CHAPTER 7: QUERYING A DATABASE

Definition and Meaning of Database Query:

Queries are the primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. Many database management systems use the Structured Query Language (SQL) standard query format.

SQL Features of Query language (SQL)

General Features	
Multi-platform Support	Supports all major DBMSs from a single interface. Ability to use the tools on all supported platforms from a single license.
Embarcadero® AppWave™	Enables centralized license management and tool deployment
Unicode Support	Full Unicode character support throughout the application
Intuitive Interface	Automates common and repetitive tasks with easy-to-use SQL editors and wizards
SQL Scripting and Editing	
Visual Query Builder	Constructs even the most complicated SQL statements with point-and-click ease
Code Templates	Eliminates the need to memorize and type SQL syntax
SQL Editor	Code folding, code collapse/roll-up, syntax coloring, hot key assignments, configurable auto replace of objects, bind variable support, selective statement execution
Context-sensitive DMBS Actions	DBMS actions, such as Extract and Drop, are available directly in the context menu of the appropriate tokens in the SQL editor
Debugging, Performance Optimization	
Code Analyst	Performs detailed response time analysis on the execution of stored procedures and functions

SQL Debugger	Debugs programmable objects such as stored procedures, functions, packages, and triggers. Available for DB2 for LUW, Oracle, SQL Server, and Sybase
SQL Profiler	Captures metrics of various PL/SQL programmable objects on Oracle 8.1.5 and higher.
Developer Features	
Advanced Code Assist	Lists context-sensitive suggestions as you type (e.g., tables, columns, procedures, functions, and code templates) and is available offline
Code Formatting and Profiles	Code folding, syntax coloring, comment toggling, and other auto formatting features make it easy to read, navigate, and edit large SQL files. Customize and share various SQL formatting options by creating SQL formatting profiles
Context-sensitive DBMS Actions	DBMS actions, such as Extract and Drop, are available directly in the context menu of the appropriate tokens in the SQL editor
Syntax & Semantic Validation	Validate SQL files and flags all DBMS-specific parser violations or references to objects not found in the target database
Quick Fixes	Real-time parsing provides code quality suggestions for improving SQL performance as you type
SQL Debugging	Debug Java, step seamlessly into SQL (i.e. stored procedure) and back into Java again – true system-wide, round-trip debugging
Data Governance*	
Inline Metadata	Gives users real-time metadata visibility in the SQL IDE and will gain valuable context in SQL query development with awareness of sensitive data. Examples of metadata attributes are: table descriptions, PII, data governance policy information, etc.
Centralized Datasource Repository	Provides the functionality for users to work from a common, centralized datasource repository

SQL Commands and categories:

SQL commands are instructions, coded into SQL statements, which are used to communicate with the database to perform specific tasks, work, functions and queries with data.

SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can *create tables*, add data to tables, or modify data, drop the table, set permissions for users.

SQL commands are grouped into four major categories depending on their functionality:

- Data Definition Language (DDL) - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.
- Data Manipulation Language (DML) - These SQL commands are used for storing, retrieving, modifying, and deleting data.
  These Data Manipulation Language commands are: SELECT INSERT, UPDATE, and DELETE.
- Transaction Control Language (TCL) - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.
- Data Control Language (DCL) - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

SQL statements/Queries design and interrogation of database

SQL statement design

✓ **The CREATE DATABASE Statement**

The CREATE DATABASE statement is used to create a database.

SQL CREATE DATABASE Syntax

CREATE DATABASE database_name

CREATE DATABASE Example

Now we want to create a database called "my_db".

We use the following CREATE DATABASE statement:

```
CREATE DATABASE my_db
```

Database tables can be added with the CREATE TABLE statement.

The CREATE TABLE Statement

The CREATE TABLE statement is used to create a table in a database.

SQL CREATE TABLE Syntax

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

CREATE TABLE Example

Now we want to create a table called "Persons" that contains five columns: P_Id, LastName, FirstName, Address, and City.

We use the following CREATE TABLE statement:


```
CREATE TABLE Persons
(
P_Id int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

The P_Id column is of type int and will hold a number. The LastName, FirstName, Address, and City columns are of type varchar with a maximum length of 255 characters.

The empty "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City

The empty table can be filled with data with the INSERT INTO statement.

✓ **Describe command**

The describe command gives you a list of all the data fields used in your database table. In the example, you can see that the table named test in the sales data database keeps track of four fields: name, description, num, and date_modified.

```
mysql> describe test;
+---------------+-------------+------+-----+------------+----------------+
| Field         | Type        | Null | Key | Default    | Extra          |
+---------------+-------------+------+-----+------------+----------------+
| num           | int(11)     |      | PRI | NULL       | auto_increment |
| date_modified | date        |      | MUL | 0000-00-00 |                |
| name          | varchar(50) |      | MUL |            |                |
| description   | varchar(75) | YES  |     | NULL       |                |
+---------------+-------------+------+-----+------------+----------------+
6 rows in set (0.00 sec)
```


SQL database interaction

✓ **The INSERT INTO Statement**

The INSERT INTO statement is used to insert a new row in a table.

SQL INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
```

```
VALUES (value1, value2, value3,...)
```

SQL INSERT INTO Example

We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to insert a new row in the "Persons" table.

We use the following SQL statement:

```
INSERT INTO Persons
VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')
```


The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

Insert Data Only in Specified Columns

It is also possible to only add data in specific columns.

The following SQL statement will add a new row, but only add data in the "P_Id", "LastName" and the "FirstName" columns:

```
INSERT INTO Persons (P_Id, LastName, FirstName)
VALUES (5, 'Tjessem', 'Jakob')
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

✓ The UPDATE Statement

The UPDATE statement is used to update existing records in a table.

SQL UPDATE Syntax

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

SQL UPDATE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

Now we want to update the person "Tjessem, Jakob" in the "Persons" table.

We use the following SQL statement:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob'
```


The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

SQL UPDATE Warning

Be careful when updating records. If we had omitted the WHERE clause in the example above, like this:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
```

The "Persons" table would have looked like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Nissestien 67	Sandnes
2	Svendson	Tove	Nissestien 67	Sandnes
3	Pettersen	Kari	Nissestien 67	Sandnes
4	Nilsen	Johan	Nissestien 67	Sandnes
5	Tjessem	Jakob	Nissestien 67	Sandnes

More SQL commands

1. **Create a database on the sql server.**

   mysql> create database [databasename];

2. **List all databases on the sql server.**

   mysql> show databases;

3. **Switch to a database.**

   mysql> use [db name];

4. **To see all the tables in the db.**

   mysql> show tables;

5. **To see database's field formats.**