

Structured Query Language (SQL)

Anurag Srivastava

Structured Query Language (SQL)

Technology Offerings

Anurag Srivastava



TATA CONSULTANCY SERVICES

Introduction

Structured Query Language (SQL) is a standard language used to communicate with relational database management systems, including Oracle, Microsoft SQL Server, Sybase, Informix, and Microsoft Access.

The prototype was originally developed by IBM using Dr. E.F. Codd's paper ("A Relational Model of Data for Large Shared Data Banks") as a model. In 1979, the first SQL product, ORACLE, was released by Relational Software, Incorporated (which was later renamed Oracle Corporation).

SQL is a 4GL. 4GLs are closer to natural language than 3GL. Languages for accessing databases are 4GLs.



How SQL Works

The strengths of SQL provide benefits for all types of users, including application programmers, database administrators, managers, and end users.

Technically speaking, SQL is a data sublanguage. The purpose of SQL is to provide an interface to a relational database such as Oracle Database, and all SQL statements are instructions to the database. In this, SQL differs from general-purpose programming languages like C and BASIC.

SQL Statements can be divided into the following major categories

► Data Definition Language (DDL)

Statements are used to build and modify the structure of your tables and other objects in the database. E.g.: CREATE, ALTER, DROP etc.

► Data Manipulation Language (DML)

Statements are used for managing data within schema objects. E.g. Select, Insert, Update, Merge.

► Data Control Language (DCL)

Statements are used for securing the database. DCL Statement control access to database. E.g. Grant, Revoke etc.

Creating Database Objects Data Definition Language (DDL)

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Synonym	Gives alternative names to objects
Index	Improves the performance of some queries



The CREATE TABLE Statement

You must have:

- CREATE TABLE privilege
- A storage area

```
CREATE TABLE [schema.]table
    (column datatype [DEFAULT expr][, ...]);
```

```
CREATE TABLE dept    (deptno    NUMBER(2),
                       dname      VARCHAR2(14),
                       loc        VARCHAR2(13));
```

Table created.

Confirm table creation - DESCRIBE dept;

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

The DEFAULT Option

- Specify a default value for a column during an insert.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

Querying the Data Dictionary

- See the names of tables owned by the user.

```
SELECT table_name
FROM   user_tables ;
```

- View distinct object types owned by the user.

```
SELECT DISTINCT object_type
FROM   user_objects ;
```



The ALTER TABLE Statement

After you create a table, you may need to change the table structure because of the following:

- Omission of a column
- Change in column definition
- Removal of columns

You can do this by using the

ALTER TABLE statement.

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column
- Rename table

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement.

```
COMMENT ON TABLE employees  
IS 'Employee Information';  
Comment created.
```

- Comments can be viewed through the data dictionary views:
 - USER_COL_COMMENTS
 - USER_TAB_COMMENTS



Constraints

What are Constraints?

Constraints enforce rules at the table-column level.

Constraints prevent the deletion of a table if there are dependencies.

The following constraint types are valid:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Defining Constraints

➤ Column constraint level

```
column [CONSTRAINT constraint_name] constraint_type,
```

➤ Table constraint level

```
column, ...  
[CONSTRAINT constraint_name] constraint_type  
(column, ...),
```

Constraints are usually created at the same time as the table. Constraints can be added to a table after its creation and also temporarily disabled.



The NOT NULL Constraint

- Ensures that null values are not permitted for the column

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

...

20 rows selected.

NOT NULL constraint
(No row can contain
a null value for
this column.)

**NOT NULL
constraint**

**Absence of NOT NULL
constraint**
(Any row can contain
null for this column.)

- The NOT NULL constraint can be specified only at the column level, not at the table level.

```
CREATE TABLE employees(
  employee_id    NUMBER(6),
  last_name      VARCHAR2(25) NOT NULL,
  salary         NUMBER(8,2),
  commission_pct NUMBER(2,2),
  hire_date      DATE
                CONSTRAINT emp_hire_date_nn
                NOT NULL,
  ...
```

**System
named**


**User
named**



The UNIQUE Constraint


A **UNIQUE key integrity constraint** requires that every value in a column or set of columns (key) be unique—that is, no two rows of a table can have duplicate values in a specified column or set of columns.

EMPLOYEES



EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

...



208	Smith	JSMITH
209	Smith	JSMITH


Allowed

**Not allowed:
already exists**

Defined at either the table level or the column level

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    CONSTRAINT emp_email_uk UNIQUE(email));
```


The PRIMARY KEY Constraint

The PRIMARY KEY constraint is a column or set of columns that uniquely identifies each row in a table.

DEPARTMENT  **PRIMARY KEY**

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...
Not allowed
(Null value)



 **INSERT INTO**

	Public Accounting		1400
50	Finance	124	1500

Not allowed
(50 already exists)



Defined at either the table level or the column level:

```
CREATE TABLE departments(
  department_id      NUMBER(4),
  department_name     VARCHAR2(30)
    CONSTRAINT dept_name_nn NOT NULL,
  manager_id         NUMBER(6),
  location_id         NUMBER(4),
  CONSTRAINT dept_id_pk PRIMARY KEY(department_id));
```

The FOREIGN KEY Constraint

- The **FOREIGN KEY**, or **referential integrity constraint** designates a column or combination of columns as a foreign key and establishes a relationship between a primary key or a unique key in the same table or a different table.
- A foreign key value must match an existing value in the parent table or be NULL.
- Foreign keys are based on data values and are purely logical, not physical, pointers.

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

**PRIMARY
KEY**

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

**FOREIGN
KEY**

...

200	Ford	9
201	Ford	60

INSERT INTO

Not allowed
(9 does not
exist)

Allowed

Defined either at the table level or the column level

```
CREATE TABLE employees(
    employee_id    NUMBER(6),
    last_name      VARCHAR2(25) NOT NULL,
    email          VARCHAR2(25),
    salary         NUMBER(8,2),
    commission_pct NUMBER(2,2),
    hire_date      DATE NOT NULL,
    ...
    department_id  NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
        REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```



FOREIGN KEY Constraint Keywords

- **FOREIGN KEY:** Defines the column in the child table at the table constraint level
- **REFERENCES:** Identifies the table and column in the parent table
- **ON DELETE CASCADE:** Deletes the dependent rows in the child table when a row in the parent table is deleted
- **ON DELETE SET NULL:** Converts dependent foreign key values to null
- **ON DELETE RESTRICT:** Prevents deletion if there are child records (this is the default)

The CHECK Constraint

- Defines a condition that each row must satisfy. Can be of two types:
 1. Column check constraint (check Salary > 0)
 2. Table check constraint (check Last_Name IS NOT NULL OR Initial IS NOT NULL)

```
..., salary    NUMBER(2)
      CONSTRAINT emp_salary_min
      CHECK (salary > 0),...
```

- A single column can have multiple CHECK constraints which refer to the column in its definition. There is no limit on the number of CHECK constraints which you can define on a column.
- CHECK constraints can be defined at the column level or table level.



Viewing Constraints

Query the USER_CONSTRAINTS table to view all constraint definitions and names.

```
SELECT    constraint_name, constraint_type,
          search_condition
FROM      user_constraints
WHERE     table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	C	SEARCH_CONDITION
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL
EMP_SALARY_MIN	C	salary > 0
EMP_EMAIL_UK	U	

...

Data Manipulation Language (DML)

A DML statement is executed when you:

- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

A *transaction* consists of a collection of DML statements that form a logical unit of work.

Adding a New Row to a Table

New row

70	Public Relations	100	1700
----	------------------	-----	------

...insert a new row into the
DEPARTMENTS table...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
70	Public Relations	100	1700

Add new rows to a table by using the INSERT statement.



The INSERT Statement Syntax

```
INSERT INTO  table [(column [, column...])]
VALUES      (value [, value...]);
```

Only one row is inserted at a time with this syntax.

Inserting new rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the INSERT clause.
- Enclose character and date values within single quotation marks.

Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
INSERT INTO  departments (department_id,
                           department_name)
VALUES      (30, 'Purchasing');
1 row created.
```

- Explicit method: Specify the NULL keyword in the VALUES clause.

```
INSERT INTO  departments
VALUES      (100, 'Finance', NULL, NULL);
1 row created.
```

Inserting Special Values

- You can use functions to enter special values in your table.
- The SYSDATE function records the current date and time.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       hire_date)
VALUES      (113,
             'Louis', 'Popp',
             SYSDATE);
1 row created.
```



Inserting Specific Date Values

- Add a new employee.

```
INSERT INTO employees
VALUES      (114,
            'Den', 'Raphealy',
            'DRAPHEAL', '515.127.4561',
            TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
            'AC_ACCOUNT', 11000, NULL, 100, 30);
```

1 row created.

- Verify your addition.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

Copying Rows from Another Table

- Write the INSERT statement with a subquery.

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM   employees
WHERE  job_id LIKE '%REP%';
```

4 rows created.

- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.



Changing Data in a Table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Update rows in the **EMPLOYEES** table.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

The UPDATE Statement Syntax

- Modify existing rows with the UPDATE statement.

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Update more than one row at a time, if required.

Updating Rows in a Table

- Specific row or rows are modified if you specify the WHERE clause.

```
UPDATE employees
SET     department_id = 70
WHERE   employee_id = 113;
1 row updated.
```

- All rows in the table are modified if you omit the WHERE clause.

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated.
```



Updating Two Columns with a Subquery

- Update employee 114's job and salary to match that of employee 205.

```
UPDATE employees
SET   job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 205),
      salary = (SELECT salary
                FROM   employees
                WHERE  employee_id = 205)
WHERE employee_id = 114;
1 row updated.
```

Updating Rows Based on Another Table

- Use subqueries in UPDATE statements to update rows in a table based on values from another table.

```
UPDATE copy_emp
SET   department_id = (SELECT department_id
                      FROM   employees
                      WHERE  employee_id = 100)
WHERE job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 200);
1 row updated.
```

Updating Rows: Integrity Constraint Error

```
UPDATE employees
SET   department_id = 55
WHERE department_id = 110;
```

```
UPDATE employees
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

Department number 55 does not exist



Removing a Row from a Table

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

Delete a row from the DEPARTMENTS table.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM]    table
[WHERE          condition];
```

Deleting Rows from a Table

- Specific rows are deleted if you specify the WHERE clause.

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
DELETE FROM copy_emp;
22 rows deleted.
```



Deleting Rows Based on Another Table

- Use subqueries in DELETE statements to remove rows from a table based on values from another table.

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name LIKE '%Public%');

1 row deleted.
```

Deleting Rows: Integrity Constraint Error

```
DELETE FROM departments
WHERE department_id = 60;
```

```
DELETE FROM departments
      *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

You cannot delete a row that contains a primary key that is used as a foreign key in another table.

Database Transactions

A database transaction consists one of the following:

- DML statements which constitute one consistent change to the data
- One DDL statement
- Begins when the first DML SQL statement is executed.
- Ends with one of the following events:
 - A COMMIT or ROLLBACK statement is issued
 - A DDL statement executes (automatic commit)
 - The system crashes



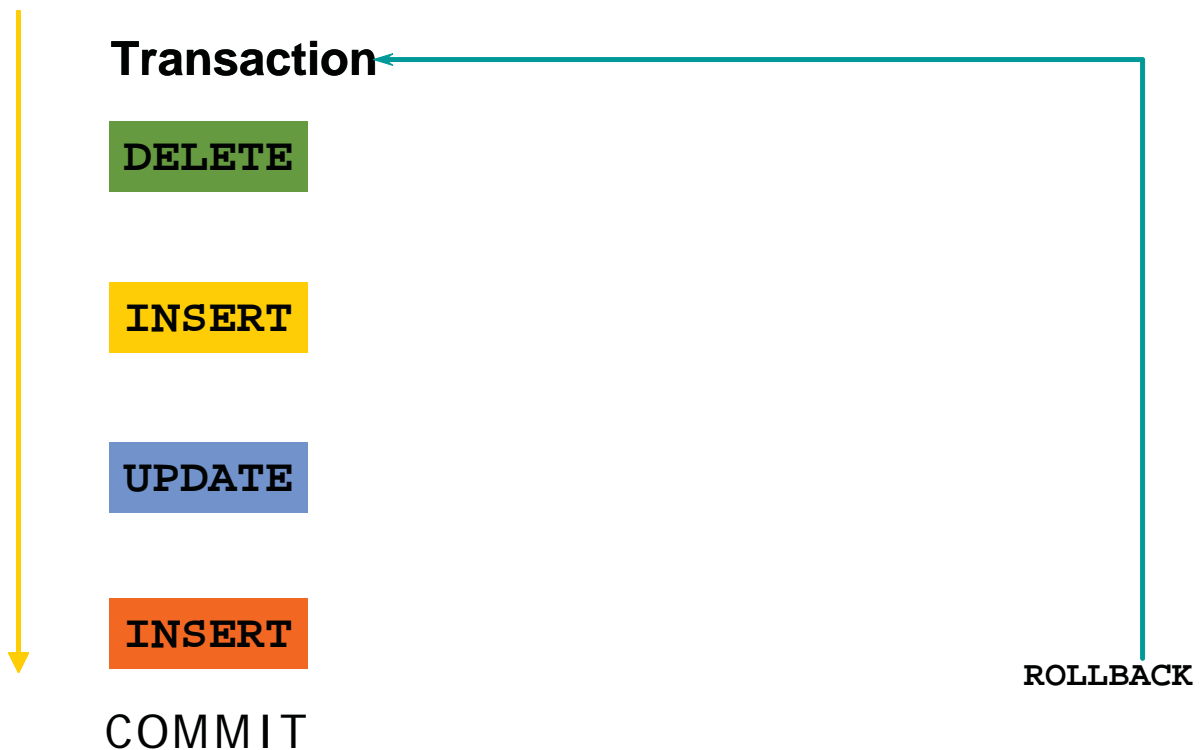
Advantages of COMMIT and ROLLBACK Statements

With COMMIT and ROLLBACK statements, you can:

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

Controlling Transactions

Time



Implicit Transaction Processing

- An automatic commit occurs under the following circumstances:
 - DDL statement is issued
 - Normal exit from iSQL*Plus, without explicitly issuing COMMIT or ROLLBACK statements
- An automatic rollback occurs under an abnormal termination of SQL*Plus or a system failure.



State of the Data Before *COMMIT* or *ROLLBACK*

- The previous state of the data can be recovered.
- The current user can review the results of the DML operations by using the `SELECT` statement.
- Other users *cannot* view the results of the DML statements by the current user.
- The affected rows are *locked*; other users cannot change the data within the affected rows.

State of the Data after *COMMIT*

- Data changes are made permanent in the database.
- The previous state of the data is permanently lost.
- All users can view the results.
- Locks on the affected rows are released. Those rows are available for other users to manipulate.

Committing Data

- Make the changes.

```
DELETE FROM employees
WHERE employee_id = 99999;
1 row deleted.

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 row inserted.
```

- Commit the changes

```
COMMIT;
Commit complete.
```



State of the Data After ROLLBACK

- Discard all pending changes by using the ROLLBACK statement
- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released.

```
DELETE FROM copy_emp;
22 rows deleted.
ROLLBACK;
Rollback complete.
```

Summary

Statement	Description
INSERT	Adds a new row to the table
UPDATE	Modifies existing rows in the table
DELETE	Removes existing rows from the table
MERGE	Conditionally inserts or updates data in a table
COMMIT	Makes all pending changes permanent
SAVEPOINT	Is used to rollback to the savepoint marker
ROLLBACK	Discards all pending data changes
CREATE TABLE	Creates a table
ALTER TABLE	Modifies table structures
DROP TABLE	Removes the rows and table structure
RENAME	Changes the name of a table, view, sequence, or synonym
TRUNCATE	Removes all rows from a table and releases the storage space
COMMENT	Adds comments to a table or view





TATA CONSULTANCY SERVICES

www.tcs.com