Fundamentals Concepts of

Data Bases

اصول طراحي پايگاه داده ها

جلسه ۵: SQL

مدرس : اسماعیل نورانی

http://www.nurani.ir/ - Info@nurani.Ir

احکام

بازیابی : ← حکم واحد SELECT

ذخیره سازی : احکام با Syntax (INSERT ,DELETE , UPDATE)

امکانات جبر رابطه ای و محاسبات رابطه ای تقریبا بطور کامل در حکم SELECT استفاده می شود

شکل کلی SELECT

```
SELECT [DISTINCT] items
FROM table(s)
[WHERE condition(s)]
[ORDER BY]
[GROUP BY]
[HAVING ]
```

برای خروجیهای Sorted است.

Option ایجاد نظم در خروجیها می باشد

مثال : مشخصات تهیه کنندگان ساکن شهر C2 را بدهید ؟

```
SELECT  S# , Sname , Status , City
FROM  S
WHERE  City ='c2' ;
```

S#	SNAME	STATUS	CITY
S1	Sn1	20	C2
S4	Sn4	20	C2

وقتی تمام ستونهای جدول را بخواهید نیازی به ذکر نام صفات خاصه نیست ، بلکه یک * بجای آنها کافی است .

SELECT * تمامی مشخصات را می آورد و مخصوصا در مواردی که تعداد صفات خاصه زیاد است مفید است .

SELECT SQL وقتی سطر کامل استخراج می کنیم شبیه سازی SELECT جبری است که تاپلهایی از جدولها به ما می دهد SELECT * را اگر شرط بدهیم و یا تدهیم بازهم SELECT جبری خواهد بود

مثال :

```
SELECT S#, City
FROM S ;
```

دو ستون از جدول را می دهد .

S#	CITY
S1	C2
S2	C3
S3	C3
S4	C2
S5	C1

مثال :

عینا ستونهای P# از جدول SP را می دهد. آیا این یک Project است ؟

خیر ← زیرا تمام اپراتورهای جبر رابطه ای حاصشان یک رابطه است و رابطه عنصر تکراری ندارد

در مثال قبل چرا رابطه بود و چرا Project بود ؟

زیرا S# کلید است و عنصر تکراری ایجاد نمی شود پس * معادل عملکرد Project نیست و باید از گزینه DISTINCT استفاده کرد.

* Select SQL می تواند عملکرد Select جبری را داشته باشد و عملکرد Project را داشته باشد.

* در حالت کلی عملکرد Select SQL معادل ترکیب عملکرد Selectجبری و Project جبری است .

```
SELECT P#
FROM SP ;
```

P#
P1
P2
P3
P4
P1
P2
P1
P2
…

```
SELECT DISTINCT P#
FROM SP;
```

P#
P1
P2
P3
P4

Qualifier ستون و یا Qualifier صفت خاصه ، قید ستون

مثال :

```
SELECT S.S# , S.Status
FROM S
WHERE City='C2'  or City='C3' ;
```

S#	Status	
S1	20	ساکن شهر C2
S3	30	ساکن شهر C3
S4	20	ساکن شهرC2

در این مثال نیازی به تصریح قید نیست ، مواردی وجود دارد که در آنها استفاده از Qualifier

توصیه می شود و گاه الزامی است .(توصیه در مواردی است که بخواهید به Query وضوح ببخشید. الزام وقتی است که ستونهای همنام در جداول مختلف داشته باشیم .مثل : City در S,SP)

تمرین : همین Query را با جبر رابطه ای بنویسید.

ادامه SELECT

- بازیابی با جدول جواب منظم

Q : شماره تهیه کنندگان و وضعیت آنها را بدهید. جدول جواب به نظم صعودی مقادیرstatus مرتب شده باشد.

```
SELECT S# ,Status
    FROM  S
    ORDER   BY  Status

    ORDER   BY  Status DESC

    ORDER    BY 2                    شماره ستون درجدول جواب

    ORDER    BY  2 DESC
```

ASCENDING صعودی default است و احتیاج به ذکر نیست.

•بازیابی مقدار محاسبه شده ❊

Q: شماره هر قطعه و وزن آنرا به گرم بدهید. فرض کنید DBA وزن را به واحد کیلو گرم ذخیره نموده .

```
SELECT P# , WEIGHT*1000  AS  'WEIGHT IN GRAMS'
FROM P
```

عبارت محاسبه شدنی

جدول جواب صورتی چنین دارد:

P#	WEIGHT IN GRAMS
P1	12000
P2	17000
P3	17000
P4	14000
p5	12000
p6	19000

روش دوم :

```
SELECT P# , 'WEIGHT IN GRAMS' , WEIGHT*1000
FROM P
```

عبارت محاسبه شدنی

جدول جواب صورتی چنین دارد:

P#		
P1	WEIGHT IN GRAMS	12000
P2	WEIGHT IN GRAMS	17000
P3	WEIGHT IN GRAMS	17000
P4	WEIGHT IN GRAMS	14000
p5	WEIGHT IN GRAMS	12000
p6	WEIGHT IN GRAMS	19000

پیاده سازی اپراتور join: بازیابی از چند جدول

Join Syntax صراحتا در DB2/SQL وجود ندارد. اما در SQL92 پیش بینی شده است.

Query: مشخصات قطعات و تهیه کنندگان از یک شهررا بدهید. (بحث کلید خارجی بحثی داشتیم که آیا ارتباط تنها توسط کلید خارجی است؟

گفتیم که نه هر عضو مشترک می تواند بیانگر ارتباط باشد. (مثال : CITY در S , P)

```
SELECT  S.*   ,  P.*
   FROM     S , P
   WHERE  S.CITY = P.CITY
```

سیستم ابتدا ضرب کارترین دو رابطه را ایجاد می کند. سپس tuple های حائز شرایط را استخراج می کند. می توان در عمل join شرایطی را نیز مطرح کرد

عمل join (Natural Join)

(Equi Join)

S#	SNAME	STATUS	CITY	P#	PNAME	COLOR	WEIGHT	CITY
S1	SN1	20	C2	P1	Nut	Red	12	C2
S1	SN1	20	C2	P4	Screw	Red	14	C2
S1	SN1	20	C2	P6	Cog	Red	19	C2
S2	SN2	10	C3	P2	Bolt	Green	17	C3
S2	SN2	10	C3	P5	Cam	Blue	12	C3
S3	SN3	30	C3	P2	Bolt	Green	17	C3
S3	SN3	30	C3	P5	Cam	Blue	12	C3
S4	SN4	20	C2	P1	Nut	Red	12	C2
S4	SN4	20	C2	P4	Screw	Red	14	C2
S4	SN4	20	C2	P6	Cog	Red	19	C2

Q: مشخصات قطعات و تهیه کنندگان از یک شهر که قطعه آبی باشد

```
AND
COLOR = ‘ BLUE’
```

•لازم نیست حتما تمام مشخصات را بخواهید می توانید فقط برخی صفات خاصه را بخواهید. مثال:

Q: شماره قطعات و شماره تهیه کنندگان همشهر را بدهید.

```
SELECT   S.S# , P.P#
  FROM    S , P
  WHERE  S.CITY = P.CITY
     AND
       COLOR = ‘ BLUE ‘ ;
```

مثال : Query : شماره جفت تهیه کنندگان همشهر را بدهید

S	S#	Sname	Status	City
	S1	.	.	C۲
	S2	.	.	C۳
	S3	.	.	C۳
	S4	.	.	C۲
	S5	.	.	C1

جدول جواب

S#	S#
S1	S4
S2	S3

برای پاسخگویی به این قبیل Query ها باید S را با خود در SQL ، Join کرد

```
SELECT FIRST.S# , SECOND.S#
FROM  S  FIRST , S  SECOND
WHERE  FIRST.CITY = SECOND.CITY
```

از یک تکنیک دگر نامی استفاده شده است S را به نام First و به نام Second نامیده ایم .

بعد از FROM هرگاه نام یک رابطه بیایید و بعد از حداقل یک Blank ، یک نام بیاید ، سیستم آن نام را ، نام دیگر برای آن رابطه در نظر می گیرد.

حاصل SELECT :

S#	S#
S1	S1
S1	S4
S2	S2
S2	S3
S3	S2
S3	S3
S4	S1
S4	S4

تکراری

بدیهات

سیستم چه می کند ؟

۲ تا CURSOR می گیرد و دومی را MOVE می دهد

برای حذف بدیهات و تکراریها شرط زیر را اضافه کنید :

AND FIRST.S# < SECOND.S#

در نتیجه خواهیم داشت

```
SELECT FIRST.S# , SECOND.S#
FROM S  FIRST , S  SECOND
WHERE FIRST.CITY = SECOND.CITY
 AND FIRST.S# < SECOND.S#
```

تمرین : شماره جفت تهیه کنندگانی را بدهید که از یک شهر نباشد

```
SELECT FIRST.S# , SECOND.S#
FROM S  FIRST , S  SECOND
WHERE FIRST.CITY != SECOND.CITY
AND FIRST.S# < SECOND.S#
```

Aggregate Functions توابع جمعی (گروهی)

در حاشیه : قدرت Select :اپراتورهای Project و Join و Select جبر رابطه ای را تواما ارایه می دهد

برای پاسخگویی به سوالاتی از قبیل چه تعداد قطعه داریم ؟ ماکزیمم Quantity چقدر است ؟ خود Select قادر به پلسخگویی نیست. توابعی پیش بینی شده است که در متن Select بکار می روند .(مستقلا نمی توانید آنها را به کار ببرید)

این توابع عبارتند از :

COUNT (DISTINCT …)

COUNT(*)

MAX

MIN

SUM

AVG

حالتی از COUNT موسوم به COUNT(*) تمام سطرها را – حتی تکراری - می شمارد

مثال : تعداد محمولات را بدهید .(هر سطر SP یک محموله است) (SHIPMENT)

```
SELECT COUNT(*)
FROM SP ;
```

مثال : چند نوع قطعه تهیه شده است .

```
SELECT COUNT (DISTINCT P#)
FROM SP  ;
```

مثال : کلا چند نوع قطعه وجود دارد

```
SELECT COUNT(*)
FROM P ;
```

مثال :

```
SELECT MAX(STATUS)
FROM S ;
```

استفاده از GROUP BY :

Query : شماره هر قطعه تهیه شده و کل تعداد تهیه شده از هر قطعه را بدهید .

```
SELECT P# ,SUM(QTY)
FROM SP
GROUP BY P# ;
```

GROUP BY CLAUSE جدول داده شده بعد از FROM را منطقا گروه بندی می کند به نحوی که در هر گروه مقدار ستون یا ستونهای داده شده پس از GROUP BY یکسان است .(گروه بندی بر اساس آنچه بعد از GROUP BY داده می شود) آنگاه تابع SUM عمل می شود.

(ستونهایی که حاصل اعمال توابع هستند بی نام می باشند)

S#	P#	QTY
S1	P1	300
S2	P1	300
S1	P2	200
S2	P2	400
S3	P2	200
S4	P2	200
S1	P3	400
S1	P4	200
S1	P5	100

P#	
P1	600
P2	1000
P3	400
P4	200
P5	100

مثالی از به عینیت درآوردن فیلد مجازی حاصل از پردازش می باشد. (Indirect Materialization)

مثالی از کاربرد HAVING: (هدف این است که مفاهیم برای شما نا آشنا نباشد)

Query: شماره قطعاتی را بدهید که توسط بیش از یک تهیه کننده تهیه شده باشد.

نکته ۱ : HAVING همیشه با GROUP BY می آید و مستقل معنی ندارد.
نکته ۲: استفاده از GROUP BY ناروشمندی SQL را کمی تضعیف می کند. (زیرا می گویید گروهبندی کن) و آنرا تا حدی روشمند میکند . از نظر کوتاهی بسیار جالب است هر چند روشمند است.

```
SELECT      P#
FROM    SP
GROUP   BY   P#
HAVING  COUNT ( * ) > 1 ;
```

نقش having : نقش having در گروه همان است که نقش where در tuple.

Where برای بیان شرط یا شرایط ناظر به سطر است. having برای بیان شرط یا شرایط ناظر به گروه .

مثال : شماره هر تهیه کننده و تعداد کل قطعاتی را که تهیه کرده بدهید

```
select s#, sum(QTY)
  from sp
  group by s#
```

مثال : شماره تهیه کنندگانی را بدهید که دو نوع قطعه تهیه کرده اند

```
select s#
   from sp
   group by s#
   having count(*)=2
```

شماره هر تهیه کننده و تعداد انواع قطعاتی که تهیه کرده را در مورد تهیه کنندگانی که بیش از یک قطعه تهیه کرده اند ، بدهید

```
select s#, count(*)
   from sp
   group by s#
   having count(*)>1
```

در مورد قطعاتی که توسط سه تهیه کننده تهیه شده اند شماره قطعه و تعداد کل تهیه شده از قطعه را بدهید

```
select p#, sum(qty)
   from sp
   group by p#
   having count(*)=3
```

پرس و جو های فرعی SUBQUERY :

مثال : Q : اسامی تهیه کنندگاه قطعه P2 را بدهید .

روش اول : با استفاده از مکانیزم Join :

```
SELECT  Sname
FROM S , SP
WHERE  SP.S# = S.S#  AND
          SP.P# = 'P2' ;
```

بویژه در SQL 92 اپراتور Join صراحتا تامین شده است .

نکته ای که به این راه حل وارد است این است که فرمالیسم طبیعی در نوشتن Sname ...
FROM SP از بین رفته است . چون Sname در SP نیست . یعنی طبیعی این است که بنویسیم :

```
SELECT    Sname
FROM      S
  WHERE   شرایط
```

از این رو امکانی به نام SUBQUERY بوجود آمده است.

SUBQUERY همان ساختار SELECT – FROM – WHERE را دارد با این تفاوت که در درون یک SELECT دیگر نوشته می شود.

```
SELECT     Sname          ──────────►  OUTTER QUERY
FROM    S
WHERE  S# IN                { S1 , S2 , S3 , S4 }
          یا  S# = ANY
   ( SELECT     S#
     FROM  SP                          } INNER QUERY پرس و جوی فرعی
     WHERE  P# = ‘ P2 ‘ )
```

عملکرد = ANY با عملکرد IN یکسان است.

- هر کجا IN بتواند بیاید =ANY نیز می تواند .

نکته : عمل تو در تویی می تواند بیش از دو سطح باشد (سطوح می توانند از لحاظ تئوری بیش از دو سطح باشد اما در عمل ۹۸٪ Q ها با یک SELECT و یا حداکثر در دو سطح SELECT پاسخ داده می شوند)

مثال : اسامی تهیه کنندگانی که حداقل یک قطعه آبی رنگ تهیه می کنند.

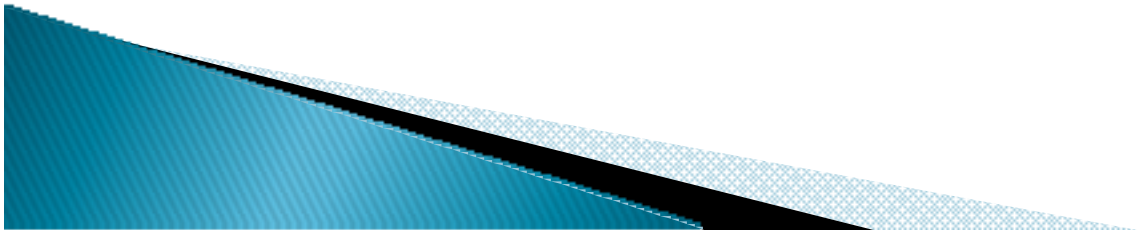
```
SELECT  Sname
FROM     S
WHERE  S# IN ( SELECT  S#
                    FROM   SP
                    WHERE  P#  IN   (  SELECT  P#
                                       FROM     P
                                       WHERE   P.COLOR = 'BLUE '  ) ) ;
```

سیستم از درونی ترین SELECT شروع میکند.

تمرین : با JOIN همین را بنویسید

با استفاده از JOIN

```
SELECT   S.SNAME
FROM S , SP , P
WHERE S.S# = SP.S# AND
          SP.P# = P.P# AND
          P.COLOR = 'BLUE'
```



نکته : می توان بجای اپراتور IN یا ANY= در شرایط خاصی از همان اپراتورهای متعارف مقایسه ای استفاده کرد.

```
SELECT  COL1
 FROM  table
 WHERE  COL2  = ( SELECT …..
                  >
                  <
```

مثال : شماره تهیه کنندگان هم شهر با 'S1' را بنویسید .

تمرین: همین Q را با JOIN بنویسید .

```
SELECT    S#
FROM      S
WHERE  CITY  = ( SELECT  CITY
                 FROM       S
                 WHERE   S#  =  ' S1' )
```

* شرط: وقتی که پاسخ SELECT درونی یک مقدار باشد یا بعبارت دیگر مجموعه جواب SELECT درونی تک عضوی باشد . بنابراین

```
SELECT    S# FROM    S
                 WHERE  CITY  =  ' C۲ '
```

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

```
SELECT      S#
        FROM   S
        WHERE  STATUS  < ( SELECT  MAX  ( STATUS )
                                FROM   S ) ;
```

نکات :

۲- استفاده از توابع جمعی در SUBQUERY ها مجاز است.

امکانات دیگر :

اپراتور Like

بازیابی از جدول دارای هیچ مقدار

اپراتور UNION

Q : شماره قطعاتی را بدهید که اسم آنها با کاراکتر C شروع شده باشد .

یعنی آغاز شونده با C

```
SELECT P#
FROM P WHERE  PNAME LIKE 'C%';
```

'%C' LIKE یعنی مختوم به حرف C

یعنی این String وجود داشته باشد مهم نیست کجا

```
WHERE PNAME LIKE '%CAB%'
```

'CAB--' یعنی ۵ حرف باشد و ۳ حرف اولش CAB باشد .

% نماینده n کاراکتر است _ نماینده فقط یک کااکتر است

```
WHERE PNAME LIKE '_C%'
```

نام قطعاتی که حرف دومشان C بوده و تعداد حروفشان نامعلوم است

Q : فرض : وضعیت S5 ، NULL می باشد .

S#	Sname	Status	CITY
S1		10	
S2		15	
S3		13	
S4		8	
S5		?	

نشان دهنده NULL

```
SELECT S#
FROM S
WHERE STATUS > 13 ;
```

جدول جواب

S#
S2

SQL ، NULL را به عنوان یک عملوند مقایسه در عمل مربوط دخالت نمی دهد . مانند این است که مقایسه صورت نگرفته است . اما اگر بنویسیم :

```
SELECT S#
FROM S
WHERE STATUS IS NULL
```

جدول جواب

S#
S5

پس می تواند NULL را بازشناسد و بنابراین تا حدی به NULL توجه شده است .

اپراتور UNION با همین Syntax وجود دارد و همچنین UNION ALL

مثال : Q : شماره قطعاتی را بدهید که یا توسط S3 تهیه شده باشد و یا وزن آنها کمتر از ۱۵ گرم باشد .

```
SELECT P#
FROM SP
WHERE S# = 'S3'
UNION
SELECT P#
FROM P
WHERE WEIGHT <15
```

UNION فقط روی رابطه هایی می تواند عمل کند که Type-Compatible باشند .

Type-Compatible : دو رابطه را گویند هرگاه heading هایشان یکسان باشد .

مجموعه جواب فاقد تکراری است . زیرا UNION اپراتور جبر رابطه ای است . ممکن است User نخواهد تکراری ها حذف شود . در این صورت از UNION ALL استفاده می کنیم .

- SQL تمام اپراتورهای جبر رابطه ای را یا بصورت صریح دارد یا می تواند شبیه سازی کند .

R1 INTERSECT R2

R1 MINUS R2

R1 PRODUCT R2

SQL امکانات محاسبات رابطه ای را هم دارد . مشخصا اپراتور EXISTS و NOTEXISTS را.

بازیابی به کمک EXISTS : Q : اسامی تهیه کنندگان قطعه P2 را بدهید . (تاکنون به سه روش این Q را زدیم) از این جا نتیجه می گیریم که یک Query در SQL به گونه های مختلف (چندین گونه) تنظیم (formulate) می شود که منطقا نوعی افزونگی در روش می باشد . Date این را یک ایراد مطرح کرده است (اما از دید برنامه نویسان یک نکته مثبت است .)

```
SELECT SNAME FROM S
WHERE EXISTS (SELECT * FROM SP
    (1)   WHERE SP.S#=S.S# AND SP.P#='P2')
```

نکته : وقتی در یک Query درونی به صفت خاصه ای بیرون از آن پرس و جو ارجاع می دهیم ، Qualifier باید قید شود . اما اگر صفت خاصه مربوط به همان Q درونی باشد ، الزامی نیست .

نکته : Clause شماره (۱) ظاهرا شبیه به Clause ای است که در join می نویسیم . سیستم در EXISTS عمل پیوند انجام نمی دهد . شاید بتوان گفت که از این لحاظ EXISTS از join کاراتر است زیرا ضرب کارتزین انجام نمی دهد .

نحوه اجرا : به ازای هر سطر از رابطه S سیستم بررسی می کند آیا وجود دارد سطری در SP که S# آن همان سطر از S و P# آن برابر P2 ؟ پس وجود را چک می کند (EXISTANCIAL) (اگر وجود داشته باشد یعنی عبارت محاسباتی True باشد) SNAME آن S# جواب است .

مثال : اسامی تهیه کنندگانی را بدهید که قطعه P2 را تهیه نکرده اند .

پاسخ با استفاده ا ز NOT EXISTS :

```
SELECT SNAME FROM S
    WHERE NOT EXISTS (SELECT * FROM SP
                          WHERE SP.S#=S.S# AND SP.P#='P2')
```

پاسخ با استفاده از SUBQUERY:

```
SELECT SNAME
FROM S
WHERE SNUM NOT IN ( SELECT SNUM
                    FROM SP
                    WHERE PNUM = 'P2')
```

توجه کنید که query ذیل پاسخ صحیح نمی باشد .

```
SELECT SNAME
FROM S,SP
WHERE S.SNUM = SP.SNUM
        AND SP.PNUM != 'P2'
```

در واقع اسامی تهیه کنندگانی را می دهد که حد اقل یک قطعه بجز P2 تهیه کرده اند .
لذا ممکن است تهیه کنندگان P2 را نیز شامل باشد و پاسخ سوال مورد نظر نمی باشد.

توجه کنید که query ذیل پاسخ صحیح نمی باشد .

```
SELECT SNAME
FROM S
WHERE SNUM  IN ( SELECT SNUM
                 FROM SP
                 WHERE PNAME != 'P2')
```

نام تهیه کنندگانی را بدهید که همه قطعات را تهیه کرده اند

```
SELECT SNAME
FROM S
WHERE   NOT EXISTS
        (SELECT *
           FROM P
           WHERE NOT EXISTS
           (SELECT *
            FROM SP
            WHERE SP.P#=P.P# AND S.S#=SP.S#))
```

روش دوم با استفاده از SUBQUERY (نام تهیه کنندگانی که همه قطعات را تهیه کرده اند)

```
select sname
from s
where s# in  ( select s#
               from sp
               group by s#
               having count(*) = (select count(*)
                                  from p))
```

نام تهیه کنندگانی را بدهید که حداقل یک قطعه را تهیه کرده اند .

```
SELECT SNAME
FROM S
WHERE  EXISTS
          (SELECT *
            FROM SP
            WHERE S.S#=SP.S#)
```

نام تهیه کنندگانی را بدهید که حداقل یک قطعه را تهیه نکرده اند

```
SELECT SNAME
FROM S
WHERE  EXISTS
        (SELECT *
           FROM P
           WHERE NOT EXISTS
           (SELECT *
            FROM SP
            WHERE SP.P#=P.P# AND S.S#=SP.S#))
```

نام تهیه کنندگانی را بدهید که هیچ قطعه ای را تهیه نکرده اند .

```
SELECT SNAME
FROM S
WHERE  NOT EXISTS
        ( SELECT *
            FROM SP
            WHERE S.S#=SP.S#)
```

روش دوم : با استفاده از SUBQUERY :

```
SELECT SNAME
FROM S
WHERE S# NOT IN (SELECT S#
                 FROM SP)
```

اپراتور INSERT :

INSERT

INTO tablename [cal1 , cal2 , . . .]

VALUES <سطر کامل یا ناقص>

مثال :

INSERT INTO P

VALUES('P7','Pn7','RED',10,'C2')

INSERT INTO P(COLOR,WEIGHT, P#,CITY,PNAME)
VALUES('RED',10,'P7', 'C2', 'PN7')

INSERT INTO P

VALUES('P7' , , ,10 , 'C2')

سطر ناقص به شرطی قابل قبول است که برای Pname و Color در P مقدار NotNULL ذکر نکرده باشید .

وارد کردن سطر ناقص :

```
INSERT INTO P(PNUM,WEIGHT,CITY)
VALUES('P7', 10 , 'C1')
```

فرم دوم دستور INSERT :

```
INSERT
INTO table
SUBQUERY
```

مثال :

```
CREATE TABLE TEMP
(PNUM  CHAR(8) NOT NULL,  SUMQTY INTEGER) PRIMARY KEY PNUM;

INSERT
INTO TEMP
SELECT P#,SUM(QTY)
FROM SP
GROUP BY P#;
```

روش دوم :

این دستور خودش جدول temp را ایجاد می کند

```
SELECT P#,SUM(QTY)
INTO TEMP
FROM SP
GROUP BY P#;
```

روش سوم :

```
CREATE TABLE TEMP
AS SELECT P#,SUM(QTY)
    FROM SP
   GROUP BY P#;
```

دستور UPDATE :

```
UPDATE tablename
SET  calname = “expression” یا  “Literal”
WHERE  . . .
```

مثال :

به هنگام سازی تک سطر (Tuple)

```
UPDATE SP
SET QTY  = 0
WHERE  S# = S2  AND  P# = ‘P2’
```

به هنگام سازی چند تاپل

```
UPDATE S
SET  STATUS = STATUS *2
WHERE  CITY = ‘C3’
```

به هنگام سازی بیش از یک جدول : شماره تهیه کننده S1 را به S11 تغییر دهید .

```
UPDATE S
SET S#='S11'
WHERE S#='S1';
```

این UPDATE باید PROPAGATE شود توسط Cascade بخاطر رعایت قاعده C2 .

```
UPDATE SP
SET S#='S11'
WHERE S#='S1' ;
```

⇒ این عمل ، توسط Cascade بخاطر رعایت قاعده C2 انجام شده است

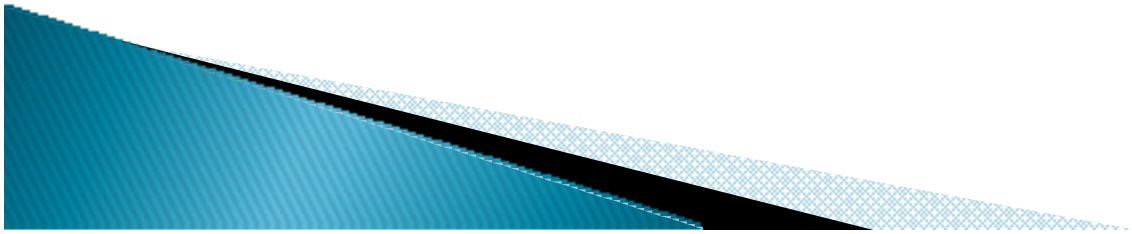
مثال : ۱۰ واحد به QTY تهیه کنندگان ساکن C3 اضافه کنید . (احتیاج به sub query دارید .)

```
UPDATE SP
  SET QTY= QTY + 10
  WHERE SNUM IN ( SELECT SNUM
                  FROM S
                  WHERE CITY = 'C3')
```

```
 UPDATE SP
   SET QTY= QTY + 10
   WHERE EXISTS ( SELECT *
                  FROM S
                  WHERE SP.S# = S.S#  AND
                        S.CITY = 'C3')
```

مثال : update چند ستون از رکورد بطور همزمان

```
UPDATE P
SET COLOR = ‘YELLOW’,
WEIGHT = WEIGHT +5,
CITY = NULL
WHERE P# = ‘P2’ ;
```



دستور DELETE

مثال : حذف تک سطر :<S3,P1,50> را حذف کنید .

```
DELETE FROM SP
WHERE S#='S3' AND P#='P1'
```

حذف چند سطر : محموله های با QTY کوچکتر از ۲۰ را حذف کنید .

```
DELETE FROM SP
WHERE QTY < 300
```

مثال : تهیه کننده S2 را حذف کنید .

```
DELETE FROM S
WHERE S#='S2';
```

بخاطر رعایت قاعده C2 باید این حکم نیز اجرا شود :
بطور اتوماتیک سیستم اعمال می کند .

```
DELETE FROM SP
WHERE S#='S2';
```

همه رکوردهای جدول را حذف
می کند .

```
DELETE FROM SP;
```

مثال : محمولات مربوط به تهیه کنندگان شهر c3 را حذف کنید

```
DELETE FROM SP
WHERE SNUM IN (SELECT SNUM
                FROM S
                WHERE CITY= 'C3')
```

```
DELETE FROM SP
WHERE 'C3' =( SELECT CITY
               FROM S
               WHERE SP.SNUM= S.SNUM )
```