SQL Commands

DDL
CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME

DML
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE

DCL
GRANT
REVOKE

TCL
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

- Data Definition Language
- Data Manipulation Language
- Data Control Language
- Transaction Control Language

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
ALTER - alters the structure of the existing database
DROP - delete objects from the database
TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
COMMENT - add comments to the data dictionary
RENAME - rename an object

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

SELECT - retrieve data from a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - Delete all records from a database table
MERGE - UPSERT operation (insert or update)
CALL - call a PL/SQL or Java subprogram
EXPLAIN PLAN - interpretation of the data access path
LOCK TABLE - concurrency Control

DCL

DCL is short name of Data Control Language which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

GRANT - allow users access privileges to the database
REVOKE - withdraw users access privileges given by using the GRANT command

TCL

TCL is short name of Transaction Control Language which deals with a transaction within a database.

COMMIT - commits a Transaction
ROLLBACK - rollback a transaction in case of any error occurs
SAVEPOINT - to rollback the transaction making points within groups
SET TRANSACTION - specify characteristics of the transaction

There are different types of SQL expressions, which are mentioned below −

- Boolean Expressions
- Numeric Expressions(count ,sum)
- Date Expressions(date,smalldate)

RDBMS stands for <u>R</u>elational <u>D</u>atabase <u>M</u>anagement <u>S</u>ystem. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

What is a table?

The data in an RDBMS is stored in database objects which are called as tables.

What is a field?

Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.

What is a Record or a Row?

A record is also called as a row of data is each individual entry that exists in a table.

What is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

- SQL CREATE DATABASE Statement

CREATE DATABASE database_name;

- SQL DROP DATABASE Statement

DROP DATABASE database_name;

- SQL USE Statement

USE database_name;

- SQL TRANSACTION Statement

  Begin transaction;

- SQL COMMIT Statement

COMMIT;

- SQL ROLLBACK Statement

 ROLLBACK;

TABLE

- SQL CREATE TABLE Statement

CREATE TABLE table_name(column1 datatype PRIMARY KEY(one or more columns) default value);

- SQL DROP TABLE Statement

DROP TABLE table_name;

- SQL CREATE INDEX Statement

CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...columnN);

- SQL DROP INDEX Statement

ALTER TABLE table_name DROP INDEX index_name;

- SQL SELECT Statement

SELECT column1 FROM   table_name;

- SQL DISTINCT Clause

SELECT DISTINCT column1 FROM   table_name;

- SQL WHERE Clause

SELECT column1 FROM   table_name WHERE CONDITION;

- SQL AND/OR Clause

SELECT column1 FROM   table_name WHERE CONDITION-1 {AND|OR}CONDITION-2;

- SQL IN Clause

SELECT column1 FROM   table_name WHERE column_name NOT IN (val-1, val-2,...val-N);

- SQL HAVING Clause

SELECT SUM(column_name) FROM   table_name WHERE CONDITION GROUP BY column_name HAVING (arithmetic function condition);

- SQL BETWEEN Clause

SELECT column1 FROM   table_name WHERE column_name BETWEEN val-1 AND val-2;

- SQL LIKE Clause

SELECT column1, FROM   table_name WHERE column_name LIKE {_ap% PATTERN% };

- SQL ORDER BY Clause

SELECT column1,FROM   table_name WHERE CONDITION ORDER BY column_name {ASC|DESC};

- SQL GROUP BY Clause

SELECT SUM(column_name) FROM   table_name WHERE CONDITION GROUP BY column_name;

- SQL COUNT Clause

SELECT COUNT(column_name) FROM   table_name WHERE CONDITION;

- SQL DESC Statement

DESC table_name;

- SQL TRUNCATE TABLE Statement

TRUNCATE TABLE table_name;

- SQL ALTER TABLE Statement

ALTER TABLE table_name {ADD|DROP|MODIFY} column_name {data_ype};

- SQL ALTER TABLE Statement (Rename)

ALTER TABLE table_name RENAME TO new_table_name;

- SQL UPDATE Statement

UPDATE table_name SET column1 = value1, [WHERE CONDITION];

SQL DELETE Statement

DELETE FROM table_name WHERE {CONDITION};

- SQL TOP Statement

SELECT TOP 3 * FROM CUSTOMERS;

- SQL TRUNCATE TABLE Statement

SELECT * FROM CUSTOMERS LIMIT 3;

- SQL ROWNUM Statement

SELECT * FROM CUSTOMERS WHERE ROWNUM <= 3;

- SQL CASE Statement (Rename)

SELECT * FROM CUSTOMERS

ORDER BY (CASE ADDRESS

WHEN 'DELHI'   THEN 1

WHEN 'BHOPAL' THEN 2

WHEN 'KOTA'   THEN 3

WHEN 'AHMEDABAD' THEN 4

  WHEN 'MP' THEN 5
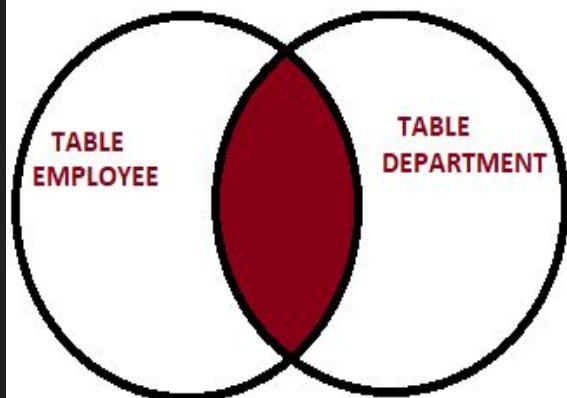
  ELSE 100 END) ASC, ADDRESS DESC;

constraints

- NOT NULL Constraint − Ensures that a column cannot have NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all values in a column are different.(null allowed,ph.no)
- PRIMARY Key − Uniquely identifies each row/record in a database table.(no null,duplicate)
- FOREIGN Key − Uniquely identifies a row/record in any of the given database table(join id is forgien key).
- CHECK Constraint − The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- INDEX − Used to create and retrieve data from the database very quickly.

ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;

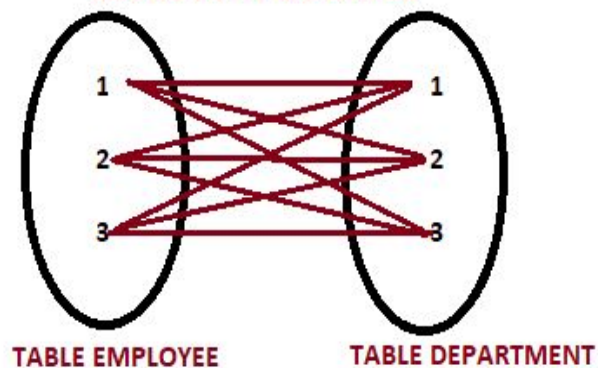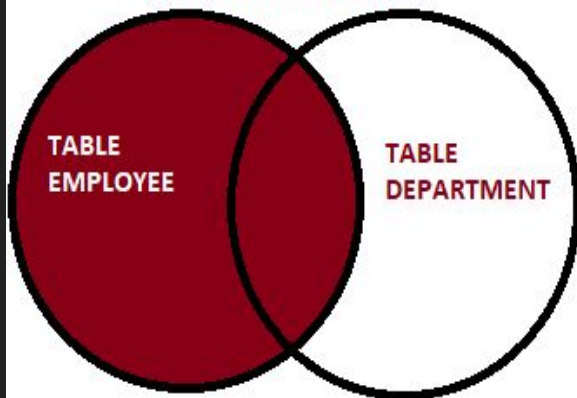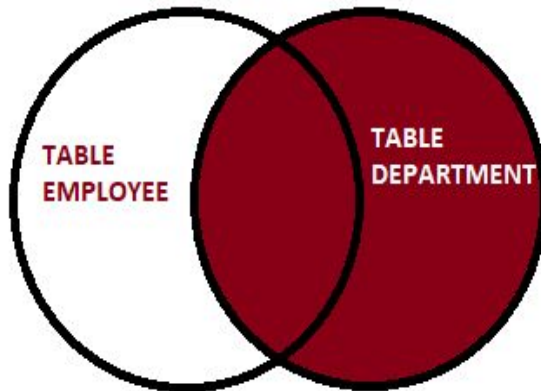# INNER JOIN EXAMPLE

TABLE EMPLOYEE

TABLE DEPARTMENT

FULL JOIN EXAMPLE

TABLE EMPLOYEE

TABLE DEPARTMENT

©tutorialgateway.org

CROSS JOIN EXAMPLE

1   1
2   2
3   3

TABLE EMPLOYEE   TABLE DEPARTMENT

LEFT JOIN EXAMPLE

TABLE EMPLOYEE

TABLE DEPARTMENT

RIGHT JOIN EXAMPLE

TABLE EMPLOYEE

TABLE DEPARTMENT

SELF JOIN EXAMPLE

CONNECTING TO ITESELF

TABLE EMPLOYEE
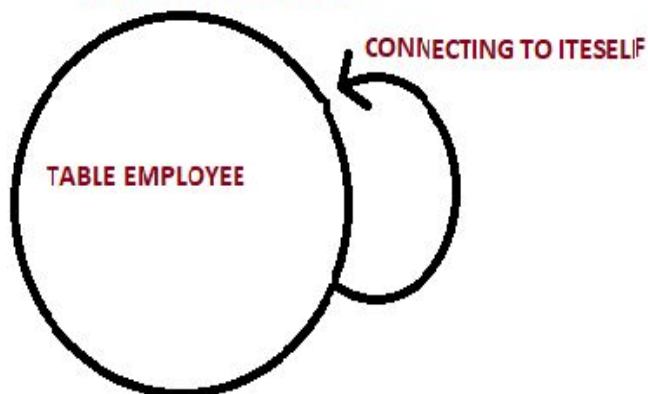
joints

- INNER JOIN − returns rows when there is a match in both tables.
- LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN − returns rows when there is a match in one of the tables.
- SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;

```sql
    select hobbies.name,persons.name from persons
    join hobbies
    on persons.id=hobbies.person_id
    where persons.name="Bobby McBobbyFace"
select customers.name,customers.email,orders.item,orders.price
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id ;
select customers.name,customers.email,sum(orders.price) as total_amount
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id  GROUP by customers.name order by total_amount desc;

select movies.title,sequel.title
FROM movies
join movies sequel
on movies.sequel_id=sequel.id;

 select persons.fullname,a.fullname from friends
  join persons
  on friends.person1_id=persons.id
  join persons a
  on  friends.person2_id=a.id;
```

There are two other clauses (i.e., operators), which are like the UNION clause

- SQL INTERSECT Clause − This is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.
- SQL EXCEPT Clause − This combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.
- The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.
- The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.
- The EXISTS operator is used to test for the existence of any record in a subquery.

SELECT column1 [, column2]FROM table1 [, table2] [WHERE condition]
UNION ALL
SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition]

SELECT *column_name(s)*
FROM *table_name*
WHERE EXISTS
(SELECT *column_name* FROM *table_name* WHERE *condition*);

The SQL NULL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank

SQL> CREATE TABLE CUSTOMERS(
  ID INT   NOT NULL,
  NAME VARCHAR (20)   NOT NULL,(primary key does not accept null,unique allows)
  AGE INT   NOT NULL,
  ADDRESS CHAR (25) ,
  SALARY DECIMAL (18, 2),
  PRIMARY KEY (ID)
);
SELECT ID, NAME, AGE, ADDRESS, SALARY
  FROM CUSTOMERS
  WHERE SALARY IS NOT NULL;

ANY AND ALL

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE
Quantity = 10);
```

ALIAS

You can rename a table or a column temporarily by giving another name known as Alias.

The basic syntax of a table alias is as follows.

SELECT column1, column2....
FROM table_name AS alias_name
WHERE [condition];

The basic syntax of a column alias is as follows.

SELECT column_name AS alias_name
FROM table_name
WHERE [condition];

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

but it slows down data input, with the UPDATE and the INSERT statements.

CREATE INDEX
CREATE INDEX index_name ON table_name;

Single-Column Indexes

CREATE INDEX index_name ON table_name (column_name);

Unique Indexes

CREATE UNIQUE INDEX index_name on table_name (column_name);

Composite Indexes

CREATE INDEX index_name on table_name (column1, column2);

DROP INDEX

DROP INDEX index_name;

add a New Column in an existing table is as follows.

    ALTER TABLE table_name ADD column_name datatype;

DROP COLUMN in an existing table is as follows.

    ALTER TABLE table_name DROP COLUMN column_name;

change the DATA TYPE of a column in a table is as follows.

    ALTER TABLE table_name MODIFY COLUMN column_name datatype;

add a NOT NULL constraint to a column in a table is as follows.

    ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

ADD UNIQUE CONSTRAINT to a table is as follows.

    ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

ADD CHECK CONSTRAINT to a table is as follows.

    ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

ADD PRIMARY KEY constraint to a table is as follows.

    ALTER TABLE table_name ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of an ALTER TABLE command to DROP CONSTRAINT from a table is as follows.

ALTER TABLE table_name DROP CONSTRAINT MyUniqueConstraint;

ALTER TABLE table_name DROP INDEX MyUniqueConstraint;

The basic syntax of an ALTER TABLE command to DROP PRIMARY KEY constraint from a table is as follows.

ALTER TABLE table_name DROP CONSTRAINT MyPrimaryKey;

ALTER TABLE table_name DROP PRIMARY KEY;

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

TRUNCATE TABLE table_name;

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];


CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;


DROP VIEW view_name;
```

If they do not satisfy the condition(s) WITH CHECK OPTION, the UPDATE or INSERT returns an error.

Transactional control commands are only used with the DML Commands such as - INSERT, UPDATE and DELETE only.

Transaction Control

The following commands are used to control transactions.

- COMMIT − to save the changes.
- ROLLBACK − to roll back the changes.
- SAVEPOINT − creates points within the groups of transactions in which to ROLLBACK.
- SET TRANSACTION − Places a name on a transaction.

SET TRANSACTION [READ WRITE | READ ONLY];

WILD CARD
LIKE OPERATOR

The percent sign (%)

The temporary tables could be very useful in some cases to keep temporary data. The most important thing that should be known for temporary tables is that they will be deleted when the current client session terminates.

```
CREATE TEMPORARY TABLE SALESSUMMARY (
  product_name VARCHAR(50) NOT NULL
  total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00
  avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00
  total_units_sold INT UNSIGNED NOT NULL DEFAULT 0
);
```

```
DROP TABLE SALESSUMMARY
```

CLONE TABLES

```
    Table: TUTORIALS_TBL
Create Table: CREATE TABLE 'TUTORIALS_TBL' (
 'tutorial_id' int(11) NOT NULL auto_increment,
 'tutorial_title' varchar(100) NOT NULL default '',
 'tutorial_author' varchar(40) NOT NULL default '',
 'submission_date' date default NULL,
 PRIMARY KEY  ('tutorial_id'),
 UNIQUE KEY 'AUTHOR_INDEX' ('tutorial_author')
) TYPE = MyISAM
```

```
SQL> CREATE TABLE `CLONE_TBL` (
 -> 'tutorial_id' int(11) NOT NULL auto_increment,
 -> 'tutorial_title' varchar(100) NOT NULL default
'',
 -> 'tutorial_author' varchar(40) NOT NULL default
'',
 -> 'submission_date' date default NULL,
 -> PRIMARY KEY  (`tutorial_id'),
 -> UNIQUE KEY 'AUTHOR_INDEX' ('tutorial_author')
-> ) TYPE = MyISAM;
```

```
SQL> INSERT INTO CLONE_TBL (tutorial_id,
 ->                         tutorial_title,
 ->                         tutorial_author
 ->                         submission_date
 -> SELECT tutorial_id,tutorial_title,
 ->        tutorial_author,submission_date
 -> FROM TUTORIALS_TBL;
```

```
select first_name || last_name as "Name" from students;
```

```sql
SELECT * FROM sales;

SELECT PRODUCT,COUNT(PRODUCT) FROM sales GROUP BY PRODUCT;

SELECT payment_type,COUNT(payment_type),product FROM sales GROUP BY payment_type,product;

SELECT DISTINCT product FROM SALES ORDER BY PRODUCT DESC;

SELECT DISTINCT city,count(city) as city_orders FROM sales  group by city having city_orders BETWEEN 2 and 3 order by city_orders desc;

SELECT country,count(country)as country_orders from sales group by country having country_orders>7 order by country DESC  ;

SELECT country,count()as country_orders from sales group by country having country_orders>7 order by country DESC  ;

select country ,sum(price) as country_sales from sales group by price;

select country,state ,sum(price)as state_sale from sales group by state order by country;

select city,sum(price)as city_sale,country from sales group by city order by city_sale desc;

select product,sum(price) from sales group by product;
```

```sql
PRAGMA TABLE_INFO(sales);

select DISTINCT product from sales;

SELECT  price from sales  limit 5;

SELECT  price from sales  rownum 5;

select country,sum(price) from sales where product like ("Chair") group by country order by country DESC  ;

select country,sum(price) from sales where product  not like ("Chair") group by country order by country DESC  ;

select * from sales limit 2;

 update sales set price=500  where id=1 and product="Chair";


select * from sales  order by sales.ID DESC limit 2;

SELECT * FROM astronauts WHERE year IS  NULL;

SELECT * FROM astronauts WHERE year IS  not NULL;

alter table astronauts rename COLUMN year to DOBYEAR

ALTER TABLE astronauts ADD COLUMN happy NOT NULL DEFAULT 'yes';
```