SQL

Jef Wijsen

UMONS

May 14, 2018

S	S#	SNAME	STATUS	CITY
	S1	Smith	20	London
	S2	Jones	10	Paris
	S3	Blake	30	Paris
	S4	Clark	20	London
	S5	Adams	30	Athens

P	P#	PNAME	COLOR	WEIGHT	CITY
	P1	Nut	Red	12	London
	P2	Bolt	Green	17	Paris
	P3	Screw	Blue	17	Rome
	P4	Screw	Red	14	London
	P5	Cam	Blue	12	Paris
	P6	Cog	Red	19	London

SP	S#	P#	QTY
	S1	P1	300
	S1	P2	200
	S1	P3	400
	S1	P4	200
	S1	P5	100
	S1	P6	100
	S2	P1	300
	S2	P2	400
	S3	P2	200
	S4	P2	200
	S4	P4	300
	S4	P5	400

Create

```
CREATE DOMAIN COLOR CHAR(6) DEFAULT '???'
       CONSTRAINT VALID_COLORS
       CHECK ( VALUE IN
             ( 'Red', 'Yellow', 'Blue', 'Green', '???' ) ) ;

CREATE DOMAIN S#    CHAR(5);
...
CREATE DOMAIN QTY   NUMERIC(9);
```

```
CREATE TABLE S
      ( S#        S#,
        SNAME     NAME,
        STATUS    STATUS,
        CITY      CITY,
      PRIMARY KEY ( S# ) ) ;

CREATE TABLE P
      ( P#        P#,
        PNAME     NAME,
        COLOR     COLOR,
        WEIGHT    WEIGHT,
        CITY      CITY,
      PRIMARY KEY ( P# ) ) ;
```

```
CREATE TABLE SP
      ( S# S# NOT NULL, P# P# NOT NULL, QTY QTY NOT NULL,
        PRIMARY KEY ( S#, P# ),
        FOREIGN KEY ( S# ) REFERENCES S
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
        FOREIGN KEY ( P# ) REFERENCES P
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
        CHECK ( QTY > 0 AND QTY < 5001 ) ) ;
```

Alter and Drop

```
ALTER TABLE S ADD COLUMN DISCOUNT INTEGER ;

DROP TABLE S ;
```

Select

Get number and status for suppliers in Paris.

```
SELECT S.S#, S.STATUS
FROM   S
WHERE  S.CITY = 'Paris' ;
```

Unqualified names are acceptable if they cause no ambiguity.

```
SELECT S#, STATUS
FROM   S
WHERE  CITY = 'Paris' ;
```

Result is always a new table.

```
  SELECT [ DISTINCT ] ...
  FROM    ...
[ WHERE   ... ]
[ GROUP  BY ... ]
[ HAVING ... ]
[ ORDER BY ... ]
```

```
SELECT S.S#, S.STATUS
FROM   S
WHERE  S.CITY = 'Paris'
ORDER  BY STATUS DESC ;
```

Get the part number for each part supplied.

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

For all parts, get the part number and the weight of that part in grams (1 pound = 454 gram).

```
SELECT P.P#, 'Weight in grams', P.WEIGHT * 454 AS GMWT
FROM   P ;
```

P#		GMWT
P1	Weight in grams	5448
P2	Weight in grams	7718
P3	Weight in grams	7718
P4	Weight in grams	6356
P5	Weight in grams	5448
P6	Weight in grams	8626

Get full details of all suppliers.

```
SELECT *
FROM   S ;
```

or

```
SELECT S.*
FROM   S ;
```

Get number and status for suppliers in Paris with status greater than 20.

```
SELECT S.S#, S.STATUS
FROM   S
WHERE  S.CITY = 'Paris'
AND    S.STATUS > 20 ;
```

Get all combinations of supplier number and part number such that the supplier and part in question are colocated.

```
SELECT S.S#, P.P#
FROM   S, P
WHERE  S.CITY = P.CITY ;
```

"`FROM S, P`" gives Cartesian product of S and P.

S.S#	S.SNAME	S.STATUS	S.CITY	P.P#	P.PNAME	...	P.CITY
S1	Smith	20	London	P1	Nut	...	London
S1	Smith	20	London	P2	Bolt	...	Paris
S1	Smith	20	London	P3	Screw	...	Rome
S1	Smith	20	London	P4	Screw	...	London
S1	Smith	20	London	P5	Cam	...	Paris
S1	Smith	20	London	P6	Cog	...	London
S2	Jones	10	Paris	P1	Nut	...	London
S2	Jones	10	Paris	P2	Bolt	...	Paris
S2	Jones	10	Paris	P3	Screw	...	Rome
S2	Jones	10	Paris	P4	Screw	...	London
S2	Jones	10	Paris	P5	Cam	...	Paris
S2	Jones	10	Paris	P6	Cog	...	London
			...				
S5	Adams	30	Athens	P6	Cog	...	London

"`WHERE S.CITY = P.CITY`" selects tuples satisfying condition.

S.S#	S.SNAME	S.STATUS	S.CITY	P.P#	P.PNAME	...	P.CITY
S1	Smith	20	London	P1	Nut	...	London
S1	Smith	20	London	P4	Screw	...	London
S1	Smith	20	London	P6	Cog	...	London
S2	Jones	10	Paris	P2	Bolt	...	Paris
S2	Jones	10	Paris	P5	Cam	...	Paris
			...				

"`SELECT S.S#, P.P#`" selects columns mentioned.

S#	P#
S1	P1
S1	P4
S1	P6
S2	P2
S2	P5
...	

Get all pairs of city names such that a supplier located in the first city supplies a part stored in the second city.

```
SELECT S.CITY, P.CITY
FROM   S, SP, P
WHERE  S.S# = SP.S#
AND    SP.P# = P.P# ;
```

Get the total number of suppliers.

```
SELECT COUNT(*) AS N
FROM   S ;
```

Get the total, the maximum, and the minimum quantity for part P2.

```
SELECT SUM ( SP.QTY ) AS TOTQ,
       MAX (SP.QTY ) AS MAXQ,
       MIN ( SP.QTY ) AS MINQ
FROM   SP
WHERE  SP.P# = 'P2' ;
```

For each part supplied, get the part number and the total shipment quantity.

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

Conceptually, "`FROM SP GROUP BY SP.P#`" gives the following table.

{S#}	P#	{QTY}
{S1, S2}	P1	{300, 300}
{S1, S2, S3, S4}	P2	{200, 400, 200, 200}
{S1}	P3	{400}
{S1, S4}	P4	{200, 300}
{S1, S4}	P5	{100, 400}
{S1}	P6	{100}

"`SELECT SP.P#, SUM ( SP.QTY ) AS TOTQTY`" gives:

P#	TOTQTY
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

```
SELECT P.P#, ( SELECT SUM ( SP.QTY)
               FROM   SP
               WHERE  SP.P# = P.P# ) AS TOTQTY
FROM   P ;
```

The following query is erroneous!

```
SELECT SP.P#, SP.QTY
FROM   SP
GROUP  BY SP.P#
```

Get part number for all parts supplied by more than one supplier.

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

Get supplier names for suppliers who supply part P2.

```
SELECT DISTINCT S.SNAME
FROM   S
WHERE  S.S# IN
     ( SELECT SP.S#
       FROM   SP
       WHERE  SP.P# = 'P2' ) ;
```

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

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

Get supplier numbers for suppliers with status less than the current maximum status in the S table.

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

Get supplier names for suppliers who do not supply part P2.

```
SELECT DISTINCT S.SNAME
FROM   S
WHERE  S.S# NOT IN
     ( SELECT SP.S#
       FROM   SP
       WHERE  SP.P# = 'P2' ) ;
```

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

Get supplier names for suppliers who supply all red parts.

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

Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both.

```
SELECT P.P#
FROM   P
WHERE  P.WEIGHT > 16
UNION
SELECT SP.P#
FROM   SP
WHERE  SP.S# = 'S2' ;
```

Update

Single-row INSERT.

```
INSERT
INTO   P ( P#, PNAME, COLOR, WEIGHT, CITY )
VALUES ('P8', 'Sprocket', 'Pink', 14, 'Nice' ) ;
```

Multi-row INSERT.

```
INSERT
INTO   TEMP ( S#, CITY )
       SELECT S.S#, S.CITY
       FROM   S
       WHERE  S.STATUS > 15 ;
```

Multi-row UPDATE.

```
UPDATE P
SET    COLOR = 'Yellow',
       WEIGHT = P.WEIGHT + 5
WHERE  P.CITY = 'Paris' ;
```

Multi-row UPDATE.

```
UPDATE P
SET    CITY = ( SELECT S.CITY
                FROM   S
                WHERE  S.S# = 'S5' )
WHERE  P.COLOR = 'Red' ;
```

Multi-row DELETE.

```
DELETE
FROM   SP
WHERE  'London' =
     ( SELECT S.CITY
       FROM   S
       WHERE  S.S# = SP.S# ) ;
```

Views

```
CREATE VIEW REDPARTS ( P#, PNAME, WT, CITY )
    AS SELECT P.P#, P.PNAME, P.WEIGHT, P.CITY
       FROM   P
       WHERE  P.COLOR = 'Red' ;

CREATE VIEW PQ
    AS SELECT SP.P#, SUM ( SP.QTY ) AS TOTQTY
       FROM   SP
       GROUP  BY SP.P# ;
```

Rewriting of Select

Get red parts that weigh more than 15 pounds.

```
SELECT P#
FROM   REDPARTS
WHERE  WT > 15 ;
```

$\Leftrightarrow$

```
SELECT P#
FROM   P
WHERE  WEIGHT > 15
AND    COLOR = 'Red' ;
```

View Update

```
UPDATE REDPARTS
SET    WT = 454 * WT ;
```

$\Leftrightarrow$

```
UPDATE P
SET    WEIGHT = 454 * WEIGHT
WHERE  COLOR = 'Red'  ;
```

View Update Problem

```
UPDATE PQ
SET    TOTQTY = TOTQTY + 1 ;
```

$\Leftrightarrow$

```
UPDATE SP
SET    ???
```

Embedded SQL

```
EXEC SQL DECLARE X CURSOR FOR
         SELECT S.S#, S.SNAME, S.STATUS
         FROM   S
         WHERE  S.CITY = :Y ;  /* colon -> pgm variable */

EXEC SQL OPEN X ;                        /* execute the query */
EXEC SQL FETCH X INTO :V1, :V2, :V3 ;       /* fetch row */
WHILE a row is fetched LOOP
      ...                                  /* process the row */
      EXEC SQL FETCH X INTO :V1, :V2, :V3 ;
END-LOOP
EXEC SQL CLOSE X ;                      /* deactivate cursor X */
```