The Structured Query Language – Get Started

©Himadri Barman

0. Prerequisites:

A database is an organized collection of related data that can easily be retrieved and used. By data, we mean known facts that can easily be recorded and have implicit meaning. A DBMS is a collection of programs that enables proper maintenance and utilization of the database.

Relational databases are so named after the data model the follow, i.e., the Relational Data Model. This model is a form of database specification based upon the mathematical concept of a relation, i.e., the database is represented as a collection of relations.

A row is called a tuple, a column header is called an attribute, and the table is called a relation. A relation is thus a set of tuples. By definition, all elements of a set are distinct, hence all tuples must be distinct.

1. Introduction:

SQL is the de-facto standard of database access languages. Originally, SQL was called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research as the interface or an experimental relational database system called SYSTEM R. A joint effort by ANSI and ISO has led to the standard version of SQL (ANSI 1986) called SQL – 86 or SQL1. A revised and much expanded standard called SQL2 (SQL – 92) has subsequently been developed. Work is going on for SQL3, which will incorporate object – oriented and other recent database concepts.

SQL provides a high-level *declarative* language interface, so the user specifies only as to what the results should be, leaving the optimizations and how to execute the query to the DBMS. It provides commands for a variety of tasks including:

- ü Creating the database structure
- ü Querying data
- ü Controlling access to the database and its objects
- ü Guaranteeing database consistency

This tutorial is based on using SQL* Plus on an Oracle database. SQL* Plus is an interactive program that allows you to type in and execute SQL statements. It draws extensively on Oracle's *Procedural Language* extension of SQL called PL/SQL.

2. Types of SQL statements

2.1 Data Definition Language (DDL)

It is the SQL construct used to define data in the database. When the data is defined, entries are made in the database's data dictionary. Common DDL keywords are **create**, **revoke**, **grant**, **drop**, **alter**, etc. These are auto-commit keywords.

2.2 Data Manipulation Language (DML)

It is the SQL construct used to manipulate the data in the database. Common DML keywords are **select**, **insert**, **update** and **delete**.

2.3 Transaction Control Language (TCL)

These keywords manage changes made by DML keywords. The TCL keywords are **commit**, **rollback**, etc.

3. SQL Data Types

The following table lists the most common data types, which will get you started. The table below is not exhaustive. It is to be kept in mind that the data types available depend on the system. For e.g., the varchar data type is automatically converted to varchar2 data type in Oracle.

Data Type	Description
char(size)	Stores fixed-length character data, with a maximum size of 2000
varchar(size)	Stores variable-length character data type, with a maximum size of 4000
number(l,d)	Stores numeric data, where "l" stands for length and "d" stands for the number of decimal digits
date	Stores dates from January 1, 4712 BC to December 31, 9999 AD
long	Stores variable-length character data up to 2 GB

4. Creating the Database

The use of data types is apparent – in creating the relational database consisting of tables with its fields(columns). The data in these fields are of a specific data type. We start by creating a ***Student Database*** – for simplicity, we will have two tables in the database – the first table will be called *student_info*, containing personal details while the second table will be called *student_course_info*, containing information on the course he/she is pursuing. The fields in the *student_info* table may be stuid, name, dob, address, etc. The fields in the *student_course_info* may be stuid, course_name, course_type, fees, exam_fees etc.

The CREATE TABLE statement is used to create the table. The syntax is –

➡ **CREATE TABLE *table_name* (*column_name column_data_type*)**

For e.g.,
CREATE TABLE student_info (sid char(2), name varchar(30), dob date, address varchar(50))

The structure of the tables created may be viewed by giving the command which is an ORACLE enhancement -
➡ **DESC(RIBE) *table_name***

The INSERT statement is used to enter data into the tables created. The syntax is –
➡ **INSERT into *table_name* [*column_name*] values (*values*)**
The column_name is optional in the sense that it can be used to selectively enter values into the desired column(s).

For e.g.,
INSERT into student_info values ('123','Himadri','28 – SEP - 1978','Guwahati-781020')
INSERT into student_info (name) values ('Utpal')

☞ In Oracle's SQL* plus, a statement is ended by using **;**.

5. Modifying the Database

Once a table structure is made, it can be modified but with some restrictions. We can change the data type for the fields, add fields, etc. Some databases even allow dropping fields but SQL as such doesn't.

The ALTER TABLE statement is used to modify the tables created. The syntax is
➡ **ALTER TABLE *table_name* {as per requirements}**

For e.g.
ALTER TABLE student_info modify (name varchar(35))
ALTER TABLE student_info add (age number(2,0))

You can change more than one column at a time. For e.g.,
ALTER TABLE student_info add (age number(2,0), sex char(1))

Data that is already in the database may be updated as well as deleted. The syntaxes are –
➡ **UPDATE *table_name* set *column_name=value* where [criteria statements]**
➡ **DELETE from *table_name* where [criteria statements]**

For e.g.,
UPDATE student _info name='Himanshu' where name='Himadri'
DELETE from student_info where name='Himanshu'

The DROP TABLE statement is used to delete tables. The syntax is

➡ **DROP TABLE *table_name***

6. Querying the Database

An important component of databases is the act of retrieving information from it. This method is known as querying and statements that accomplish this are called query statements.

The SELECT statement is used to query the database. The syntax is –

➡ **SELECT [*column_name(s)* / *] from *table-name(s)* where [criteria statements] order by [column_name(s) asc/desc]**

For e.g.,
SELECT * from student_info where address=’Guwahati– 781020’ order by sid asc

The above query will select all students who have their *address* as Guwahati – 781020 and the results will be displayed after sorting them by their *sid* in the ascending order.

Including more than one table and including multiple criterions, one may construct more complex queries, which will be explored later on.

7. Relational Operators

7.1 Comparision Operators

Operator	**Purpose**	**Example**
=	Test for equality	select * from student__info where sid=’123’
!= , <>, ^=	Test for inequality	
<	Less than	
>	Greater than	
<=	Less than or equal to	
>=	Greater than or equal to	
in	Equal to any member in parentheses	select name from student_info where sid in(‘123’,’124’)
not in	not equal to any member in parentheses	
between [a] and [b]	Greater than or equal to a and less than or equal to b	select * from student_info where age between 25 and 30
not between [a] and [b]	Not greater than or equal to a and not less than or equal to b	

like ‘%text%’	Contains given text	select * from student_info where name like ‘%imad%’. % replaces arbitrary number fo characters. Use _ to replace fixed number of characters.

7.2 Mathematical Operators

Operator	Purpose	Example
+	Addition	select fees+exam_fees from student__course_info where sid=’123’
-	Subtraction	update student_course_info set fees=fees-1000
*	Multiplication	
/	Division	

7.3 Logical Operators

Operator	Purpose	Example
and	Test for conditions simultaneously	select * from student__info where name=’Himadri’ and dob=’28 – SEP - 1978’
or	Test for any of the given conditions	
not	Test for non – occurrence of the condition	select * from student_info where not name=’Himadri’

8. Aggregate Functions

Function	Description	Usage
count	Returns the number of entries in the table	select count(name) from student_info
sum	Returns the sum	
max		
min		
avg		

9. Constraints

9.1 Keys

A minimal set of attributes that uniquely identify a tuple is called a *key*. The key that is used to uniquely identify a tuple is called the *primary key*.

A non-minimal set of attributes uniquely identifying a tuple is called a *superkey*. A *superkey* have redundant attributes.

Each of the keys is called a *candidate key*.

9.2 Foreign Keys

A key of a relation R1 is called a *foreign key* if it is used to refer to another relation R2 satisfying the rules –

(1) FK of R1 have the same domain as the domain (set of values) of PK of R2.
(2) FK can be null.

R1 is called the referencing relation and R2 is called the referenced relation. A relation can refer to itself with the help of the *foreign key*.

9.2 Integrity Constraints

(1) The *Entity Integrity Constraint* states that n no primary key value can be null.
(2) *Referential Integrity Constraint* is specified between two relations (tables) and is used to maintain the consistency among tuples of the two relations.

9.2 Specifying Constraints in SQL

The CONSTRAINT keyword is used to specify constraints.If we specify the constraints at the time of creating the table , the syntax is –

- **CONSTRAINT *constraint_name* PRIMARY KEY(*column_name)***
- **CONSTRAINT *constraint_name* FOREIGN KEY*(column_name)* REFERENCES *referencing_relation(primary_key_column_name)***

For e.g.,

CONSTRAINT stuinfopk PRIMARY KEY(sid)
CONSTRAINT stuinfocoursefk FOREIGN KEY(stuid) REFERENCES student_info(sid)

If we specify the constraints after designing the table, we use the ALTER TABLE statement in the following way –

ALTER TABLE student_info add CONSTRAINT stuinfopk PRIMARY KEY(sid)

It should always be kept in mind that adding constraints is a tricky business, but is the core idea behind implementing a relational database. It is therefore advisable to clearly plan beforehand to avoid the potential pitfalls.