Course code : **CSE2007**
Course title : **Database Management System**
Module : **3**
Topic : **3**

Structured Query Language

Objectives

This session will give the knowledge about

- Structured Query Language
- Types of Queries
- Types of clauses

What is SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like

MySQL,	MS Access,	Oracle,	Sybase,
Informix,	Postgres,	SQL Server	

use SQL as their standard database language.

Applications of SQL

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

History of SQL

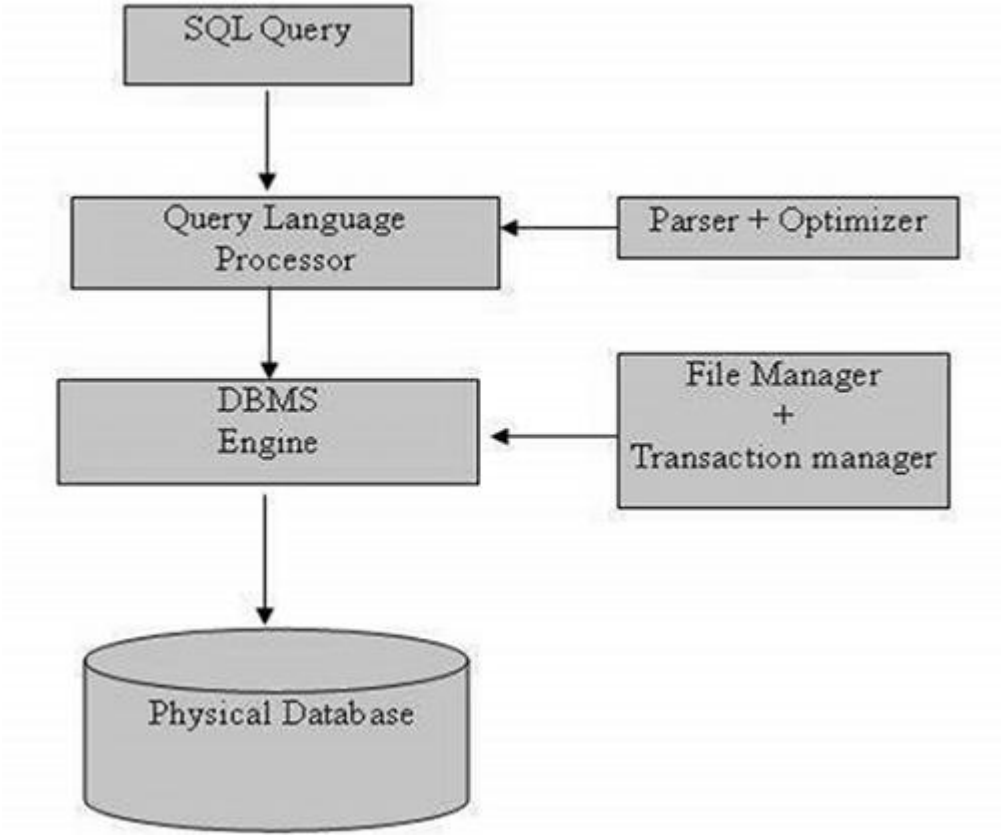
- 1970 − Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases. He described a relational model for databases.
- 1974 − Structured Query Language appeared.
- 1978 − IBM worked to develop Codd's ideas and released a product named System/R.
- 1986 − IBM developed the first prototype of relational database and standardized by ANSI.
- The first relational database was released by Relational Software which later came to be known as Oracle.

SQL Components

When executing an SQL command for any RDBMS, the system determines the best way to carry out the request and SQL engine figures out how to interpret the task.

There are various components included in this process as follows.

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.



SQL Commands

DDL - Data Definition Language

S.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in the database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other objects in the database.

SQL Commands

DML - Data Manipulation Language

S.No.	Command & Description
1	**SELECT** - Retrieves certain records FROM one or more tables.
2	**INSERT** - Creates a record.
3	**UPDATE** - Modifies records.
4	**DELETE** - Deletes records.

SQL Commands

DCL - Data Control Language

S.No.	Command & Description
1	**GRANT** - Gives a privilege to user.
2	**REVOKE** - Takes back privileges granted FROM user.

TCL - Transaction Control Language

S.No.	Command & Description
1	**COMMIT** - to save the changes.
2	**ROLLBACK** - to roll back the changes.
3	**SAVEPOINT** - creates points within the groups of transactions in which to ROLLBACK

SQL Constraints

NOT NULL Constraint − Ensures that a column cannot have a NULL value.

```
CREATE TABLE customers(
        id  INT NOT NULL,
        name VARCHAR (20) NOT NULL,
        age  INT NOT NULL,
        address  CHAR (25) ,
        salary DECIMAL (18, 2),
        PRIMARY KEY (id)
);
```

```
ALTER TABLE customers MODIFY
salary  DECIMAL (18, 2) NOT NULL;
```

SQL Constraints

DEFAULT Constraint − Provides a default value for a column when none is specified.

```
CREATE TABLE customers(
   id INT NOT NULL,
   name VARCHAR (20) NOT NULL,
   age  INT  NOT NULL,
   address CHAR (25) ,
   salary DECIMAL (18, 2)
   DEFAULT 5000.00,
   PRIMARY KEY (id)
);
```

```
ALTER TABLE customers MODIFY
salary  DECIMAL (18, 2) DEFAULT
5000.00;
```

```
ALTER TABLE customers MODIFY
age DEFAULT NULL;
```

SQL Constraints

UNIQUE Constraint − Ensures that all the values in a column are different.

```
CREATE TABLE customers(
   id INT NOT NULL,
   name VARCHAR (20) NOT NULL,
   age  INT  NOT NULL UNIQUE,
   address CHAR (25) ,
   salary DECIMAL (18, 2)
   DEFAULT 5000.00,
   PRIMARY KEY (id)
);
```

```
ALTER TABLE customers
   MODIFY age INT UNIQUE;
```

```
ALTER TABLE customers ADD UNIQUE (age);
```

```
ALTER TABLE customers
   ADD CONSTRAINT myUniqueConstraint
UNIQUE (age, salary);
```

```
ALTER TABLE customers
   DROP CONSTRAINT myUniqueConstraint;
```

SQL Constraints

PRIMARY Key − Uniquely identifies each row/record in a database table.

```
CREATE TABLE customers(
   id INT NOT NULL,
   name VARCHAR (20) NOT NULL,
   age INT NOT NULL,
   address CHAR (25) ,
   salary DECIMAL (18, 2),
   PRIMARY KEY (id)
);
```

```
ALTER TABLE customers ADD
PRIMARY KEY (id);
```

```
ALTER TABLE customers DROP
PRIMARY KEY;
```

SQL Constraints

FOREIGN Key − Uniquely identifies a row/record in any another database table.

```
CREATE TABLE customers (
   id INT NOT NULL,
   name VARCHAR(20) NOT NULL,
   age INT NOT NULL,
   address CHAR(25) ,
   salary DECIMAL(18, 2),
   PRIMARY KEY (id)
);
```

```
ALTER TABLE orders ADD CONSTRAINT mycons
FOREIGN KEY (id) REFERENCES customers;
```

```
CREATE TABLE orders (
   ord_id INT NOT NULL,
   ord_date DATE,
   id INT REFERENCES customers,
   amount DECIMAL(18, 2),
   PRIMARY KEY (ord_id)
);
```

```
ALTER TABLE orders
   DROP CONSTRAINT mycons;
```

SQL Constraints

CHECK Constraint – The CHECK constraint ensures that all values in a column satisfy certain conditions.

```
CREATE TABLE customers(
   id INT NOT NULL,
   name VARCHAR (20) NOT NULL,
   age  INT  NOT NULL CHECK (age>=18),
   address CHAR (25) ,
   salary DECIMAL (18, 2)
   DEFAULT 5000.00,
   PRIMARY KEY (id)
);
```

```
ALTER TABLE customers
   MODIFY age INT CHECK (age>=18);
```

```
ALTER TABLE customers ADD CONSTRAINT
myUniqueConstraint CHECK (age>=18);
```

```
ALTER TABLE customers  DROP CONSTRAINT
myUniqueConstraint;
```

SQL Constraints

INDEX – Used to create and retrieve data FROM the database very quickly.

```
CREATE TABLE customers(
   id INT NOT NULL,
   name VARCHAR (20) NOT NULL,
   age INT NOT NULL,
   address CHAR (25) ,
   salary DECIMAL (18, 2),
   PRIMARY KEY (id)
);
```

```
CREATE INDEX idx_age
   ON customers (age);
```

```
DROP INDEX idx_age;
```

Data Integrity

The following categories of data integrity exist with each RDBMS

- **Entity Integrity** − There are no duplicate rows in a table.
- **Domain Integrity** − Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential integrity** − Rows cannot be deleted, which are used by other records.
- **User-Defined Integrity** − Enforces some specific business rules that do not fall into entity, domain or referential integrity.

Oracle Datatypes

Data Type	Syntax	Description	Range	Valid Data
Character	char (length) ex: char(10)	Fixed length character	1 to 2000 bytes	‘1234567890’ ‘dfee’
Character	varchar2(length) ex: varchar2(5)	Variable length Character string	1 to 4000 Bytes	‘asqeq’ ‘1234’ ‘ssf%.sd’
Number	number	Integer of any	Integer range	Any number
	number(3) number (4,1)	maximum range Only 3 digits Float of max 1 decimal place	38 digits after decimal -84 to 127	123, 789 123.4,111.5 12.4

Oracle Datatypes

Data Type	Syntax	Description	Range	Valid Data
Date	date	Fixed length date -7 bytes for each date, month	Jan 1 ,4712 BC to Dec 31, 4712 AD	’01-jan-01’ ’31-feb-2005’
Time	Timestamp	Date with time (No separate time type)		’24-sep-75,06:12:12’
Long	Long	To store Variable character length (one table only one long type)	Max 2 GB	‘ggfg….’
Raw	Raw	Binary data or byte strings (manipulation of data cannot be done)	Max 2000 bytes	

Oracle Datatypes

Data Type	Syntax	Description	Range	Valid Data
Long Raw	Long raw	Binary data of Variable length	Max 2GB	
Large Object	CLOB	Stores character Object with single byte Character	Max 4 GB	BFILE('dir. Name', 'filename')
	BLOB	Stores large binary objects(Graphics, video clips and sound files)		
	BFILE	Stores file pointers to LOBs managed by file systems external to the Db.		

CHAR vs VARCHAR

```
CREATE TABLE test (
name1    CHAR(10),
name2 VARCHAR2(10));

SELECT LENGTH(name1),
LENGTH(name2) FROM test;
```

```
LENGTH(name1) LENGTH(name2)
------------- -------------
           10             1
```

```
'ABC'ASVARCHAR2(10)  'ABC'ASCHAR(10)
-------------------  ----------------
"abc"                "abc       "
```

CREATE TABLE

SYNTAX

CREATE TABLE < tablename1> (
<column name 1> < datatype>,
<column name 2> < datatype>);

EXAMPLE

CREATE TABLE employee (
eid VARCHAR2(5),
ename VARCHAR2(20));

CREATE TABLE with CONSTRAINT

SYNTAX

CREATE TABLE < tablename1> (
<column name 1> < datatype>,
<column name 2> < datatype>,
primary key (<column name1>));

EXAMPLE

```
CREATE TABLE employee (
eid VARCHAR2(5),
ename VARCHAR2(20),
PRIMARY KEY (eid) );

CREATE TABLE department (
did VARCHAR2(5),
dname VARCHAR2(20),
PRIMARY KEY (did) );
```

CREATE TABLE with CONSTRAINT NAME

SYNTAX

CREATE TABLE < tablename1> (
<column name 1> < datatype>,
<column name 2> < datatype>,
constraint < constraint name1 >
primary key (<column name1>),
constraint <constraint name2> foreign key (<column name2>) references <tablename2> (<column name1>));

EXAMPLE

CREATE TABLE employee (
eid VARCHAR2(5),
ename VARCHAR2(20),
dept_id varchar(2),
CONSTRAINT pk_const PRIMARY KEY (eid),
CONSTRAINT fk_const FOREIGN KEY (dept_id) REFERENCES department(did));

CREATE TABLE with CHECK CONSTRAINT

SYNTAX

CREATE TABLE < tablename> (
<column name1 > < datatype> ,
<column name 2> < datatype>,
check (< column name 1 > in (values)
)
check (< column name 2 > between
<val1> and <val2>)

EXAMPLE

```
CREATE TABLE employee(
eid VARCHAR2(5),
ename VARCHAR2(20),
age NUMBER(2),grade NUMBER(1),
CONSTRAINT check1 CHECK (age
BETWEEN 18 AND 24),
CONSTRAINT check2 CHECK (grade
IN (1,2,3)));
```

CREATE SEQUENCE

SYNTAX

```
CREATE SEQUENCE
sequence_name
  MINVALUE value
  MAXVALUE value
  START WITH value
  INCREMENT BY value
  CACHE value;
```

EXAMPLE

```
CREATE SEQUENCE supplier_seq
  MINVALUE 1
  MAXVALUE 99
  START WITH 1
  INCREMENT BY 1
  CACHE 20;

SELECT supplier_seq.NEXTVAL FROM
DUAL;

DROP SEQUENCE sequence_name;
```

CREATE ROLE

SYNTAX

for grouping together a set of access rights

CREATE ROLE <role name>
IDENTIFIED BY <password>

EXAMPLE

CREATE ROLE 17bcd2212
IDENTIFIED BY 17bcd2212;

GRANT select, insert, update, delete ON employee TO 17bcd2212;

GRANT all ON employee TO 17bcd2212;

ALTER STATEMENT

ALTER TABLE table_name

- Add column defintion
- Drop column_name
- **Add**
  - Primary key definition
  - Foriegn key definition
  - Unique constraints
  - Check constraints
- **Drop** constraints column_name

ALTER STATEMENT

To add a new column definition to an existing table

EXAMPLE

ALTER TABLE employee ADD (dept_id number(2), salary number(8,2),ph_no number(10));

Drop a column FROM an existing table

EXAMPLE: ALTER TABLE employee DROP (grade);

Add or drop primary key to/FROM an existing table

EXAMPLE

Add: ALTER TABLE customer_details ADD PRIMARY KEY(account_no);

Drop: ALTER TABLE customer_details DROP PRIMARY KEY;

ALTER STATEMENT

Add or drop foreign key to/FROM an existing table

EXAMPLE

Add: ALTER TABLE employee
ADD FOREIGN KEY(dept_id)
REFERENCES department (did);

Drop: ALTER TABLE employee DROP fk_const;

Add unique constraint to an existing table

EXAMPLE

ALTER TABLE employee UNIQUE (ph_no);

ALTER STATEMENT

Add check constraint to an existing table

EXAMPLE

```
ALTER TABLE employee ADD
CHECK (salary BETWEEN 1000 AND 10000);
```

Add NOT NULL constraints to an existing table

EXAMPLE

```
ALTER TABLE employee MODIFY
ph_no char(10) NOT NULL;
```

TRUNCATE STATEMENT

Truncate table statement is used to remove all rows FROM the table.

When truncate table statement is used ,all the contents of the specified table are lost but its definition remains intact.

It releases the memory occupied by the contents of the specified table.

SYNTAX

```
TRUNCATE TABLE table_name;
```

EXAMPLE

```
TRUNCATE TABLE employee;
```

DROP STATEMENT

Drop table statement is used to drop or remove a table permanently FROM the database.

Both the schema/structure of the table and all its contents are lost when drop table command is used.

There is no way to recover the data.

SYNTAX

```
DROP TABLE table_name;
```

EXAMPLE

```
DROP TABLE employee;
```

INSERT STATEMENT

SYNTAX

INSERT INTO <table name> VALUES (a list of data values);

INSERT INTO <table name(column names)> VALUES (list of values);

EXAMPLE

INSERT INTO employee (eid,ename)values(‘18bd1','dhivya');

INSERT INTO employee VALUES('&eid','&ename',&age,&grade);

INSERT INTO employee VALUES(’18ec2’,'harini',21,1);

SELECT STATEMENT

PURPOSE of SELECT COMMAND

- SELECT --→ is a list of one or more columns.
- * --→ SELECTs all columns
- DISTINCT --→ suppresses duplicates
- Column expression --→ Expression SELECTs the named column or
- Alias --→ gives the SELECTed columns of different headings
- From table --→ specifies the table containing the column

SELECT STATEMENT

```
SELECT * FROM employee;
SELECT eid,ename FROM employee;
SELECT eid,ename AS employee FROM employee;
SELECT ename || dept_id AS name FROM employee;
SELECT distinct dept_id FROM employee;
SELECT ename,salary,salary+300 AS updated_salary FROM employee;
SELECT ename,salary,12*(salary+100) AS updated FROM employee;
SELECT * FROM employee WHERE salary<=5000;
```

DELETE STATEMENT

PURPOSE of DELETE COMMAND

- Delete s one or more records
- Delete cannot delete column(s) from a table.
- It deletes only rows
- To delete a column from a table , ALTER table statement must be used

Delete statement

SYNTAX

- DELETE FROM table_name [where condition] ;

DELETE STATEMENT

Deleting all rows of the table

```
DELETE FROM customer_details ;
```

Deleting some rows of a table

```
DELETE FROM customer_details  WHERE Cust_id=102;
```

Invalid Delete Statement

```
DELETE * FROM customer_details; (OR)
DELETE Cust_Id  FROM customer_details;
```

Difference between Truncate and Delete

TRUNCATE	DELETE
Truncate is a DDL statement	Delete is a DML statement
Truncate deletes all records	Delete is used to selectively delete some record from the table
Truncate releases the memory occupied by the records of the table	Delete statement deletes the specific rows and its content only
Data removed using truncate cannot be recovered.	Data removed using delete could be recovered(using Rollback)

UPDATE STATEMENT

PURPOSE of UPDATE COMMAND

Update modifies the values of one or more columns in selected rows of a table.

SYNTAX

UPDATE <table_name> SET <field> = <value> WHERE <condition>

- Target table to be updated is named
- ‘WHERE’ clause selects the rows of the table to be modified.
- ‘SET’ clause specifies which column are to be updated and calculates the new values for them.

UPDATE STATEMENT

Changing all rows

```
UPDATE employee SET hra=NULL ;
```

Changing value for more than one column

```
UPDATE employee SET salary=9000,hra=900 WHERE eid=102;
```

Changing some rows

```
UPDATE employee  SET  hra=1000 WHERE salary>3000;
```

Data Control Language - DCL

- The Data Control Language provides users with privilege commands.
- Grant privileges can be granted to others using SQL command **GRANT**.
- To withdraw the privileges that has been granted to user, we use the **REVOKE** command

SYNTAX:

Commit:

```
COMMIT employee;
COMMIT;
```

Data Control Language - DCL

Savepoint:

SAVEPOINT savepoint_id

Rollback:

ROLLBACK employee;

ROLLBACK;

ROLLBACK TO SAVEPOINT savepoint_id;

Grant privilege:

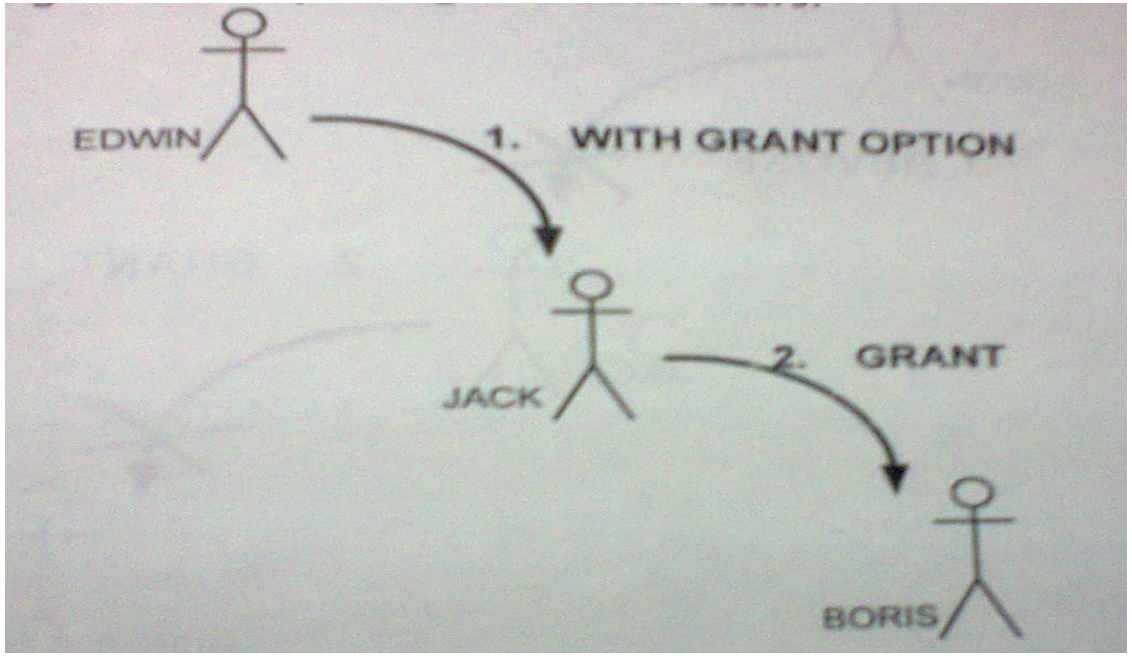
GRANT PRIVILEGES ON <object-name> TO <username>

Revoke privilege:

REVOKE PRIVILEGES ON <object-name> FROM <username>

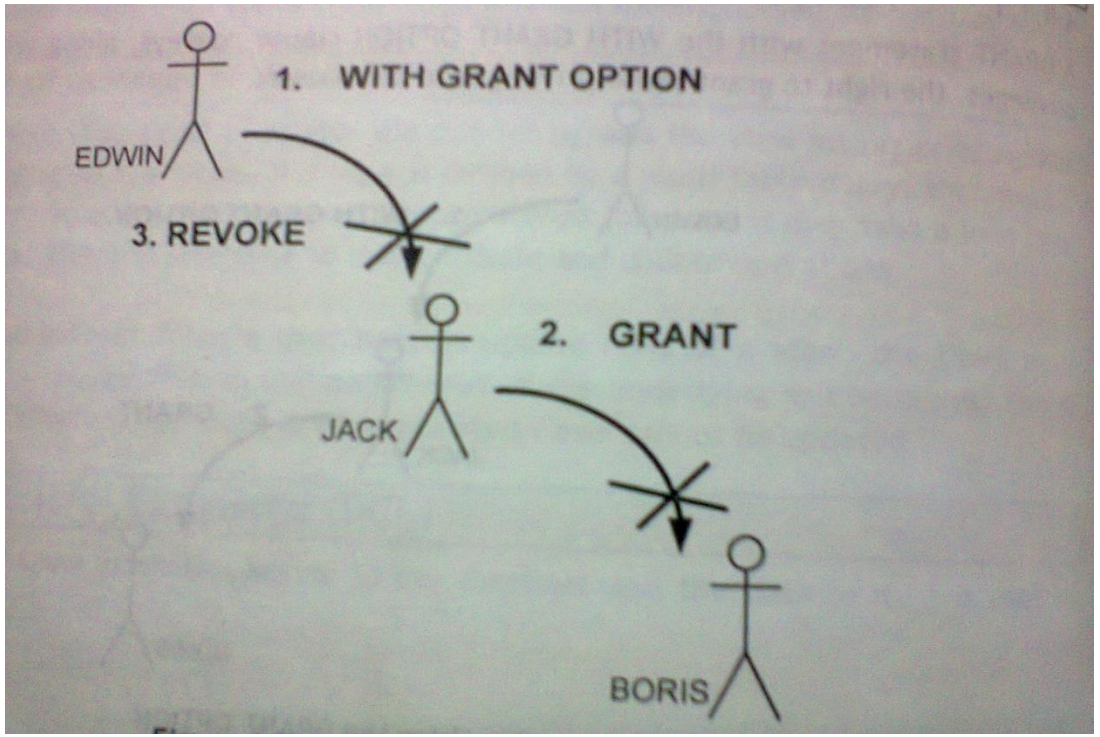
GRANT Privileges

- The Grant statement is used to grant security privileges on database objects to specific users.
- Grant Statement is used by the owner of the table to give other user access to data



Revoke Privileges

- The revoke statement is used to REVOKE privileges previously granted with GRANT statement



Data Control Language - DCL

UPDATE employee SET eid=‘17s23 WHERE name=’reddy’;

SAVEPOINT s1;

DELETE FROM employee WHERE name= ’reddy’;

SAVEPOINT s2;

ROLLBACK to SAVEPOINT s1;

ROLLBACK;

GRANT select,update ON employee TO 17bcd2212;

REVOKE select,update ON employee FROM 17bcd2212;

Summary

This session will give the knowledge about

- Structured Query Language
- Types of Queries
- Types of clauses