SQL summary

MJ Bezuidfenhout, credit to Jaques Steyn

June 19, 2017

1 Creating tables

1.1 Syntax

```
CREATE TABLE tablename (column1 data type [constraint] [,column2
    data type [constraint] ] [,

PRIMARY KEY (column1 [, column2]) ] [,FOREIGN KEY (column1 [,
    column2]) REFERENCES tablename] [,CONSTRAINT constraint ] );
```

1.2 Example

```
       CREATE TABLE VENDOR (V_CODE INTEGER NOT NULL UNIQUE,V_NAME
           VARCHAR(35) NOT NULL,V_CONTACT VARCHAR(25) NOT NULL,

V_AREACODE CHAR(3) NOT NULL,V_PHONE CHAR(8) NOT NULL,

V_STATE CHAR(2) NOT NULL,V_ORDER CHAR(1) NOT NULL,

PRIMARY KEY (V_CODE));
```

2 SQL Indexes

2.1 Syntax

```
CREATE [UNIQUE]INDEX indexname ON tablename(column1 [, column2])
```

2.2 Example

```
CREATE INDEX P_INDATEX ON PRODUCT(P_INDATE);

CREATE UNIQUE INDEX P_CODEX ON PRODUCT(P_CODE);
```

3 Adding Table Rows

3.1 Syntax

```
INSERT INTO tablename VALUES (value1, value2, , valuen)
```

3.2 Example

```
INSERT INTO VENDOR VALUES (21225,'Bryson, Inc.','Smithson
    ','615','223-3234','TN','Y');

INSERT INTO VENDOR VALUES (21226,'Superloo, Inc.','Flushing
    ','904','215-8995','FL','N');
```

4 Listing Table rows

4.1 Syntax

```
SELECT columnlist FROM tablename;
```

4.2 Example

```
SELECT * FROM PRODUCT;

SELECT P_CODE, P_DESCRIPT, P_INDATE, P_QOH, P_MIN, P_PRICE,
    P_DISCOUNT, V_CODE FROM PRODUCT;
```

5 Updating table Rows

5.1 Syntax

```
UPDATE tablename SET columnname = expression [, columnname =
    expression] [WHERE conditionlist ];
```

5.2 Example

```
UPDATE PRODUCT SET P_INDATE = '18-JAN-2016' WHERE P_CODE = '13-Q2
    /P2';

UPDATE PRODUCT SET P_INDATE = '18-JAN-2016', P_PRICE = 17.99,
    P_MIN = 10

WHERE P_CODE = '13-Q2/P2';
```

6 Deleting Table rows

6.1 Syntax

```
DELETE FROM tablename [WHERE conditionlist ];
```

6.2 Example

```
DELETE FROM PRODUCT WHERE P_CODE = 'BRT-345';
```

7 Selecting rows with conditional restrictions

7.1 Syntax

```
SELECT columnlist FROM tablelist [WHERE conditionlist ];
```

7.2 Example

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE FROM PRODUCT
WHERE V_CODE = 21344;
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE FROM PRODUCT
WHERE V_CODE <> 21344;
```

8 Logical Operators: AND, OR, NOT

8.1 Syntax

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE FROM PRODUCT
WHERE V_CODE = 21344 OR V_CODE = 24288;
```

8.2 Example

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE FROM PRODUCT
WHERE P_PRICE < 50 AND P_INDATE > '15-Jan-2016';

SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE FROM PRODUCT
WHERE (P_PRICE < 50 AND P_INDATE > '15-Jan-2016') OR V_CODE =
    24288;

SELECT * FROM PRODUCT WHERE NOT (V_CODE = 21344);
```

8.3 List of operators

```
BETWEEN: Used to check whether an attribute value is within a
    range
IS NULL: Used to check whether an attribute value is null
LIKE: Used to check whether an attribute value matches a given
    string pattern
IN: Used to check whether an attribute value matches any value
    within a value list
EXISTS: Used to check whether a subquery returns any rows
```

9 Aditional Data Definition Commands

9.1 Syntax

```
ALTER TABLE tablename {ADD | MODIFY} ( columnname datatype [ {ADD
     | MODIFY}
columnname datatype] );


ALTER TABLE tablename ADD constraint [ ADD constraint ];

ALTER TABLE tablename DROP {PRIMARY KEY | COLUMN columnname |
    CONSTRAINT
constraintname };
```

10 Ordering a Listing

10.1 Syntax

```
SELECT columnlist FROM tablelist [WHERE conditionlist ] [
    ORDER BY columnlist [ASC | DESC]];
```

10.2 Example

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE FROM PRODUCT ORDER BY
    P_PRICE;
```

11 Grouping Data

11.1 Syntax

```
SELECT columnlist FROM tablelist [WHERE conditionlist ][GROUP BY
    columnlist ][HAVING conditionlist ][ORDER BY columnlist [ASC
    | DESC] ];
```

11.2 Example

```
SELECT V_CODE, P_CODE, P_DESCRIPT, P_PRICE FROM PRODUCT GROUP BY
    V_CODE;
SELECT V_CODE, SUM(P_QOH * P_PRICE) AS TOTCOST FROM PRODUCT GROUP
     BY V_CODE HAVING (SUM(P_QOH * P_PRICE) > 500) ORDER BY SUM(
    P_QOH * P_PRICE) DESC;
```

12 Joining Database Tables

12.1 Syntax

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT,V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR WHERE PRODUCT.V_CODE = VENDOR.V_CODE;
```

12.2 Example

```
SELECT PRODUCT.P_DESCRIPT, PRODUCT.P_PRICE,VENDOR.V_NAME, VENDOR.
    V_CONTACT,VENDOR.V_AREACODE, VENDOR.V_PHONE
```

```
FROM PRODUCT, VENDOR WHERE PRODUCT.V_CODE = VENDOR.V_CODE
ORDER BY PRODUCT.P_PRICE;

SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT,V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR WHERE PRODUCT.V_CODE = VENDOR.V_CODE
AND P_INDATE > '15-Jan-2016';

SELECT CUS_LNAME, INVOICE.INV_NUMBER, INV_DATE, P_DESCRIPT
FROM CUSTOMER, INVOICE, LINE, PRODUCT
WHERE CUSTOMER.CUS_CODE = INVOICE.CUS_CODE
AND INVOICE.INV_NUMBER = LINE.INV_NUMBER AND LINE.P_CODE =
    PRODUCT.P_CODE AND CUSTOMER.CUS_CODE = 10014
ORDER BY INV_NUMBER;
```

13 Joining Database Tables using an Alias

13.1 Syntax

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE,V_PHONE
FROM PRODUCT P, VENDOR V WHERE P.V_CODE = V.V_CODE ORDER BY
    P_PRICE;
```

14 Recursive Joins

14.1 Syntax

```
       FROM EMP E, EMP M WHERE E.EMP_MGR=M.EMP_NUM ORDER BY E.
           EMP_MGR;
```

15 SQL Join operators

15.1 Syntax

```
SELECT P_CODE, P_DESCRIPT, P_PRICE, V_NAME FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE;
```

16 Cross Joins

16.1 Syntax

```
SELECT column-list FROM table1 CROSS JOIN table2
```

16.2 example

```
SELECT * FROM INVOICE CROSS JOIN LINE;

SELECT INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE
FROM INVOICE CROSS JOIN LINE;

SELECT INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE
FROM INVOICE, LINE;
```

17 Natural Joins

17.1 Syntax

```
SELECT column-list FROM table1 NATURAL JOIN table2
```

17.2 example

```
SELECT CUS_CODE, CUS_LNAME, INV_NUMBER, INV_DATE
FROM CUSTOMER NATURAL JOIN INVOICE;

SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
FROM INVOICE NATURAL JOIN LINE NATURAL JOIN PRODUCT;
```

18 Join Using Clause

18.1 Syntax

```
SELECT column-list FROM table1 JOIN table2 USING (common-column)
```

18.2 Example

```
SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
FROM INVOICE JOIN LINE USING (INV_NUMBER) JOIN PRODUCT
USING (P_CODE);
```

19 Join ON Clause

19.1 Syntax

```
SELECT column-list FROM table1 JOIN table2 ON join-
    condition
```

19.2 Example

```
SELECT INVOICE.INV_NUMBER, PRODUCT.P_CODE, P_DESCRIPT, LINE_UNITS
    , LINE_PRICE
FROM INVOICE JOIN LINE ON INVOICE.INV_NUMBER = LINE.INV_NUMBER
JOIN PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;

SELECT E.EMP_MGR, M.EMP_LNAME, E.EMP_NUM, E.EMP_LNAME
FROM EMP E JOIN EMP M ON E.EMP_MGR = M.EMP_NUM ORDER BY E.EMP_MGR
    ;
```

20 Outer Joins

20.1 Syntax

```
SELECT column-list FROM table1 LEFT [OUTER] JOIN table2 ON join-
    condition
```

21 Example

```
SELECT P_CODE, VENDOR.V_CODE, V_NAME
FROM VENDOR LEFT JOIN PRODUCT ON VENDOR. V_CODE = PRODUCT.V_CODE;
```

21.1 Explanation

```
The right outer join returns not only the rows matching the join
    condition (that is, rows with matching values in the common
    column), it returns the rows in the right table with
    unmatched values in the left table. The syntax is:
SELECT column-list FROM table1 RIGHT [OUTER] JOIN table2 ON join-
    condition

SELECT P_CODE, VENDOR.V_CODE, V_NAME
FROM VENDOR RIGHT JOIN PRODUCT ON VENDOR. V_CODE = PRODUCT.V_CODE
    ;

The full outer join returns not only the rows matching the join
    condition (that is, rows with matching values in the common
    column), it returns all of the rows with unmatched values in
    the table on either side. The syntax is:
SELECT column-list FROM table1 FULL [OUTER] JOIN table2 ON join-
    condition

SELECT P_CODE, VENDOR.V_CODE, V_NAME
FROM VENDOR FULL JOIN PRODUCT ON VENDOR.
V_CODE = PRODUCT.V_CODE;
```

22 Triggers

22.1 syntax

```
       CREATE OR REPLACE TRIGGER trigger_name
[BEFORE / AFTER] [DELETE / INSERT / UPDATE OF column_name] ON
    table_name
[FOR EACH ROW]
[DECLARE]
[variable_namedata type[:=initial_value] ]
BEGIN
PL/SQL instructions;
```