
SELECT  *  FROM  STT
           ORDER BY STNAME DESC
           LIMIT  5
```

We can also use the standard (SQL:2008) fetch first

```
SELECT  *   FROM          STT
          ORDER BY STNAME DESC
          fetch first 10 rows only
```

In Microsoft SQL Server

```
SELECT  Top 10 *   FROM      STT
          ORDER BY STNAME DESC
```

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: (Asc):
- نزولی (Desc): باید قید شود.

مثال

قابلیت‌های پیشرفته (Advanced features):

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

- LIKE / NOT LIKE ‘%N’ → با N تمام شود
- LIKE / NOT LIKE ‘M%’ → با M شروع شود
- LIKE / NOT LIKE ‘_ _ A _ _’ → دقیقا ۵ کاراکتر، کاراکتر سوم A

بازیابی داده‌ها (ادامه)

BETWEEN


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

- شماره و نام 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

<*SELECT-statement1*> *op* <*SELECT-statement2*> [CORRESPONDING [BY {*column*, [, *column* …]}]]

op ∈ { **UNION [ALL]** | **INTERSECT [ALL]** | **EXCEPT [ALL]** }

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

عملگرهای جبر مجموعه‌ها (ادامه)

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

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

مدل دیگر

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

عملگرهای جبر مجموعه‌ها (ادامه)

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.

ایجاد یک جدول با استفاده از جداول دیگر

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

- A copy of an existing table can also be created using CREATE TABLE.
- The new table gets the same column definitions. All columns or specific columns can be selected.
- If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

```
CREATE TABLE NewPersons as
select PersonID , city from  Persons;
```

توابع جمعی (گروهی)

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

مثال: بیشینه وضعیت تهیه کنندگان در شهرهای c1 یا c2

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

توابع جمعی (ادامه)

تعداد انواع قطعات تولیدی توسط تولیدکنندگان

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

تعداد انواع قطعات قابل تولید

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

تعداد کل قطعات تولیدی توسط s2

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

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

GROUP BY

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

تعداد کل قطعات تولیدی توسط هر تولیدکننده

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

گروه‌بندی SP 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– مثال

```
Table: Subject_Selection

Subject    Semester    Attendee
-------------------------------
ITB001     1           John
ITB001     1           Bob
ITB001     1           Mickey
ITB001     2           Jenny
ITB001     2           James
MKB114     1           John
MKB114     1           Erica
```

```
select Subject,
Count(*) from
Subject_Selection
group by Subject
```

```
Subject     Count
-----------------
ITB001      5
MKB114      2
```

```
select Subject,
Semester, Count(*) from
Subject_Selection group
by Subject, Semester
```

```
Subject     Semester    Count
-----------------------------
ITB001      1           3
ITB001      2           2
MKB114      1           2
```

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

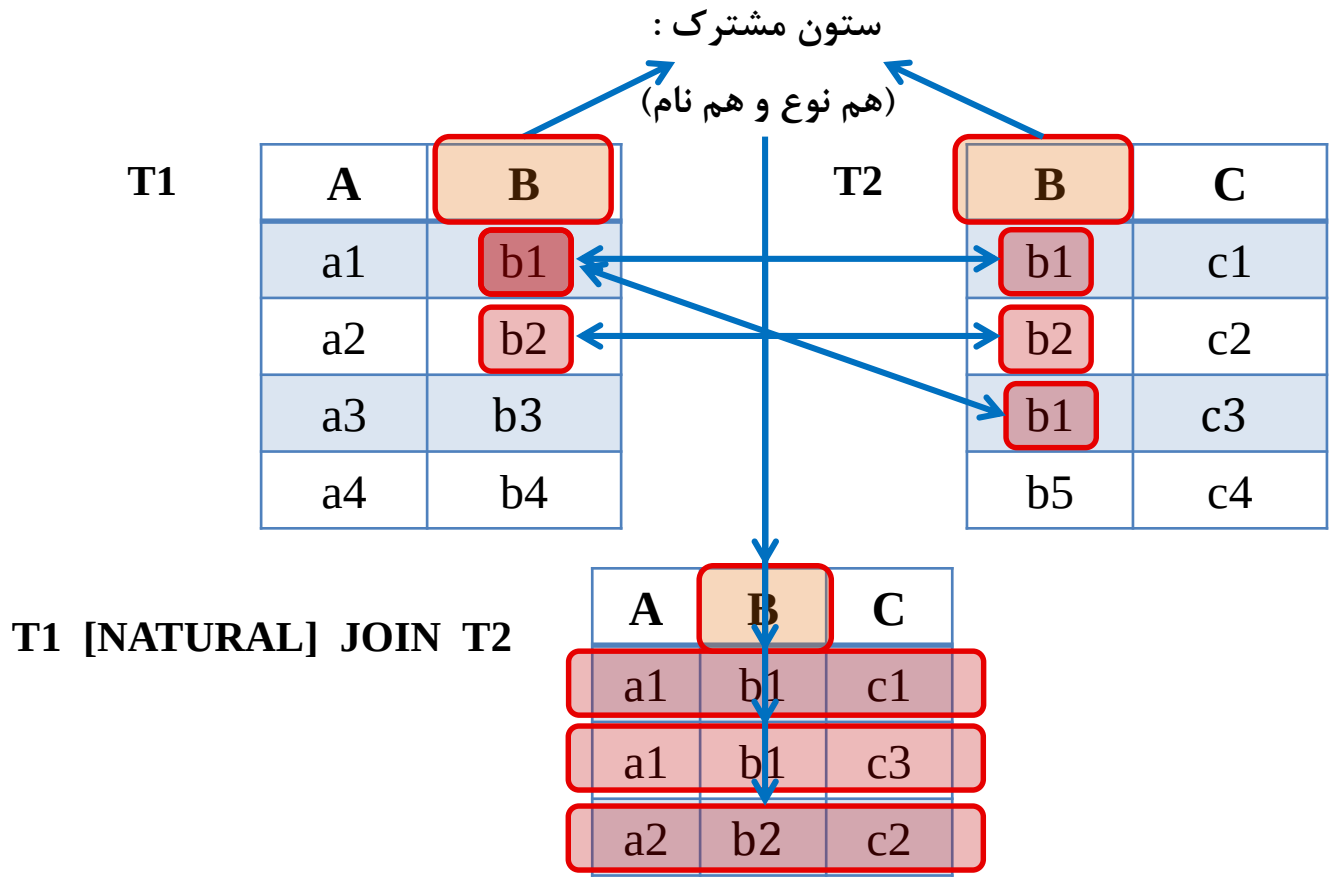
```
SELECT  T1.* , T2.*
        FROM    T1, T2
```

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 – عملگر پیوند یا JOIN (ادامه)

توضیحات تکمیلی

- می توان ستونی را که میخواهیم روی آن پیوند بزنیم به صورت صریح مشخص کنیم:

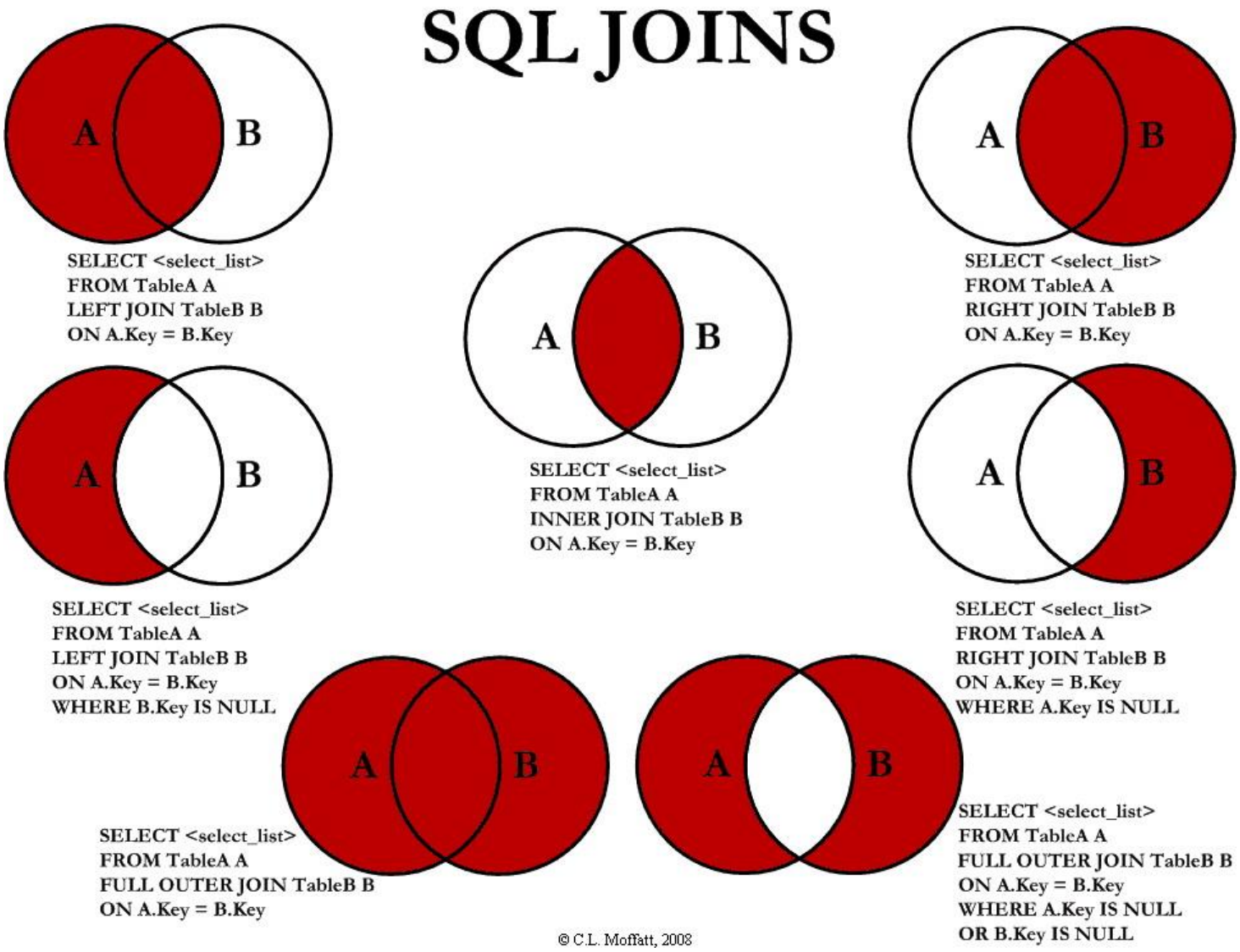
Standard SQL:

```
SELECT  SNAME
        FROM    S  JOIN  SP  ON  S.S# = SP.S#
        WHERE   P# = ‘p2’
```

PostgreSQL:

```
SELECT  SNAME
        FROM    S  JOIN  SP  Using S#
        WHERE   P# = ‘p2’
```

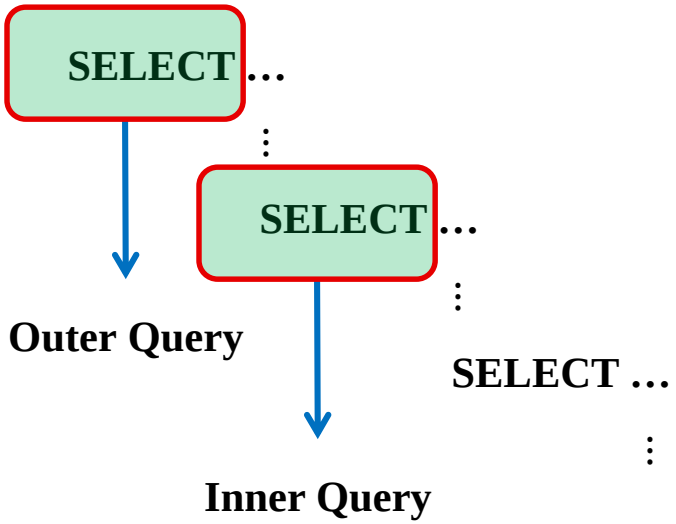
بازیابی از بیش از یک جدول – انواع عملگر پیوند یا JOIN (بعدا مفصل بحث می شود)



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' )
```

عملگر تعلق

روش چهارم

یا

= ANY

روش پنجم

یا

= SOME

مکانیزم اجرا:

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

بازیابی از بیش از یک جدول – پرسش های بهم بسته

تعریف

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

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

مثال

روش ششم

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

روش هفتم: یا = ANY

روش هشتم: یا = SOME

زیرپرسش بهم‌بسته یا CORRELATED

بازیابی از بیش از یک جدول – پرسش های بهم بسته– سوال سرکلاسی

فرض کنید جدولی به شکل زیر داریم که اطلاعات کارمندان را نگه داری می کند:

EMPLOYEE(EID, ENAME, DEPT, SALARY)

نام و مشخصات افرادی که حقوقشان از میانگین حقوق دپارتمانشان بیشتر است

```
SELECT  POSSIBLE.ENAME, POSSIBLE.DEPT, POSSIBLE.SALARY
        FROM   EMPLOYEE as  POSSIBLE
        WHERE      SALARY >
                (SELECT   AVG (SALARY)
                    FROM     EMPLOYEE AVERAGE
                    WHERE    POSSIBLE.DEPT = AVERAGE.DEPT) ;
```

تولیدکنندگانی که بیشتر از میانگین تولید شهر خودشان قطعه تولید کرده اند.

بازیابی از بیش از یک جدول (ادامه)

$$\text{theta} \in \left\{ = \quad \neq \quad < \quad \leq \quad \geq \quad > \right\}$$

❑ امکان $\left\{ \begin{array}{ll} \textbf{theta} & \textbf{ANY} \\ \textbf{theta} & \textbf{SOME} \\ \textbf{theta} & \textbf{ALL} \end{array} \right\}$

- The ANY, ALL and SOME operators are used with a WHERE or HAVING clause.
  - The ANY operator returns true if any of the subquery values meet the condition.
  - The ALL operator returns true if all of the subquery values meet the condition.
  - The SOME operator returns true if at least one of the subquery values meet the condition.
- SOME and ANY are equivalent.

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

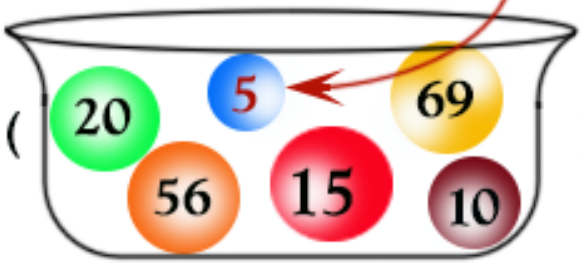
```
1- SELECT   S#
   FROM   S
   WHERE   STATUS < ANY ( SELECT  DISTINCT   STATUS   FROM   S )
```

چون جواب SELECT تک مقداری است نیازی به ANY نیست.

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

بازیابی از بیش از یک جدول (ادامه)

> SOME means greater than at least one value,
> that is, greater than the minimum.

WHERE 70 > SOME (

);

So >SOME (20,56,5,15,69,10) means greater than 5.

So 70 >5 is true, and data returns.

< SOME means less than at least one value,
that is, less than the maximum.

WHERE 70 < SOME (

);

So <SOME (20,56,5,15,69,10) means less than 69.

So 70 < 69 is false, and no data returns.

بازیابی از بیش از یک جدول (ادامه)

نام تهیه کنندگان قطعه ’p2‘ را بدهید:

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
- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns true if the subquery returns one or more records.

روش دهم

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

روش‌های دیگر ؟

تمرین سر کلاسی از پیاده سازی و SQL

فرض کنید پایگاهی با سه جدول زیر داریم.

Students (STID, STFNAME, STLNAME, Gender, STMJRCODE, SupervisorCode)

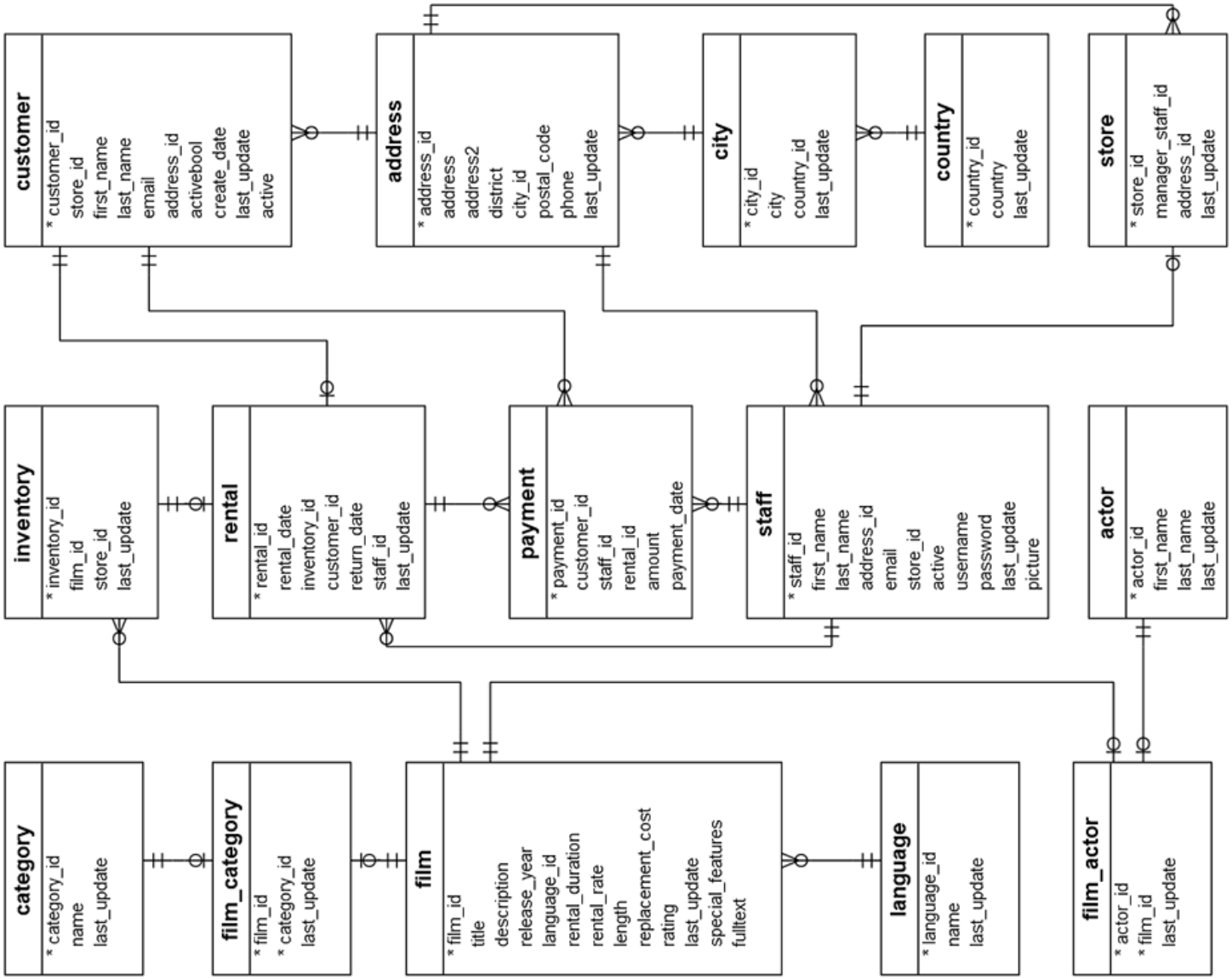
Professors (ProfID, ProfFullName, DepID) جدول مشخصات اساتید

Majors (MajorCode, MajorTitle) جدول کد و عناوین رشته ها

Query لازم برای پرسشهای زیر را به زبان SQL بنویسید:

1. نام کامل و شماره دانشجویی دانشجویانی پسری که ورودی سال ۹۳ هستند (فرض کنید که دو رقم اول شماره دانشجویی نشان دهنده سال ورود است)
2. نام کامل دانشجو، نام کامل استاد راهنمای مربوط به همه دانشجویان
3. نام کامل دانشجو، نام کامل استاد راهنما و عنوان رشته تحصیلی مربوط به همه دانشجویان

مثال


customer
* customer_id
store_id
first_name
last_name
email
address_id
activebool
create_date
last_update
active
address
* address_id
address
address2
district
city_id
postal_code
phone
last_update
city
* city_id
city
country_id
last_update
country
* country_id
country
last_update
store
* store_id
manager_staff_id
address_id
last_update
inventory
* inventory_id
film_id
store_id
last_update
rental
* rental_id
rental_date
inventory_id
customer_id
return_date
staff_id
last_update
payment
* payment_id
customer_id
staff_id
rental_id
amount
payment_date
staff
* staff_id
first_name
last_name
address_id
email
store_id
active
username
password
last_update
picture
actor
* actor_id
first_name
last_name
last_update
category
* category_id
name
last_update
film_category
* film_id
* category_id
last_update
film
* film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
fulltext
language
* language_id
name
last_update
film_actor
* actor_id
* film_id
last_update

There are 15 tables in the DVD Rental database:

- actor – stores actors data including first name and last name.
- film – stores films data such as title, release year, length, rating, etc.
- film_actor – stores the relationships between films and actors.
- category – stores film’s categories data.
- film_category- stores the relationships between films and categories.
- store – contains the store data including manager staff and address.
- inventory – stores inventory data.
- rental – stores rental data.
- payment – stores customer’s payments.
- staff – stores staff data.
- customer – stores customers data.
- address – stores address data for staff and customers
- city – stores the city names.
- country – stores the country names

1) Retrieve the films with rating of PG or G.

SELECT * FROM film WHERE rating='G' or rating='PG' ;→ return 372 rows

PostgreSQL:

SELECT * FROM film WHERE **cast** (rating AS varchar) like '%G%'; → return 595 rows

SELECT * FROM film WHERE (rating **::** varchar) like '%G%'; → return 595 rows

Cast is required because, rating is defined as *Enumeration Type* as {G, PG, PG-13, R, NC-17}

```
CAST ( expression AS target_type );
```

or

```
expression::type
```

```
SELECT
  '100'::INTEGER,
  '01-OCT-2015'::DATE;
```

	int4 integer	date date
1	100	2015-10-01

2) Retrieve customer's *full name* and all *its payments*

```
SELECT cu.first_name || ' ' || cu.last_name as FullName, pa.amount FROM payment pa join customer cu using (customer_id)
```

3) Retrieve the *full name* and *total payments* of each customer, sorted by their total payments

```
SELECT cu.first_name || ' ' || cu.last_name as FullName, sum(pa.amount) as sumsp FROM payment pa join customer cu using (customer_id) group by FullName order by sumsp desc
```

5) Retrieve the *full name* and *total payments* of the customers that their total payment are more than 100$

```
SELECT cu.first_name || ' ' || cu.last_name as FullName, sum(pa.amount) as sumsp FROM payment pa join customer cu using (customer_id) group by FullName having sum(pa.amount)>100 order by sumsp desc
```

6) Retrieve all payments of each film.

```
SELECT film.title,   payment.amount
FROM payment join rental using (rental_id) join inventory using (inventory_id) join film
using (film_id)
order by film.title
```

7) Retrieve the film's title and its total rental's amount.

```
SELECT film.title,   sum(payment.amount )
FROM payment join rental using (rental_id) join inventory using (inventory_id) join film
using (film_id) group by film.title
order by film.title
```

دیگر امکانات SQL

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

پرسش و پاسخ . . .

ایمیل : zarepour@iust.ac.ir

ارتباط حضوری: ساعت مشخص شده در برنامه هفتگی به عنوان رفع اشکال دانشجویی (روزهای یکشنبه و سه شنبه ساعت ۱:۳۰ تا ۳ بعد ظهر)

www.ezarepour.ir