Introduction to Microsoft Jet SQL

Microsoft Jet SQL is a relational database language based on the SQL 1989 standard of the American Standards Institute (ANSI). Microsoft Jet SQL contains two kinds of statements:

- Data Definition Language (DDL) statements. These statements let you define and modify the schema of your database.

- Data Manipulation Language (DML) statements. These statements let you create, access and modify data in your database.

Statement	Description
CREATE TABLE	Defines a table in a database
DROP TABLE	Deletes a table from a database
ALTER TABLE	Modifies a table in a database
CREATE INDEX	Adds an index to a table
DROP INDEX	Deletes an index from a table

Table 1 Some DDL statements

Statement	Description
SELECT	Retrieves and displays one or more rows from a table.
INSERT INTO	Adds new rows to a table
UPDATE	Changes column values in one or more rows of a table
DELETE FROM	Deletes one or more rows from a table
UNION	Combines two or more SELECT statements to create a complex query

Table 2. Some DML statements

Data Definition Language

Creating Tables

The CREATE TABLE statement creates a new, empty table with the columns that you specify. Here is an example that creates a table named sp:

```
CREATE TABLE SP
( SNUM TEXT NOT NULL ,
  PNUM TEXT NOT NULL,
  QTY     SHORT NOT NULL,
  CONSTRAINT  one PRIMARY KEY (SNUM,PNUM),
  CONSTRAINT two FOREIGN KEY (SNUM) REFERENCES S (SNUM),
  CONSTRAINT three FOREIGN KEY (PNUM) REFERENCES   P
(PNUM));
```

The table SP has three columns; SNUM, PNUM, and QTY. For each column you name, you must enter a data type for that column. To indicate that the column must have a value, use the optional keyword NOT NULL. The CONSTRAINT instruction can be used to specify primary keys, foreign keys, and candidate keys. To specify that a column is a candidate key, use the keyword UNIQUE.

A CREATE TABLE statement can contain zero or more CONSTRAINT instructions (a primary key is therefore not necessary to specify!). For every key that is specified, an index is automatically created.

Deleting Tables

The DROP TABLE statement deletes a table from the database. It also deletes all indexes defined on that table, and all data associated with the table. The following example deletes the table SUPPLIERS:

```
DROP TABLE SUPPLIERS;
```

Altering Tables

The ALTER TABLE statement lets you add new columns to a table or delete columns from a table. To add a new column DISCOUNT to the SUPPLIERS table you write:

```
ALTER TABLE SUPPLIERS ADD COLUMN DISCOUNT SINGLE;
```

To delete the same column you write:

```
ALTER TABLE SUPPLIERS DROP COLUMN DISCOUNT;
```

Creating Indexes

The CREATE INDEX statement creates an index on one or more columns. The following example creates an index XCITY on SUPPLIERS.

```
CREATE INDEX XCITY on SUPPLIERS (CITY);
```

Deleting Indexes

The DROP INDEX statement deletes one or more indexes from the database. To delete the index XCITY write:

```
DROP INDEX XCITY;
```

Data Types

Table 3 lists some of the data types that SQL supports.

SQL Data Type	Size	Description
BINARY		Data of this type is stored in binary, i. e. no conversion is done
BYTE	1 byte	Integer between 0 and 255
COUNTER	4 bytes	A number that is automatically incremented when a new record is inserted
CURRENCY	8 bytes	Values between –922 337 203 685 477,5808 and 922 337 203 685 477,5807
DATETIME	8 bytes	Date or time (years 100 – 9999)
SINGLE	4 bytes	Decimal number with single precision
DOUBLE	8 bytes	Decimal number with double precision
SHORT	2 bytes	Integer between –32 768 and 32 767
LONG	4 bytes	Integer between –2 147 483 648 and 2 147483647
LONGTEXT	1 byte/char	Between 0 and 1,2 GB
LONGBINARY		Between 0 and 1,2 GB
TEXT	1 byte/char	Up to 255 characters

Table 3. Microsoft Jet SQL Data Types

Some Words on Syntax

Each Microsoft Jet SQL statement must end with a semicolon. Any number of spaces, tabs, and newline characters are treated as a single space.

When you use the same column name in more than one table and those columns are referenced in the same query, you must qualify the column names with their table names (e.g. suppliers.city and parts.city in the suppliers and parts database).

A character string or a date must be enclosed in either single or double quotation marks.

There exist two wildcard characters that can be used in LIKE clauses to match character strings. The asterisk character (*) matches zero or more characters. The question mark character (?) matches any single character.

Data Manipulation Language

Inserting Rows

The INSERT INTO statement inserts one or more rows into an existing table. The data you insert can be a list of values that you supply or values from another table.

Add a new part record to PARTS. (By including the field list, the color field can be omitted)

```
INSERT INTO PARTS( P_NUM, CITY, WEIGHT)  VALUES( ‘P7’,
‘Athens’, 24);
```

Add a new record to PARTS (By omitting the field list, all values must be present)

```
INSERT INTO PARTS VALUES( ‘P8’, ‘Sprocket’, ‘pink’, 14, ‘Nice’);
```

Add those suppliers who have a status greater than 15 to a table HIGH_STATUS (Multirow insert).

```
INSERT INTO HIGH_STATUS (S_NUM, SNAME, STATUS, CITY)
            SELECT * FROM SUPPLIERS
            WHERE STATUS > 15;
```

Updating Rows

The UPDATE statement changes column values in one or more rows of a table.

Change the color of part P2 to yellow, increase the weight by 5 and set the city to unknown (single row update)

```
UPDATE P
SET  COLOR = ‘yellow’
        WEIGHT = WEIGHT + 5
        CITY = NULL
WHERE P_NUM = ‘P2’;
```

Double the status of all suppliers in London (multi row update).

```
UPDATE SUPPLIERS
SET  STATUS = 2 * STATUS
WHERE  CITY = ‘London’;
```

Deleting Rows

The DELETE statement deletes column values in one or more rows of a table.

Delete supplier S5 (single row DELETE)

```
DELETE  FROM  S  WHERE  S_NUM = ‘S5’;
```

Delete all shipments with quantity greater than 300 (multirow delete)

```
DELETE  FROM  SHIPMENTS  WHERE  QTY > 300;
```

Delete all shipments (SHIPMENTS is then empty!)

```
DELETE  FROM  SHIPMENTS;
```

SELECT Statement

The basic SELECT statement retrieves and displays as many rows of data as satisfy the selection criteria you specify.

Select all suppliers who are located in Paris.

```
SELECT  *
FROM SUPPLIERS
WHERE CITY = ‘Paris’;
```

The asterix(*) here means that all columns are selected.

A WHERE clause consists of one or more search conditions connected by the logical operators AND, OR and NOT.

Get all supplier citys

```
SELECT DISTINCT CITY
FROM SUPPLIERS;
```

Without the keyword DISTINCT, the result would contain five rows. DISTINCT removes all duplicate rows and hence the result will contain only two rows.

Get color and city for "nonParis" parts with weight between 13 and 18

```
SELECT COLOR, CITY
FROM PARTS
WHERE CITY <> 'PARIS'
AND WEIGHT BETWEEN 13 AND 18;
```

Select all suppliers who are located in a city that has an ’o’ as its second letter

```
SELECT *
FROM SUPPLIERS
WHERE CITY LIKE   "?o*";
```

When using LIKE, the question mark character (?) matches any single character and the asterisk (*) matches any sequence of characters.

Select all parts where the color is unknown

```
SELECT *
FROM PARTS
WHERE COLOR IS NULL;
```

Likewise, parts where the color is known can be selected with the condition WHERE COLOR IS NOT NULL.

Select all parts from London, Paris and Rome

```
SELECT *
FROM PARTS
WHERE CITY IN ("London", "Paris", "Rome");
```

Instead of OR, the IN syntax together with a group of values can be used.

Selecting Data from Multiple Tables (Joins)

Selecting data from multiple tables using a single SELECT statement is referred to as a join operation.

Do a natural join of suppliers and parts over city

```
SELECT SNUM, SNAME, STATUS, SUPPLIER.CITY, PNUM, PNAME,
COLOR, WEIGHT
FROM SUPPLIERS, PARTS
WHERE SUPPLIERS.CITY = PARTS.CITY;
```

Note: If the condition is left out, the result will be the cartesian product.

This query can also be written using the INNER JOIN operation:

```
SELECT SNUM, SNAME, STATUS, SUPPLIER.CITY, PNUM, PNAME,
COLOR, WEIGHT
FROM SUPPLIERS INNER JOIN PARTS
ON SUPPLIERS.CITY = PARTS.CITY;
```

Select all pairs of suppliers such that the two suppliers concerned are colocated.

```
SELECT SUPPLIERS.SNUM, SECOND.SNUM
FROM SUPPLIERS, SUPPLIERS AS SECOND
WHERE SUPPLIERS.CITY = SECOND.CITY
AND SUPPLIERS.SNUM < SECOND.SNUM.
```

By using an alias (in this case SECOND) rows within the same table can be compared. A copy of the table SUPPLIERS called SECOND is created for this purpose.

Aggregate functions

Function	**Meaning**
AVG(*expression*)	Calculates an average value for all field values in the result list
COUNT(*)	Counts the number of rows in the result list (this count includes rows with NULL values).
COUNT(*expression*)	Counts the number of rows in the result list (this count does not include rows with NULL values).
MAX(*expression*)	Returns the maximum value of *expression* for all rows.
MIN(*expression*)	Returns the minimum value of *expression* for all rows.
SUM(*expression*)	Calculates the sum of expression for all rows.
STDEV(*expression*)	Returns the standard deviation for the values in *expression*
VAR(*expression*)	Returns the variance for the values in *expression*

Get the total number of suppliers

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

Get the total number of suppliers in London

```
SELECT COUNT (CITY)
FROM SUPPLIERS
WHERE CITY = ‘London’;
```

Get the maximum, minimum and average weight from parts.

```
SELECT MAX(WEIGHT), MIN(WEIGHT), AVG(WEIGHT)
FROM PARTS;
```

Get the total quantity of part P2 supplied

```
SELECT SUM(QTY)
FROM SHIPMENT
WHERE PNUM = 'P2';
```

Grouping Data

A group is a set of rows that has the same value for a specified column or columns. The GROUP BY clause results in a single row in the result table for each group of rows.

Get the part number and the total quantity shipped for each part

```
SELECT PNUM, SUM(QTY)
FROM SHIPMENT
GROUP BY PNUM
```

Get part numbers for all parts supplied by more than one supplier

```
SELECT PNUM
FROM SHIPMENT
GROUP BY PNUM
HAVING COUNT (SNUM) >1.
```

The HAVING clause is to groups what the WHERE clause is to rows; in other words, HAVING is used to eliminate groups, just as WHERE is used to eliminate rows.

Sorting Data

The ORDER BY clause sorts query results by the values in one or more columns. You can specify the columns by name or by order. If you want to sort the results in descending order, you can use the keyword DESC. Ascending is the default sort order.

Get all suppliers in ascending order

```
SELECT *
FROM SUPPLIERS
ORDER BY SNAME.
```

Get the part number and the total quantity shipped for each part. The results should be sorted so the part with the largest total quantity comes first.

```
SELECT PNUM, SUM(QTY)
FROM SHIPMENT
GROUP BY PNUM
ORDER BY 2 DESC.
```

Using Subqueries

The Query **"Get supplier names for suppliers who supply red parts"** is usually written:

```
SELECT SNAME
FROM SUPPLIERS INNER JOIN
(SHIPMENT INNER JOIN PARTS
ON SHIPMENT.PNUM  =   PARTS.PNUM)
ON SUPPLIERS.SNUM  =   SHIPMENT.SNUM
AND  PARTS.COLOR    =   'Red'.
```

But can also be written using subqueries:

```
SELECT SNAME
FROM SUPPLIERS
WHERE SNUM IN
            ( SELECT SNUM
              FROM SHIPMENT
              WHERE PNUM IN
                        ( SELECT PNUM
                          FROM PARTS
                          WHERE COLOR = 'Red' )).
```

The result of a subquery is used as a value in the WHERE clause. You must enclose a subquery in parentheses (), and it can have only one column in its SELECT list.

Another type of subqueries use the EXISTS clause:

Get supplier names for suppliers who supply part P2

```
SELECT SNAME
FROM SUPPLIERS
WHERE EXISTS
                  ( SELECT *
                    FROM SHIPMENT
                    WHERE SNUM = SUPPLIERS.SNUM
                    AND PNUM = 'P2' ).
```

Each SNAME is tested to see if its SNUM causes the EXISTS test to evaluate to true.

It is also possible to use subqueries involving NOT IN and NOT EXISTS.

The UNION Statement

You can use the UNION statement to combine the results of two or more SELECT statements.

Get part numbers for parts that either weigh more than 16 OR are supplied by supplier S2 (or both)

```
SELECT PNUM
FROM PARTS
WHERE WEIGHT > 16
UNION
SELECT PNUM
 FROM SHIPMENT
WHERE SNUM = ’S2’.
```

Parameters

Parameters can be used in questions to change a condition automatically.

Ex:

```
PARAMETERS [Choose a supplier number ]  text, [Choose a part] text;

SELECT S.SNUM, S.SNAME, P.PNUM, P.PNAME, QTY
FROM S INNER JOIN (SP INNER JOIN P ON SP.PNUM = P.PNUM)
ON S.SNUM = SP.SNUM
WHERE S.SNUM = [Choose a supplier number]
AND P.PNAME = [Choose a part] ;
```

Each parameter has a *name* and a *data type*. When the question is executed, a dialog box is shown for each parameter where you are asked to type in a value.