



سپاه پاسخ  
زیر مجموعه  
نیازمندی

## یادداشت‌های تکمیلی

(سری ۲)

۱- تفاوت‌های رابطه با جدول

۲- رابطه‌ی درجه‌ی صفر

۳- درباره‌ی طراحی

۱-۳ مراحل اساسی

۲-۳ خصوصیات طراحی خوب

۴- مثال‌هایی از قواعد جامعیت

۵- جبر رابطه‌ای

۱- خواص عملگرها

۲- مجموعه‌ی کامل عملگرها

۳- مثال‌ها

۴- مثال‌های دیگر

۶- حساب رابطه‌ای

۱- روابط همارزی

۲- مثال‌ها

۳- مثال‌های دیگر

۷- تمرینات تکمیلی

۱- صورت پرسش‌ها

۲- پاسخ در جبر رابطه‌ای

۳- پاسخ در حساب رابطه‌ای

۸- درباره‌ی تئوری وابستگی

۹- دلایل برز افزونگی

# B - course(1)

بازار رابطہ

## Relations vs. Tables

-تفاوتی رابطہ باہمی

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

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

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

What are the column names in the result of this query?

5. The rows of a table have a top-to-bottom ordering, but the tuples of a relation do not.
6. A table might contain duplicate rows, but a relation does not contain duplicate tuples.

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

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

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

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

DATE 03 : 2023 \*

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

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

## Relations with No Attributes

relations without attributes :

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

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

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

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

In Tutorial D, the expressions TABLE\_DEE and TABLE\_DUM can be used as short-hand for the relation selector invocations

RELATION { } { TUPLE { } }

and

RELATION { } { }

respectively.

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

## مثال‌هایی از قواعد جامعیت



### • دستور تعریف دامنه (میدان)

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

مثال ▪

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

### • دستور تعریف "اظهار"

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

مثال ۱ :

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

مثال ۲ : رابطه‌ی R (A,B, ...)

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

مثال ۳ : رابطه‌ی R (A, B, C, D) مفروض است.

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

مثال ۴ :

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

مثال ۵ :

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

## • دستور تعریف رهانای (TRIGGER)

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

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

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

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

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

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

```
EMPSALAUG (EMPID , AUGM)
C.K.
```

صفت AUGM مقدار افزایش حقوق کارمند است.  
رهانای زیر سازگاری داده‌ای پایگاه را پس از عمل بهنگام سازی تضمین می‌کند.

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

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

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

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

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

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

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

(b)

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

تمرين: عکس در میرک لذتمنی "رهانا" را اندوخته (حکایت).

مثال ۴: کاربرد رهانادر عیایت لز دید در پایگاه داده : اعمال "محدودیت دید" :  
فرض میکنیم دید  $\vee$  چنین تعریف شده باشد :

CREATE VIEW V

```
AS SELECT STID, STNAME, STDEG, STDEID
FROM STT
WHERE STMJR = 'Comp';
```

اگر کاربر بخواهد تابعی مزایی دید در پایگاه داده درج کند، در رابطه هنباش STT بر اساس ست STMJR = "Computer" و یا "تماریس زناده" ولردیم سود، خلاصه مقاله پیش نمایه "Comp" بناسنده، در بازنایی لز دید  $\vee$  کاربر تاپل درج شده را خواهد دید. بر ارض فرع مکمل لز رهانا راستفاده میکنیم. به صورت زیر:

CREATE TRIGGER IT001

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

با این ترتیب درج مکمل شد. مثلاً < 999, Stg, b8, D19 >

در دفعه  $\vee$ ، مخفی درج تاپل نماید در اینجا دستور:

< 999, Stg, b8, 'Comp', D19 >

و با نتیجه مکمل جدید در بازنایی لز دید  $\vee$ ، دیده خواهد شد.

در این مثال ماعده C2 در عمل حفظ اعمال میشود.

مثال ۵:  
CREATE TRIGGER COTDELTRIG  
BEFORE DELETE ON COT  
REFERENCING OLD AS OLCO  
(DELETE FROM STCOT

```
WHERE STCOT.COID = OLCO.COID
FOR EACH ROW);
```

CREATE TRIGGER STCOTITRIG

BEFORE INSERT ON STCOT

REFERENCING NEW AS NSTCOT

(WHEN (NOT EXISTS (SELECT \* FROM COT WHERE COT.COID =  
NSTCOT.COID)))

ABORT TRANSACTION FOR EACH ROW);

مثال ۶:

در این مثال، ماعده C2 در عمل حفظ اعمال میشود.

در عمل درج اعمال میشود.

FOR EACH STATEMENT  $\sqrt{ } \vdash : v J \alpha$

دار رضا خان میرزا، نهدشت زیر امروزی است:

"سیانلین حقوق کاربران نباید لزمه، رسیل کتر باشد."

چون با یک دستور درج یا بینجام سازی ممکن است تعدادی سطر درج شوند و با بینجام در آنها، در اینجا لزین تغییرات در رابطه EMPh ممکن است می‌شوند حقوق روز اخیر را موقتاً کمتر مسود و لزی پن لز پایان اجرای دستور درج یا بینجام سازی از نظر ریال بسته مسود.

سیا برانی اگر بخواهیم می‌رویست داده شده را بگذر رهانا کسرل

FOR EACH STATEMENT justify.

اوستفاده کنیم. رهانیت زیرین کار را در عمل نهاده می‌نماییم

ریاضیات

CREATE TRIGGER AVSALTRIG

AFTER UPDATE OF ESAL ON EMPL

## REFERENCING

OLD AS OEMPL, NEW AS NEMPL

**FOR EACH STATEMENT**

WHEN 'a' > (SELECT AVG(ESAL)  
                FROM EMPL)

BEGIN

DELETE FROM EMPH

WHERE (EMID,ENAME,...,EJOB,ESAL)

IN NEMPL;

INSERT INTO EMPL

SELECT \* FROM OEMPL;

END ;

# جبر را بصر

## ۹- خواص عملگرها



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

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

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

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

و نیز فرض کنیم:  $r$ ،  $p$ ،  $q$ ،  $p_{Atts}$  و  $q_{Atts}$  به ترتیب مجموعه صفات استفاده شده در این مستندات باشند و  $N$ ،  $M_1$ ،  $L_1$ ،  $L_2$ ،  $L_3$  و  $M_2$  هر یکی مجموعه از صفات باشند.

خواص زیر در عملگرهای جبر رابطه ای برقرارند [CONN2002]

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

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

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

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

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

$$R \times S = S \times R$$

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

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

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

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

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

# B

$$8) \prod_{\langle L_1 \cup L_2 \rangle} (R \bowtie_r S) = (\prod_{\langle L_1 \rangle} (R)) \bowtie_r (\prod_{\langle L_2 \rangle} (S))$$

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

این خاصیت در حالت ضرب کارتزین  $R \times S$  نیز برقرار است.

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

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

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

توجه! در اینجا منظور عملگر پیوند طبیعی است.

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

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

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

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

$$R \cap S = S \cap R$$

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

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

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

$$12) \prod_{\langle L \rangle} (R \cup S) = \prod_{\langle L \rangle} (S) \cup \prod_{\langle L \rangle} (R)$$

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

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

۴- مجموعه کامل علکر کریبر ارائه از:

• - : MINUS

• Ⓜ : RESTRICT

• X : TIMES

• Π : PROJECT

• ∪ : UNION

RDB :  $\left\{ \begin{array}{l} S(S\#, SNAME, STATUS, CITY) \\ P(P\#, PNAME, COLOR, WEIGHT, CITY) \\ SP(S\#, P\#, QTY) \end{array} \right.$

۳- مطالعی لزجبر ابطایی :

نام شرکت مانی را بمحض که پرداخت نکن :  $Q_1$

$((SP \text{ JOIN } S) \text{ where } P\# = P_2')[SNAME]$

$\prod_{\langle SNAME \rangle} (\delta_{\langle P\# = P_2 \rangle} (SP \bowtie S))$  و با خارجی پایه :

نام شرکت مانی که قطعه قرمز را صادر کرده باشد :  $Q_2$

$((P \text{ where } COLOR = 'Red') \text{ JOIN } SP)[S\#] \text{ JOIN } S)[SNAME]$

نام شرکت مانی که قطعه که را صادر کرده باشد :  $Q_3$

$((SP[S\#, P\#] \text{ DIVIDE BY } P[P\#]) \text{ JOIN } S)[SNAME]$

نام شرکت مانی که قطعه که را صادر کرده باشد :  $Q_4$

$SP[S\#, P\#] \text{ DIVIDE BY } (SP \text{ where } S\# = S_2')[P\#]$

نام شرکت مانی که قطعه که را صادر کرده باشد :  $Q_5$

$((S[S\#] \text{ MINUS } (SP \text{ where } P\# = P_2'))[S\#] \text{ JOIN } S)[SNAME]$

EXTEND  $\neq \otimes$

1) EXTEND  $S$  ADD 'supplier' AS TAG.

2)  $(\text{EXTEND } S \text{ ADD CITY AS } SCITY)[S\#, SNAME, STATUS, SCITY]$ .

$S$  RENAME CITY AS SCITY : مدل رسمی

3) EXTEND  $(P \text{ JOIN } SP)$  ADD (WEIGHT \* QTY)  
AS SHIPWT.

SUMMARIZE  $\neq \otimes$

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

- 2) SUMMARIZE SP BY (S#) ADD COUNT AS NP.
- 3) SUMMARIZE (P JOIN SP) BY (CITY) ADD COUNT AS NSP.

CITY	NSP
C1	5
C2	6
C3	1

مذکور میں حصہ :

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

: Aggregate operators جمعیاتی اپریٹور

..... , AVG , MAX , MIN , SUM , COUNT : اضافی

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

..... ، تعداد کے نام سے COUNT اضافی

SUM ( SP WHERE S# = 31 , QTY ) : جمعیاتی

? subquery UNGROUP , GROUP : [ (x, y) ] اضافی

$$1) (R_1 \div R_2) \times R_2 \subseteq R_1 \quad : \text{موجب} \leftarrow$$

$$2) R_1 (x, y) \div R_2 (y) = \Pi_{\langle x \rangle} (R_1) - \left( \Pi_{\langle x \rangle} \left( \Pi_{\langle x \rangle} (R_1) \times R_2 - R_1 \right) \right)$$

RDB:  $\left\{ \begin{array}{l} STT(STID, STNAME, STDEG, STMJR, STDEID) \\ COT(COID, COTITLE, COTYPE, CREDIT, COPEID) \\ STCOT(STID, COID, TR, YRYR, GRADE) \end{array} \right.$

B)

۴- مساله‌ی دیگر رله جبر رابطه‌ای (بانماده‌ی ریاضی)

- عنوان درس‌ای را بدهید که دانشجوی با ساره 'N' در ترم اول ۸۲-۸۳ انتخاب کرد.  $Q_1$

$\prod_{\langle COTITLE \rangle} ( \exists_{STID=N \wedge TR=1 \wedge YRYR=82-83} (STCOT) \bowtie COT )$

- نام دانشجویانی را بدهید که در درس 'B' با عنوان 'C' در ترم دوم ۸۱-۸۲ مردود شوند.  $Q_2$

$\prod_{\langle STNAME \rangle} ( \exists_{COTITLE=C \wedge TR=2 \wedge YRYR=81-82 \wedge GRADE < 10} (STCOT \bowtie COT) \bowtie STT )$

- نام و رشته دانشجویانی را بدهید که درس علی در ترم اول ۸۲-۸۳ انتخاب کردند.  $Q_3$

$\prod_{\langle STNAME, STMJR \rangle} ((\exists_{COTYPE='f'} (COT) \bowtie \exists_{TR=1 \wedge YRYR=82-83} (STCOT)) \bowtie STT)$

- عنوان درس‌ای عملی را بدهید که دانشجویان گروه آموزشی 'D' در سال ۸۱-۸۲ انتخاب نکردند.  $Q_4$

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

- عنوان درس‌ای را بدهید که تمام دانشجویان رشته کامپیوتر در ترم دوم ۸۱-۸۲ در زیر قبول شده باشند.

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

تمام:

۱- پرسش  $Q_4$  را با علّه SEMIMINUS بتوانیم.

۲- عنوان هر رشته و نهاد دانشجویان پرداخته را بدهید.

۳- پرسش  $Q_5$  را بدون استفاده از زمان تقسیم نمایم.

## حساب رابطه ای

۱- روابط هم‌ارزی بین سورکر وجودی و عدمی :

- $\text{FORALL } T(f) \equiv \text{NOT EXISTS } T(\text{NOT } f)$
- $\text{EXISTS } T(f) \equiv \text{NOT } (\text{FORALL } T(\text{NOT } f))$
- $\text{FORALL } T((f) \text{ AND } (g)) \equiv \text{NOT EXISTS } (\text{NOT}(f) \text{ OR } \text{NOT}(g))$
- $\text{FORALL } T((f) \text{ OR } (g)) \equiv \text{NOT EXISTS } (\text{NOT}(f) \text{ AND } \text{NOT}(g))$
- $\text{EXISTS } T((f) \text{ OR } (g)) \equiv \text{NOT FORALL } T(\text{NOT}(f) \text{ AND } \text{NOT}(g))$
- $\text{EXISTS } T((f) \text{ AND } (g)) \equiv \text{NOT FORALL } T(\text{NOT}(f) \text{ OR } \text{NOT}(g))$
- $\text{FORALL } T(f) \Rightarrow \text{EXISTS } T(f)$  و نزدیکی :
- $\text{NOT EXISTS } T(f) \Rightarrow \text{NOT FORALL } T(f)$

لیگاتور [ اختصار ] :

، "iterated OR" معرفت کرده است،  $\exists$  معرفت معرفت "iterated AND" معرفت کرده است،  $\forall$  معرفت کرده است. این مطلب یعنی چه؟

برای مطالعه بیشتر [ اختصار ] : الگوریتم کامپیس ماد (Codd's reduction alg.)  
مطالعه مسود ( این الگوریتم برآورده بر عبارت دلخواه حساب روابطه ای را به عبارت مبدل آن در جهت رابطه ای نمایی کند )

- \*  $SX, PX, SPX, SPY$ :  $\exists_{S_1} \forall_{P_1} \exists_{SPX} \exists_{SPY}$  :  $SX \cdot S\# = SPX \cdot S\# \text{ AND } SX \cdot P\# = P_1 \text{ AND } SPX \cdot P\# = SPY \cdot P\# \text{ AND } SPY \cdot S\# = S_1$  :  $Q_1$ .
- $SX \cdot S\# \text{ where } SX \cdot CITY = 'C_2' \text{ AND } SX \cdot STATUS > 20$  :
- $SX \cdot SNAME \text{ where } \exists_{SPX} SPX ( SPX \cdot S\# = SX \cdot S\# \text{ AND } SPX \cdot P\# = P_2 )$  :  $Q_2$ .
- $SX \cdot SNAME \text{ where } \exists_{SPX} SPX ( SX \cdot S\# = SPX \cdot S\# \text{ AND } \exists_{PX} ( PX \cdot P\# = SPX \cdot P\# \text{ AND } PX \cdot COLOR = 'Red' ) )$  :  $Q_3$ .
- $SX \cdot SNAME \text{ where } \exists_{SPX} \exists_{SPY} ( SX \cdot S\# = SPX \cdot S\# \text{ AND } SPX \cdot P\# = SPY \cdot P\# \text{ AND } SPY \cdot S\# = S_2 )$  :  $Q_4$ .
- $SX \cdot SNAME \text{ where } \exists_{SPX} \exists_{SPY} ( SX \cdot S\# = SPX \cdot S\# \text{ AND } SPX \cdot P\# = SPY \cdot P\# \text{ AND } SPY \cdot S\# = S_2 )$  :  $Q_5$ .
- $SX \cdot SNAME \text{ where } \forall_{PX} ( \exists_{SPX} SPX ( SPX \cdot S\# = SX \cdot S\# \text{ AND } SPX \cdot P\# = PX \cdot P\# ) )$  :  $Q_6$ .
- $SX \cdot SNAME \text{ where } \neg \exists_{SPX} PX ( \neg \exists_{SPY} SPY : SPX \cdot S\# = SX \cdot S\# \text{ AND } SPX \cdot P\# = SPY \cdot P\# )$  :  $Q_7$ .
- $SX \cdot SNAME \text{ where } \neg \exists_{SPX} SPX ( SPX \cdot S\# = SX \cdot S\# \text{ AND } SPX \cdot P\# = P_2 )$  :  $Q_8$ .

۳- مسائلی دیگر لزحاب مرتبط ای

تغییراتی تاییدی  $STCO \cdot CO \cdot ST$  تعریف شده به ترتیب روش مرتبط ای  $STCO \cdot COT$  و  $COT$  مفروضه.

- سفاره و عنوان درسی دو ماهی را بخواهی  $Q_1$   
 $(CO \cdot COID, CO \cdot COTITLE)$  where  $CO \cdot CREDIT = '2'$ ;

- عنوان درسی را بخواهی که دانشجوی با شماره 'N' در ترم اول ۸۲-۸۳ انتخاب کرد  $Q_2$   
 $CO \cdot COTITLE$  where EXISTS  $STCO$  ( $STCO \cdot COID = CO \cdot COID$   
 $AND STCO \cdot STID = 'N'$  AND  $STCO \cdot TR = '1'$  AND  $STCO \cdot YRYR = '82-83'$ )

- نام دانشجویی را بخواهی که درس با شماره 'C' را انتخاب نکرده باشد  $Q_3$   
 $ST \cdot STNAME$  where NOT EXISTS  $STCO$  ( $STCO \cdot STID = ST \cdot STID$   
 $AND$   
 $STCO \cdot COID = 'C'$ )

- نام و رشته دانشجویی را بخواهی که درس علی در ترم اول ۸۲-۸۳ انتخاب کرد  $Q_4$   
 $ST \cdot STNAME, ST \cdot STMJR$  where EXISTS  $STCO$  ( $ST \cdot STID = STCO \cdot STID$   
 $AND STCO \cdot TR = '1'$  AND  $STCO \cdot YRYR = '82-83'$  AND EXISTS  $CO$   
 $(CO \cdot COID = STCO \cdot COID$  AND  $CO \cdot COTYPE = 'X'$ )) .

- عنوان درسی را بخواهی که تمام دانشجویان رشته کامپیوتر در ترم دوم ۸۱-۸۲ را دریافت  
 جدول سه ماهه باشند  $Q_5$ .

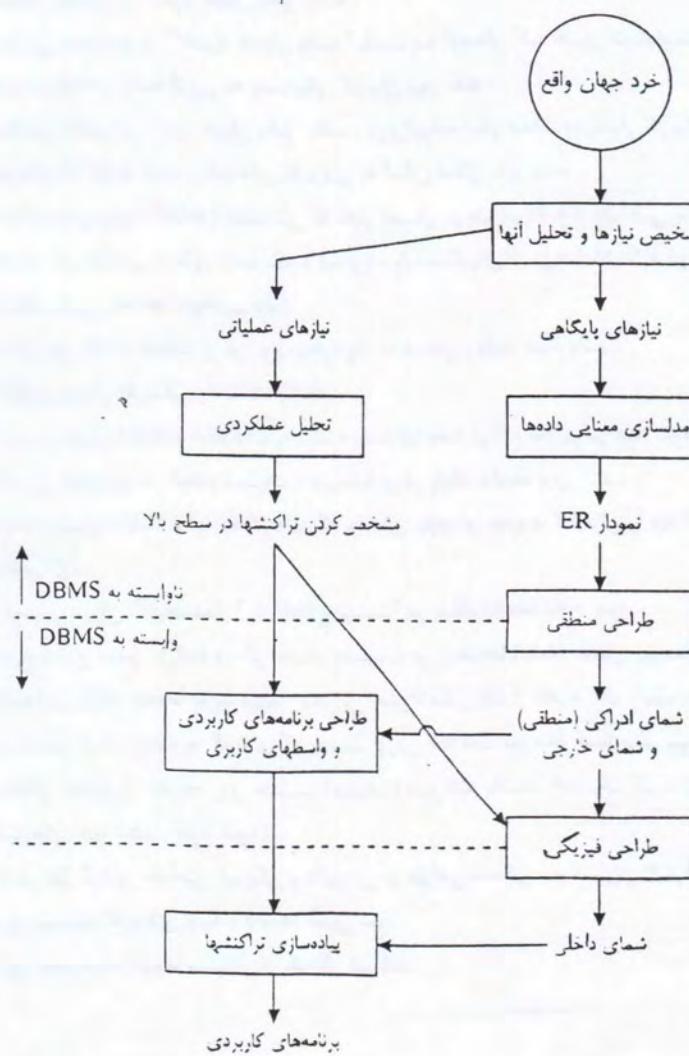
$CO \cdot COTITLE$  where FOR ALL  $ST$  (EXISTS  $STCO$  (  
 $STCO \cdot STID = ST \cdot STID$  AND  $STCO \cdot COID = CO \cdot COID$   
 $AND STCO \cdot TR = '2'$  AND  $STCO \cdot YRYR = '82-83'$   
 $AND STCO \cdot GRADE \geq '10'$  AND  $ST \cdot STMJR = 'comp'$ ))

• حین: پرسش  $Q_5$  این استفاده از نوشتة سود.

# طراحی ...

۱- مراحل اساسی طراحی پایگاه داده (سده سده)

بر از جریان طاری وظایف DBA (در مادرست رشته) مراجعه شود.



## ۴ - خصوصیات طراحی خوب

طراحی متعاقن خوب باید خصوصیات زیر را داشته باشد:

- ۱- نهایی واضح از "خرد جهان واقع" باشد.
- ۲- نهایی صحیح از "خرد جهان واقع" باشد.
- ۳- نهایی در پاسخگویی به برآمدهای کاربران بروز نکند.
- ۴- تمام جامی از "خرد جهان واقع" باشد. در برگیرنده تمام دنیات موردنیاز کاربران، به گونه‌ای که تولید همه برنامه‌های کاربردی به آسانی امکان پذیر باشد.
- ۵- تمام محدودیت‌ها (قواعد) جامعه‌ی که قابل اعمال در هر مرحله از طراحي منصفی باشد، در طراحی منظور شده باشد (به ویژه واسیجیاتی بین صفات که در بحث فوایل سازی رابطه‌ها خواهیم دید).
- ۶- کترین میزان افرادیکی را داشته باشد.
- ۷- کترین میزان اختلاط اطلاعات را داشته باشد (به بعث نرم‌الترسازی مراجعه شود).
- ۸- کترین دشواری در انجام عملیات ذخیره‌سازی در پایگاه داده‌ها بروز کند.
- ۹- انتقال بین‌بری داشته باشد به گونه‌ای که بتوان نیازهای جدید کاربران را به آسانی منظور کرد.
- ۱۰- کترین میزان "چیزگذر" یا "اطلاع نیست" در پایگاه داده‌ها ایجاد نشود.
- ۱۱- هیچ اطلاع حسنه (الله) در اینجا عملیات در رابطه‌ها (شلا عسل پیشنهاد دو رابطه) در پایگاه داده‌ها پدید نیاید. بایاری حقیقت امکان پاید از تجزیه یک رابطه به دو یا بیش از دو رابطه به گونه‌ای که بدهست از دن اطلاعات مورد نظر نیازمند بیووند رابطه‌ای حاصل از تحریره روی صفت (حصت) غیر کلید باشد. احتساب کرد (به مختارهای جبار دهم روحش شود).
- ۱۲- بادر نظر گرفتن طراحی فیزیکی و تاثیر آن در طراحی منطقی، پیشتر کار آیند برای سیستم کاربردی پایگاه داده‌ها نامیم شود.
- بعضی از این خصوصیات لزوماً مستقل از یکدیگر نیستند.

کنیادس: آرای خصوصیات یکجا هم مطروح است؟

# B-course(1) مركبات - نكيلى درجبر وحساب

## Query Exercises

أ- صورت درس

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

```

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

```

- 6.13 Get full details of all projects.
- 6.14 Get full details of all projects in London.
- 6.15 Get supplier numbers for suppliers who supply project J1.
- 6.16 Get all shipments where the quantity is in the range 300 to 750 inclusive.
- 6.17 Get all part-color/part-city combinations. *Note:* Here and subsequently, the term "all" is to be taken to mean "all currently represented in the database," not "all possible."
- 6.18 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are all colocated (i.e., all in the same city).
  
- 6.19 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are not all colocated.
- 6.20 Get all supplier-number/part-number/project-number triples such that no two of the indicated supplier, part, and project are colocated.
- 6.21 Get full details for parts supplied by a supplier in London.
- 6.22 Get part numbers for parts supplied by a supplier in London to a project in London.
- 6.23 Get all pairs of city names such that a supplier in the first city supplies a project in the second city.
- 6.24 Get part numbers for parts supplied to any project by a supplier in the same city as that project.
- 6.25 Get project numbers for projects supplied by at least one supplier not in the same city.
- 6.26 Get all pairs of part numbers such that some supplier supplies both the indicated parts.
- 6.27 Get the total number of projects supplied by supplier S1.
- 6.28 Get the total quantity of part P1 supplied by supplier S1.
- 6.29 For each part being supplied to a project, get the part number, the project number, and the corresponding total quantity.
- 6.30 Get part numbers of parts supplied to some project in an average quantity of more than 350.
- 6.31 Get project names for projects supplied by supplier S1.
- 6.32 Get colors of parts supplied by supplier S1.
- 6.33 Get part numbers for parts supplied to any project in London.
- 6.34 Get project numbers for projects using at least one part available from supplier S1.

- 6.35** Get supplier numbers for suppliers supplying at least one part supplied by at least one supplier who supplies at least one red part.
- 6.36** Get supplier numbers for suppliers with a status lower than that of supplier S1.
- 6.37** Get project numbers for projects whose city is first in the alphabetic list of such cities.
- 6.38** Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.
- 6.39** Get supplier numbers for suppliers supplying some project with part P1 in a quantity greater than the average shipment quantity of part P1 for that project.
- 6.40** Get project numbers for projects not supplied with any red part by any London supplier.
- 6.41** Get project numbers for projects supplied entirely by supplier S1.
- 6.42** Get part numbers for parts supplied to all projects in London.
- 6.43** Get supplier numbers for suppliers who supply the same part to all projects.
- 6.44** Get project numbers for projects supplied with at least all parts available from supplier S1.
- 6.45** Get all cities in which at least one supplier, part, or project is located.
- 6.46** Get part numbers for parts that are supplied either by a London supplier or to a London project.
- 6.47** Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.
- 6.48** Get all pairs of supplier numbers, Sx and Sy say, such that Sx and Sy supply exactly the same set of parts each. (Thanks to a correspondent, Fatma Mili of Oakland University, Rochester, Michigan,

gan, for this problem. For simplicity, you might want to use the original suppliers and parts database for this exercise, instead of the expanded suppliers-parts-projects database.)

- 6.49** Get a “grouped” version of all shipments showing, for each supplier-number/part-number pair, the corresponding project numbers and quantities in the form of a binary relation.
- 6.50** Get an “ungrouped” version of the relation produced in Exercise 6.49.

6.13 J

6.14 J WHERE CITY = 'London'

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

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

6.17 P { COLOR, CITY }

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

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

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

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

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

```
WITH ( S WHERE CITY = 'London' ) AS T1,  

      ( J WHERE CITY = 'London' ) AS T2,  

      ( SPJ JOIN T1 ) AS T3,  

      T3 { P#, J# } AS T4,  

      ( T4 JOIN T2 ) AS T5 :  

      T5 { P# }
```

Here is the same query without using WITH:

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

```

What does this query return if relvar J is empty?

```

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

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

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

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

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

6.43 S { S#, P# } DIVIDE BY J { J# } PER SPJ { S#, P#, J# }

```

Q-F

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

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

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

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

**6.48** We show two solutions to this problem.

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

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

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

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

```

T19

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

```

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

```

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

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

SPQ UNGROUP JQ

٤- بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

7.13.13 JX

7.13.14 JX WHERE JX.CITY = 'London'

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

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

7.13.17 (PX.COLOR, PX.CITY)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Q - V

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
RANGEVAR RX RANGES OVER R
RANGEVAR RY RANGES OVER RX.JQ ;
( RX.S#, RX.P#, RY.J#, RY.QTY )
```

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

## در باره تئوری وابستگی

I- قواعد (اسیوری) آسترزنگ در مورد FD :

- if  $B \subseteq A$  Then  $A \rightarrow B$  : قاعده انعکاس
- if  $A \rightarrow B$  and  $B \rightarrow C$  Then  $A \rightarrow C$  : قاعده تعددی (ترانزیتی)
- if  $A \rightarrow B$  Then  $(A, C) \rightarrow (B, C)$  : قاعده افزایش
- if  $A \rightarrow (B, C)$  Then  $A \rightarrow B$  and  $A \rightarrow C$  : قاعده تجزیه
- if  $A \rightarrow B$  and  $C \rightarrow D$  Then  $(A, C) \rightarrow (B, D)$  : قاعده ترکیب
- if  $A \rightarrow B$  and  $A \rightarrow C$  Then  $A \rightarrow (B, C)$  : قاعده اجتماع
- if  $A \rightarrow B$  and  $(B, C) \rightarrow D$  Then  $(A, C) \rightarrow D$  : قاعده شبیه تعددی
- if  $A \rightarrow B$  and  $C \rightarrow D$  Then  $AU(C-B) \rightarrow (B, D)$  : قاعده گلانگی هجومی (توسط Hugh DARWEN اثبات شده است)

نکته: قواعد ۱، ۲ و ۳ رسمی و کامل هستند، به این معنا که با داشتن این چهار قاعده از وابستگی کسی تابعی F، تمام وابستگی تابعی منطقاً کابل استنتاج لذ F، با همین سه قاعده پر دسته هستند آنقدر و همچو وابستگی تابعی دیگر (که قابل استنتاج از F نباشد) نیز پر دسته نمی‌آیند.

توجه: سه قاعده اول به آسانی قابل اثبات زند و قواعد دیگر لذ روسی همان روابط را نهادند.

## II- کاربردی قواعد آسترزنگ

▪ یافتن بستارگی صفت:  $A^+$

▪ یافتن بستارگی تجویه از وابستگی کسی تابعی:  $F^+$

▪ یافتن تجویه طبقه نایابه از وابستگی کسی تابعی: تابعی کم رابطه (تجویه بینه)

▪ یافتن پوشش کافی نسبت

▪ یافتن کلید (کلید) رابطه

▪ روابط خوب نودن بجزء رابطه

- جمع FD در آن افزونه نباشد.

- مجموعہ کا ہیں ناپذیر : ۲- سوت راست پر FD، صفت سادہ باشد۔

۴- سمت پهیز FD کاہن نانڈر بابو

IV - پوش کافونک : ۷-۱ - سنت چپ یر FD کاہن ماندر بارہ.

۲- در همین در FD، سمت چپ تکیه نباشد. مجموعه‌ای است زیر FD که در آن

۷: کاربرد بعض نفاہیم:

۱- یافتن سرمه کلیند و کلیند کانزید

### ۲- روابط وجودیک FD مفروض در $F^+$

۲: کاربرد  $F^+$  : سرط : ۱- تئیس معادل بودت دو مجموعه از  $FP$  :

۲- تشخیص لازم نه بود. لک

۳- تشخیص پوسش یک محبوب از  $F_D$  :  $F \rightarrow F_D$  یا  $F$  را  $F_D$  باشد.

۵- مطبوعه خبر مستقیم : در مایهه تجویه کا شن نایذر و یوس کانویک

لکچہارمی: کاربردی مجموعہ کاہیں نایڈر، یوسٹ کانوننک؟

۲- آیا بحبوحہ مابھس نایڈر ویوں کا فونٹ کیتا ہے ؟

لوجه: تعداد FD کی مکن در تابع رابطه درجه  $m$ ، را از لاست با  $2^{n+1}$  (جواب)

نوعیه: متوات FD بین دو صفت ردیگر همانس "اطمار" بیان کرد:

فرض:  $B \rightarrow C$ ,  $R(\bar{A}, B, C, \dots)$

CREATE ASSERTION CTOBFD

CHECK (NOT EXISTS (SELECT B  
FROM R

FROM R

## GROUP BY B

HAVING MAX(c) ≠ MIN(c));

تمرين: الگوریتم ماختن سری محلید و الگوریتم ماختن مکعب کاچه را بنویسید.

توضیح: وابستگی تابعی  $C \rightarrow B \rightarrow C$  در مجموعه  $R$  با صفات روابطی ای هم بیان کرد:

CONSTRAINT CTOBFD

فرض:  $R_1, R_2$ : دو تئوری تابعی تعریف

FORALL  $R_1$  ( FORALL  $R_2$  (

شده دری روابط

IF  $R_1 \cdot B = R_2 \cdot B$  THEN

$R_1 \cdot C = R_2 \cdot C$  ));

فرض:  $R_1, R_2$ : دو تئوری تابعی تعریف

شده دری روابط

تمرين: الگوریتم ماشین  $A^+$  ، الگوریتم حذف FD از زمرة را تنوییه.

(B-1)

مثال: در رابطه  $R(A, B, C, D)$

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

مجموعه اصلی  $FD$  حذف نماین:  $A \rightarrow B, B \rightarrow C, A \rightarrow D$

برای:

$$A \rightarrow C, A \rightarrow B : \text{برای } A \rightarrow (B, C) \quad \text{و} \quad 1$$

$$C \rightarrow D \Rightarrow A \rightarrow (A, C) : \text{پس } (A, A) \rightarrow (A, C) : \text{برای } A \rightarrow C \quad \text{و} \quad 2$$

$$A \rightarrow D : \text{برای}$$

$$(A, B) \rightarrow C : \text{برای } A \rightarrow C \quad \text{و} \quad 3$$

$(A, C) \rightarrow D$  پس وابستگی تابعی  $A \rightarrow C$  هم داشت:  $B \rightarrow C, A \rightarrow B$  و  $A \rightarrow C$  و  $(A, B) \rightarrow C$  از زمرة حذف نمود.

$$F_1 = \left\{ \begin{array}{l} B \rightarrow D \\ E \rightarrow C \\ (A, C) \rightarrow D \\ (C, D) \rightarrow A \\ (B, E) \rightarrow A \end{array} \right\}$$

تمرين: در مجموعه اصلی  $R$  روابطی ای بیان کرد:  $A \rightarrow (C, D, E)$

$$F_2 = \left\{ \begin{array}{l} B \rightarrow (C, E) \\ (A, D) \rightarrow E \\ (C, D) \rightarrow F \\ (B, D) \rightarrow A \\ (C, E, D) \rightarrow (A, B, D) \end{array} \right\}$$

تمرين:  $(C, D) \rightarrow B \Rightarrow (A, D) \rightarrow C$ : برای  $R(A, B, C, D)$  روابطی ای بیان کرد:  $(A, D)^+$  و  $(A, D)^-$  شود آنرا  $(A, D)^+$  نویسید.

توضیح: وابستگی تابعی  $C \rightarrow B \rightarrow C$  را که توان با صاب رابطه ای هم بیان کرد:

CONSTRAINT CTOBFD

فرض:  $R_2, R_1$ : دو تغییر تابعی تعریف

FORALL  $R_1$  (FORALL  $R_2$  (

$\exists$  مجموعه رابطه

IF  $R_1 \cdot B = R_2 \cdot B$  THEN

$R_1 \cdot C = R_2 \cdot C$  ));

تمرين: الگوریتم پائس  $A^+$  و الگوریتم حذف FD از زیر نویسید.

(B-1)

مثال: در رابطه  $R(A, B, C, D)$

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

مجموعه اصلی  $FD$  چنین است:  $\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

برای:

$$A \rightarrow C, A \rightarrow B : \text{پس } A \rightarrow (B, C) \quad \text{J-1}$$

$$C \rightarrow D : \text{پس } A \rightarrow (A, C) : \text{پس } (A, A) \rightarrow (A, C) : \text{پس } A \rightarrow C \quad \text{J-2}$$

$$A \rightarrow D : \text{پس}$$

$$(A, B) \rightarrow C : \text{پس } A \rightarrow C \quad \text{J-3}$$

$(A, C) \rightarrow D$  میں وابستگی تابعی  $A \rightarrow C$  است:  $A \rightarrow C : \text{پس } B \rightarrow C, A \rightarrow B \quad \text{J-4}$

از زیر نویسید و حذف شود:  $A \rightarrow C, (A, B) \rightarrow C,$

$$F_1 = \left\{ \begin{array}{l} B \rightarrow D \\ E \rightarrow C \\ (A, C) \rightarrow D \\ (C, D) \rightarrow A \\ (B, E) \rightarrow A \end{array} \right\}$$

تمرين: در مجموعه اصلی  $F$  رابطه

$$F_2 = \left\{ \begin{array}{l} A \rightarrow (C, D, E) \\ B \rightarrow (C, E) \\ (A, D) \rightarrow E \\ (C, D) \rightarrow F \\ (B, D) \rightarrow A \\ (C, E, D) \rightarrow (A, B, D) \end{array} \right\}$$

تمرين: در رابطه  $R(A, B, C, D)$  میکشیم

?  $\rightarrow$  مجموعه اصلی  $(A, D)$  است:  $(A, D)^+ \rightarrow$  مجموعه  $A \rightarrow D$ :

توضیح: وابستگی تابعی  $C \rightarrow B \rightarrow C$  را می‌توان با صاب رابطه‌ای هم بیان کرد:

CONSTRAINT CTOBFD

فرض:  $R_1, R_2$ : دو تغییر تابعی تعریف

FORALL  $R_1$  (FORALL  $R_2$  (

$\exists$  در می‌راید

IF  $R_1 \cdot B = R_2 \cdot B$  THEN

$R_1 \cdot C = R_2 \cdot C$  ));

تمرين: الگوریتم پائس  $A^+$  و الگوریتم حذف FD از زیر نویسید.

(B-1)

مثال: در رابطه  $R(A, B, C, D)$

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

مجموعه امکانات ناپذیر FD هایی که می‌توانند این است:

نمای:

$$A \rightarrow C, A \rightarrow B : \text{نمای } A \rightarrow (B, C) \quad \text{ن} - 1$$

$$C \rightarrow D \Rightarrow A \rightarrow (A, C) : \text{نمای } (A, A) \rightarrow (A, C) : \text{نمای } A \rightarrow C \quad \text{ن} - 2$$

$$A \rightarrow D : \text{نمای}$$

$$(A, B) \rightarrow C : \text{نمای } A \rightarrow C \quad \text{ن} - 4$$

$(A, C) \rightarrow D$  می‌تواند وابستگی تابعی  $A \rightarrow C$  باشد

$A \rightarrow C : \text{نمای } B \rightarrow C, A \rightarrow B \quad \text{ن} - 5$

از زیر نویسید و حذف شود.

تمرين: در مجموعه امکانات ناپذیر روابطه:

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

نمای: در مجموعه امکانات ناپذیر روابطه:

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

تمرين: در رابطه  $R(A, B, C, D)$  می‌تواند وابستگی تابعی  $(A, D) \rightarrow B$  باشد؟

نمای: در مجموعه امکانات ناپذیر روابطه  $(A, D)^+$  می‌تواند وابستگی تابعی  $A \rightarrow D$  باشد؟

## قواعد استنتاج آرمسترانگ برای MVD

: در رابطه (..., Y, Z)

- 1)  $Y \subset X \Rightarrow X \rightarrow \rightarrow Y$
- 2)  $X \rightarrow \rightarrow Y, Y \rightarrow \rightarrow Z \Rightarrow X \rightarrow \rightarrow Z - Y$
- 3)  $X \rightarrow \rightarrow Y \Rightarrow X \rightarrow \rightarrow (R(H) - (X, Y))$
- 4)  $X \rightarrow Y \Rightarrow X \rightarrow \rightarrow Y$
- 5)  $X \rightarrow \rightarrow Y, X \rightarrow \rightarrow Z \Rightarrow X \rightarrow \rightarrow (Y, Z)$
- 6)  $X \rightarrow \rightarrow Y, (Y, Z) \rightarrow \rightarrow W \Rightarrow (X, Z) \rightarrow \rightarrow (W - (Y, Z))$
- 7)  $X \rightarrow \rightarrow Y, Z \subseteq W \Rightarrow (X, W) \rightarrow \rightarrow (Y, Z)$
- 8)  $X \rightarrow \rightarrow (Y, Z) \Rightarrow X \rightarrow \rightarrow Y \cap Z, X \rightarrow \rightarrow Y - Z, X \rightarrow \rightarrow Z - Y$
- 9)  $X \rightarrow \rightarrow Y, Z \rightarrow W, W \subseteq Y, Y \cap Z = \emptyset \Rightarrow X \rightarrow \rightarrow W$

## تعریف فرمال MVD

در رابطه  $R(X, Y, Z)$ ، وابستگی تابعی چندمقداری  $Y \rightarrow X$  وجود دارد اگر: در هر مقدار از رابطه  $R$  (در گستردگی  $R$  در تمام لحظات)، به ازاء تمام جفت تاپلهای  $t_1$  و  $t_2$ ، چنانچه  $t_1[X] = t_2[X]$  باشد، آنگاه تاپلهای  $t_3$  و  $t_4$  نیز وجود داشته باشند به نحوی که:

$$\left\{ \begin{array}{l} t_1[X] = t_2[X] = t_3[X] = t_4[X] \\ t_3[Y] = t_1[Y] \\ t_3[R(H) - Y] = t_2[R(H) - Y] \\ t_4[Y] = t_2[Y] \\ t_4[R(H) - Y] = t_1[R(H) - Y] \end{array} \right.$$

## دو تعریف دیگر رابطه 4NF

- رابطه  $R$  با مجموعه وابستگیهای تابعی و چندمقداری  $F$ ، در 4NF است اگر برای هر وابستگی تابعی در  $F^+$  به صورت  $Y \subseteq R(H), X \subseteq R(H)$   $X \rightarrow \rightarrow Y$  (یعنی  $Y \subseteq R(H)$ ) حداقل یکی از

دو وضع زیر برقرار باشد [RICA 90]:

- $Y \rightarrow X$  یک وابستگی تابعی چندمقداری نامهم باشد.

- $X$  سوپرکلید  $R$  باشد.

- رابطه  $R$  در 4NF است اگر در صورت وجود وابستگی چندمقداری  $Y \rightarrow X$  در  $F^+$ .

[SIMO 95]  $X$  کلید کاندید  $R$  باشد  $(Y \subseteq R(H), X \subseteq R(H))$

B-1

## ۹- دلایل بروز افزونگی

در سیستم کسی ISR در معنای عام

(از جمله در سیستم کار پایگاهی)

۱- ماهیت داده کسر سازمان (افزونگی طبیعی)

۲- مکنیک رستفاده شده در تأمین و ستد اخیر در سیستم کار آر (مثلًا نایابی سازی ...)

۳- هستی طراحی و تولید سیستم کار بر دی (مثلًا در هستی فایلینگ: نایابیگاهی)

۴- ماهیت ساختار داده از (مثلًا کلید خارجی در ساختار داده رابطه ای ...)

۵- طراحی بد

۶- رضانمودن صفت (صفات) به رابطه بر ار افزایش سرعت اجرای برنامه

۷- تولید نسخه (کپی) دیگر لز داده (به دلیل عدمیابی پایه در لذتی ...)

۸- عایبگاهی کردن صفات

۹- کاهش درجه تمایلی رابطه

۱۰- تولید نسخه کمی لز داده بر ار کفیض بسیت که (در سیستم نویزیع شده)

۱۱- مکنیک طراحی (مثلًا نیزه افزونگی برای کاهش هیچ تهار، ...)

۱۲- سکر از دخیره سازنر نیزی که لز داده (مثلًا نیزی لز کپی یا سبی لز کپ رابطه)

بر ار سرعت اجرای اگر کم و را بازیابی ... ( مثل دید دخیره شده )

۱۳- نیزه سازی رابطه (جزئیتی محدودی رابطه به دو یا بیش لز دخیره در صورت انجام تجزیه خوب ...

توجه: معایب افزونگی: کامپوت حافظه بیشتر

ک افزونگاری (بر ار ایجاد بینگام سلای متر گزند)

منتهی افزونگی: ک افزایش کار ای سیستم در بازیابی

ک ایعات لازم برای ترمیم، رکنی و ...

مأخذ: معاهجه بنادری یا تکاه داده

R-1 کجاوی: دلایل دیگر؟

Query Exercises

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

```

S { S#, SNAME, STATUS, CITY }
  PRIMARY KEY { S# }

P { P#, PNAME, COLOR, WEIGHT, CITY }
  PRIMARY KEY { P# }

J { J#, JNAME, CITY }
  PRIMARY KEY { J# }

SPJ { S#, P#, J#, QTY }
  PRIMARY KEY { S#, P#, J# }
  FOREIGN KEY { S# } REFERENCES S
  FOREIGN KEY { P# } REFERENCES P
  FOREIGN KEY { J# } REFERENCES J

```

- 6.13 Get full details of all projects.
- 6.14 Get full details of all projects in London.
- 6.15 Get supplier numbers for suppliers who supply project J1.
- 6.16 Get all shipments where the quantity is in the range 300 to 750 inclusive.
- 6.17 Get all part-color/part-city combinations. *Note:* Here and subsequently, the term "all" is to be taken to mean "all currently represented in the database," not "all possible."
- 6.18 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are all colocated (i.e., all in the same city).
  
- 6.19 Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are not all colocated.
- 6.20 Get all supplier-number/part-number/project-number triples such that no two of the indicated supplier, part, and project are colocated.
- 6.21 Get full details for parts supplied by a supplier in London.
- 6.22 Get part numbers for parts supplied by a supplier in London to a project in London.
- 6.23 Get all pairs of city names such that a supplier in the first city supplies a project in the second city.
- 6.24 Get part numbers for parts supplied to any project by a supplier in the same city as that project.
- 6.25 Get project numbers for projects supplied by at least one supplier not in the same city.
- 6.26 Get all pairs of part numbers such that some supplier supplies both the indicated parts.
- 6.27 Get the total number of projects supplied by supplier S1.
- 6.28 Get the total quantity of part P1 supplied by supplier S1.
- 6.29 For each part being supplied to a project, get the part number, the project number, and the corresponding total quantity.
- 6.30 Get part numbers of parts supplied to some project in an average quantity of more than 350.
- 6.31 Get project names for projects supplied by supplier S1.
- 6.32 Get colors of parts supplied by supplier S1.
- 6.33 Get part numbers for parts supplied to any project in London.
- 6.34 Get project numbers for projects using at least one part available from supplier S1.

- 6.35 Get supplier numbers for suppliers supplying at least one part supplied by at least one supplier who supplies at least one red part.
- 6.36 Get supplier numbers for suppliers with a status lower than that of supplier S1.
- 6.37 Get project numbers for projects whose city is first in the alphabetic list of such cities.
- 6.38 Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.
- 6.39 Get supplier numbers for suppliers supplying some project with part P1 in a quantity greater than the average shipment quantity of part P1 for that project.
- 6.40 Get project numbers for projects not supplied with any red part by any London supplier.
- 6.41 Get project numbers for projects supplied entirely by supplier S1.
- 6.42 Get part numbers for parts supplied to all projects in London.
- 6.43 Get supplier numbers for suppliers who supply the same part to all projects.
- 6.44 Get project numbers for projects supplied with at least all parts available from supplier S1.
- 6.45 Get all cities in which at least one supplier, part, or project is located.
- 6.46 Get part numbers for parts that are supplied either by a London supplier or to a London project.
- 6.47 Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.
- 6.48 Get all pairs of supplier numbers, Sx and Sy say, such that Sx and Sy supply exactly the same set of parts each. (Thanks to a correspondent, Fatma Mili of Oakland University, Rochester, Michigan,

for this problem. For simplicity, you might want to use the original suppliers and parts database for this exercise, instead of the expanded suppliers-parts-projects database.)

- 6.49 Get a “grouped” version of all shipments showing, for each supplier-number/part-number pair, the corresponding project numbers and quantities in the form of a binary relation.
- 6.50 Get an “ungrouped” version of the relation produced in Exercise 6.49.

: SQL  $\rightarrow$   $\Sigma^*$

7.15 We have numbered the following solutions as 7.15.n, where 6.n is the number of the original exercise in Chapter 6.

7.15.13 SELECT \*  
FROM J ;

Or simply:

TABLE J ;

7.15.14 SELECT J.\*  
FROM J  
WHERE J.CITY = 'London' ;

7.15.15 SELECT DISTINCT SPJ.S#  
FROM SPJ  
WHERE SPJ.J# = 'J1' ;

7.15.16 SELECT SPJ.\*  
FROM SPJ  
WHERE SPJ.QTY >= 300  
AND SPJ.QTY <= 750 ;

7.15.17 SELECT DISTINCT P.COLOR, P.CITY  
FROM P ;

7.15.18 SELECT S.S#, P.P#, J.J#  
FROM S, P, J  
WHERE S.CITY = P.CITY  
AND P.CITY = J.CITY ;

7.15.19 SELECT S.S#, P.P#, J.J#  
FROM S, P, J  
WHERE NOT ( S.CITY = P.CITY AND  
P.CITY = J.CITY ) ;

7.15.20 SELECT S.S#, P.P#, J.J#  
FROM S, P, J  
WHERE S.CITY <> P.CITY  
AND P.CITY <> J.CITY  
AND J.CITY <> P.CITY ;

7.15.21 SELECT DISTINCT SPJ.P#  
FROM SPJ  
~~X~~ WHERE ( SELECT S.CITY  
FROM S  
WHERE S.S# = SPJ.S# ) = 'London' ;

Select P.\* from S, P, SPJ

where SPJ.S# = S.S#

7.15.22 SELECT DISTINCT SPJ.P#  
FROM SPJ  
WHERE ( SELECT S.CITY  
FROM S  
WHERE S.S# = SPJ.S# ) = 'London'  
AND ( SELECT J.CITY  
FROM J  
WHERE J.J# = SPJ.J# ) = 'London' ;

And SPJ.P# = P.P#

And J.CITY = 'London'

7.15.23 SELECT DISTINCT S.CITY AS SCITY, J.CITY AS JCITY  
FROM S, J  
WHERE EXISTS  
( SELECT \*  
FROM SPJ  
WHERE SPJ.S# = S.S#  
AND SPJ.J# = J.J# ) ;

7.15.24 SELECT DISTINCT SPJ.P#  
FROM SPJ  
WHERE ( SELECT S.CITY  
FROM S  
WHERE S.S# = SPJ.S# ) =  
( SELECT J.CITY  
FROM J  
WHERE J.J# = SPJ.J# ) ;

7.15.25 SELECT DISTINCT SPJ.J#  
FROM SPJ  
WHERE ( SELECT S.CITY  
FROM S  
WHERE S.S# = SPJ.S# ) <>  
( SELECT J.CITY  
FROM J  
WHERE J.J# = SPJ.J# ) ;

7.15.26 SELECT DISTINCT SPJX.P# AS PA, SPJY.P# AS PB  
FROM SPJ AS SPJX, SPJ AS SPJY  
WHERE SPJX.S# = SPJY.S#  
AND SPJX.P# < SPJY.P# ;

7.15.27 SELECT COUNT ( DISTINCT SPJ.J# ) AS N  
FROM SPJ  
WHERE SPJ.S# = 'S1' ;

7.15.28 SELECT SUM ( SPJ.QTY ) AS X  
FROM SPJ  
WHERE SPJ.S# = 'S1'  
AND SPJ.P# = 'P1' ;

7.15.29 SELECT SPJ.P#, SPJ.J#, SUM ( SPJ.QTY ) AS Y  
FROM SPJ  
GROUP BY SPJ.P#, SPJ.J# ;

7.15.30 SELECT DISTINCT SPJ.P#  
FROM SPJ  
GROUP BY SPJ.P#, SPJ.J#  
HAVING AVG ( SPJ.QTY ) > 350 ;

7.15.31 SELECT DISTINCT J.JNAME  
FROM J, SPJ  
WHERE J.J# = SPJ.J#  
AND SPJ.S# = 'S1' ;

7.15.32 SELECT DISTINCT P.COLOR  
FROM P, SPJ  
WHERE P.P# = SPJ.P#  
AND SPJ.S# = 'S1' ;

7.15.33 SELECT DISTINCT SPJ.P#  
FROM SPJ, J  
WHERE SPJ.J# = J.J#  
AND J.CITY = 'London' ;

7.15.34 SELECT DISTINCT SPJX.J#  
FROM SPJ AS SPJX, SPJ AS SPJY  
WHERE SPJX.P# = SPJY.P#  
AND SPJY.S# = 'S1' ;

7.15.35 SELECT DISTINCT SPJX.S#  
FROM SPJ AS SPJX, SPJ AS SPJY, SPJ AS SPJZ  
WHERE SPJX.P# = SPJY.P#  
AND SPJY.S# = SPJZ.S#  
AND ( SELECT P.COLOR  
FROM P  
WHERE P.P# = SPJZ.P# ) = 'Red' ;

7.15.36 SELECT S.S#  
FROM S  
WHERE S.STATUS < ( SELECT S.STATUS  
FROM S  
WHERE S.S# = 'S1' ) ;

7.15.37 SELECT J.J#  
FROM J  
WHERE J.CITY = ( SELECT MIN ( J.CITY )  
FROM J ) ;

7.15.38 SELECT DISTINCT SPJX.J#  
FROM SPJ AS SPJX  
WHERE SPJX.P# = 'P1'  
AND ( SELECT AVG ( SPJY.QTY )  
FROM SPJ AS SPJY  
WHERE SPJY.J# = SPJX.J#  
AND SPJY.P# = 'P1' ) >  
( SELECT MAX ( SPJZ.QTY )  
FROM SPJ AS SPJZ  
WHERE SPJZ.J# = 'J1' ) ;

7.15.39 SELECT DISTINCT SPJX.S#  
FROM SPJ AS SPJX  
WHERE SPJX.P# = 'P1'  
AND SPJX.QTY > ( SELECT AVG ( SPJY.QTY )  
FROM SPJ AS SPJY  
WHERE SPJY.P# = 'P1'  
AND SPJY.J# = SPJX.J# ) ;

7.15.40 SELECT J.J#  
FROM J  
WHERE NOT EXISTS  
( SELECT \*  
FROM SPJ, P, S  
WHERE SPJ.J# = J.J#  
AND SPJ.P# = P.P#  
AND SPJ.S# = S.S#  
AND P.COLOR = 'Red'  
AND S.CITY = 'London' ) ;

7.15.41 SELECT J.J#  
FROM J  
WHERE NOT EXISTS  
( SELECT \*  
FROM SPJ  
WHERE SPJ.J# = J.J#  
AND NOT ( SPJ.S# = 'S1' ) ) ;

7.15.42 SELECT P.P#  
FROM P  
WHERE NOT EXISTS  
( SELECT \*  
FROM J  
WHERE J.CITY = 'London'  
AND NOT EXISTS  
( SELECT \*  
FROM SPJ  
WHERE SPJ.P# = P.P#  
AND SPJ.J# = J.J# ) ) ;

7.15.43 SELECT S.S#  
FROM S  
WHERE EXISTS  
( SELECT \*  
FROM P  
WHERE NOT EXISTS  
( SELECT \*  
FROM J  
WHERE NOT EXISTS  
( SELECT \*  
FROM SPJ  
WHERE SPJ.S# = S.S#  
AND SPJ.P# = P.P#  
AND SPJ.J# = J.J# ) ) ;

7.15.44 SELECT J.J#  
FROM J  
WHERE NOT EXISTS  
( SELECT \*  
FROM SPJ AS SPJX  
WHERE SPJX.S# = 'S1'  
AND NOT EXISTS  
( SELECT \*  
FROM SPJ AS SPJY  
WHERE SPJY.P# = SPJX.P#  
AND SPJY.J# = J.J# ) ) ;

7.15.45 SELECT S.CITY FROM S  
UNION  
SELECT P.CITY FROM P  
UNION  
SELECT J.CITY FROM J ;

7.15.46 SELECT DISTINCT SPJ.P#  
FROM SPJ  
WHERE ( SELECT S.CITY  
FROM S  
WHERE S.S# = SPJ.S# ) = 'London'  
OR ( SELECT J.CITY  
FROM J  
WHERE J.J# = SPJ.J# ) = 'London' ;

7.15.47 SELECT S.S#, P.P#  
FROM S, P  
EXCEPT  
SELECT SPJ.S#, SPJ.P#  
FROM SPJ ;

7.15.48 Solution omitted.

7.15.49-7.15.50 Cannot be done.

B

## View Updations in SQL-standard

نمای اساز دیده در SQL-standard

- المصطلح بنظام‌سازی درینجا معنی: عمليات درج، حذف و تغییر (بنظام‌سازی)
- در SQL-standard، هو صنوح دیده‌ی قابل‌عمليات بنظام‌سازی (دیده‌رندرا)
- \* همچنان روش نسيت. اما لگسته از جزئيات متداول گفت که دیده‌ی آنها
  - تمام سرالطيزير مراد استه باشند، فقط قابل‌عمليات بنظام‌سازه است (توضیح! همان است برخی دیگر لز دیده بهم قابل‌عمليات بنظام‌ساز باشند):
  - عبارت تعریف کته دید، آنکه عبارت SELECT ساده باشد (معنی عبارت EXCEPT، INTERSECT، UNION، JOIN نباشد).
  - در عبارت SELECT گردنده DISTINCT وجود نداشته باشد.
  - در كلارن SELECT، فقط یک صدول وجود را شه باشد.
  - صدول قيد شده در كلارن FROM، همچنان مبدل هبنا یا یک دیده قابل‌بنظام‌سازی باشد.
  - در لیست نام ستون در عبارت SELECT، ستون که مرور نظر باشد در صدول مبتدا هستا خواسته باشد و به همکنی ستون لز بود مبدل هبنا بپسی لز یک بار راجع وجود نداشته باشد.
  - در عبارت SELECT، كلارن HAVING و/یا كلارن GROUP BY وجود نداشته باشد.
  - كلارن WHERE در عبارت SELECT خواهد کله FROM نباشد
    - لگهای که در آن به صفات صدولی راجع را داشده باشد که در كلارن ذکر شده در سطر ۴.
    - نسبتی که علاوه بر دیده‌ی آنها نیز محبوبه رفته - عوایدی دلایلی طبعاً لز یک صدول مبتدا (یا لز یک دیده قابل‌بنظام‌سازی) باشد، فقط قابل‌بنظام‌سازی هستند.
    - (توضیح! به سطر رعایت نموده است که جایی، مثل کتابی که کلمه، همچو (زنگنه) و ...)

B

## عملیات بازگشتی در برنامه زیرگرد

۱- آیا می توانست با SQL عملیات بازگشتی را برنامه زیرگرد؟

پاسخ:بله . ساختار درس پیشناز ساختار درس  
مثال: جدول (COPRECO (COID, PRECOID) مارک نظری هستند . این جدول ارتباط  
پیشنازی بین نوع موجود است درس و خود من را نشان می دهد . سوال کادربر  
چنین است:

Q: ساختار درس تمام درسی پیشناز درس شماره 'COM222' را بنویسید .

: وظایف "برنامه" ذیر زدن نیاز پاسخ می دهد (در SQL / ستاره ای )

WITH RECURSIVE

PRECORS ( COID , PRECOID ) AS

( SELECT COID , PRECOID

FROM COPRECO

WHERE COID = 'COM222'

UNION ALL

SELECT COPRECO ( COID ) , COPRECO ( PRECOID )

FROM PRECORS , COPRECO

WHERE COPRECO . COID = PRECORS ( PRECOID )

SELECT \* FROM PRECORS

ORDER BY COID , PRECOID ;

جزئی اول: این قسم "برنامه" را بازگشتی برداشت.

جزئی دوم: ساختار وظایفی

PP ( MAJP# , MINP# , Q ) معرفی می شود : جزوی دوم

بازگشتی در SQL پاسخ می دهد :

Q: ساختار تمام وظایف را تکمیل کرده تقدیر می کنید :

\* مراحل می باشد: "نمایم بنا بر این پاسخ داده شد"

\* مراحل می باشد: "DATE 03 7:20"

در SQL استاندارد امکان انجام عملیات بازگشتی وجود ندارد. برای این منظور کلار

کلار می خواهد - با این کلار که دید (دیده داشت) تعریف شود. در تعریف این دید، لزخود ریس دید  
هم بخوب استفاده می شود. در واقع مگر فهم دید بازگشتی با مرتبه تعریف شود. این دید به صورتی

اجتماع دو پرسش تعریف می شود: یک پرسش صنایع بازگشتی است و دیگر پرسش بازگشتی  
که آن را دید پارگرگشتی استفاده می شود. سری نهاده بازیاب لزک جدول با

عملیات بازگشتی دید می شود:

WITH RECURSIVE Tempz ( col1 ) AS

( SELECT col1 )

FROM table

UNION

SELECT table . col , Tempz . col

FROM table , Tempz

WHERE ----- )

SELECT \* FROM Tempz ;

مانند پرسش میباشد و کلار بازگشتی

- table , Tempz , table ) : می خواهد جدول SELECT

پس از

1- Base query

2- Recursive query