SQL

Table of Contents

What is Database?

Database is a collection of information organized for easy access, management and maintenance

- Example:
  - Telephone directory
  - Customer data
  - Product inventory
  - Visitors register
  - Weather records.

Types of Data Models

- Records based logical model
  - Hierachical data model
  - Network data model
  - Relational data model
- object based logical model
  - Entity relationship model.

DBMS Operations

- Adding new files
- Inserting data
- Retrieving data
- Modifying data
- Removing data
- Removing files

Advantages of DBMS

- Sharing of data across applications
- Enhanced security mechanism
- Enforce integrity constraints
- Better transaction support
- Backup and recovery features

Introduction to RDBMS

- A relational database refers to a database that stores data in a structured format, using rows and columns.
- This makes it easier to locate and access specific values within the database.
- It is "relational" because the value within each table are related to each other, Tables may also be related to others tables.
- The relational structure makes it possible to run queries across multiple tables at once.

Features of RDBMS

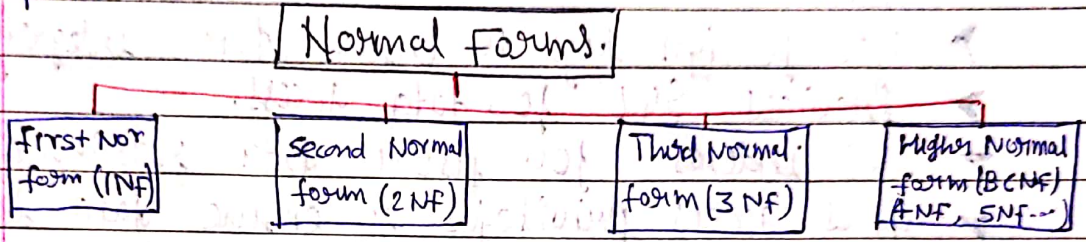
- Every piece of information is stored in the form of tables
- Has primary keys for unique identification of rows
- Has foreign keys to ensure data integrity
- Provides SQL for data access.
- Uses indexes for faster data retrieval.
- Gives access privileges to ensure data security.

RDBMS VS TRADITIONAL APPROCH

- The key difference is that RDBMS applications store data in a tabular form, whereas in tradition approch, applications store data as files.
- There can be, but there will be no "relation" between the tables, like in a RDBMS. In traditional approch, data is generally stored in either a hierarchical form/ navigational form. This means that a single data unit will have 0, 1 or more children nodes and 1 parent node.

Normalization

- Decompose larger, complex table into simpler and smaller ones.
- Moves from lower normal forms to higher normal forms.



Need for Normalization

- In order to produce good database design
- To ensure all database operations to be efficiently performed.
- Avoid any expensive DBMS operations
- Avoid unnecessary replication of information

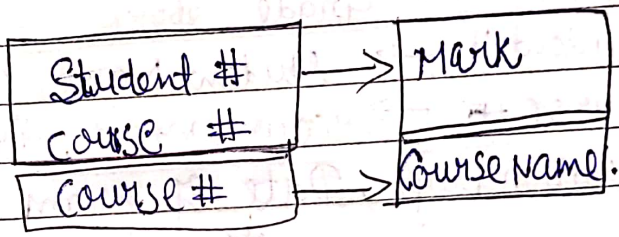
Functional Dependency

- In a given relation R, P and Q are attributes Q is functionally dependent on attribute P if each value of P determines exactly one value of Q.

FD Types 1) Partial functional Dependency
2) Transitive Dependency

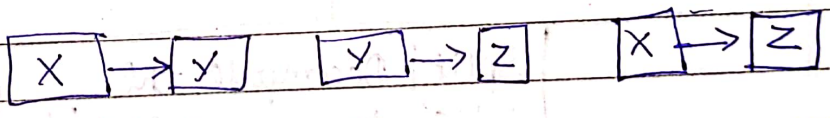
Partial FD :- o Attribute Q is partially dependent on attribute P, if and only if it is dependent on the subset of an attribute P.

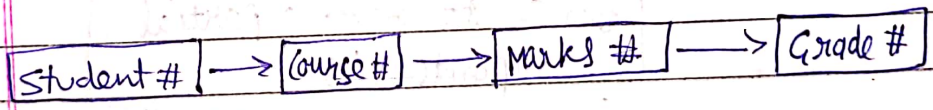
- REPORT (Student#, Course#, Student Name, Course Name Marks, Grade)



Transitive Dependency

X, Y, Z are three attribute





First Normal Form - (1NF)

A relation Schema is in 1NF, if and only if:

- All attributes in the relation are atomic (indivisible value)

And there are no repeating elements or group of elements

Second Normal (2NF)

A relation is said to be in 2NF, if and only if
- it is in 1st Normal form
- Nor partial dependency exits between non-key attributes and key attributes.

- Student #, Course # → Marks
- Student #, Course # → Grade
- Marks → Grade
- Student # → Student Name, DOB
- Course # → Course Name, Pre-Requisite
- Duration Days, Date of exam

↓

Partial Dependency with the key attribute

↓

Split / Decompose the tables to remove partial dependencies

Third NORML FORM (3NF)

A relation R is said to be in 3NF if & only if:
- It is in 2NF
- Nor transitive dependency exists between non key attributes and key attributes through another non-key attribute.

Result_table

student #	Course #	Marks	Grade
101	M1	82	A
102	P4	83	A
109	B3	68	B

Student #, Course # → Marks
Student #, Course # → Grade
Marks → Grade

Student #, Course # → Marks → Grade : TD → Remove

SQL Structured Query language.

programming language specifically designed for working with Database to - -

- CREATE
- MANIPULATE
- SHARE / ACCESS

Advantages :-

- Allows user to communicate i.e, access and manipulate the database.
- Allows users to retrieve data from a database.
- Allows users to create, update, modify and delete the database.

SQL is the language for defining the structure of a database.

SQL Terms

Data Data is defined as a facts or figure or information thats stored in or used by a computer

Database :- A database is an organized collection of data / information so that it can be easily accessed, manged and updated

SQL Data Types :-

1) Numeric :- bit, tinyint, smalint, int, bigint, decimal, numeric, float, real.
2) Character/String :- char, varchar, Text.
3) Date/Time - Date, Time, Datetime, Timestamp, year,
4) Miscellaneous, Json, XML

SQL Constraints

Constraint	Description
Not Null	Ensures that a column does not have a NULL value
Default	Provides a default value for a column when none is specified
Unique	Ensure that all the values in a column are different
Primary	identifies each row/record in a database table uniquely
Check	Ensure that all values in a column satisfy certain condition
index	Create and retrives data from the database very quickly

SQL Command Groups

- DDL [Data Definition language]: Creation of object
- DML [Data Manipulation language]: manipulation of data
- DCL [Data control language]: assignment and removal of permission
- TCL [Transaction control language]: Saving and restoring changes to a database.

DDL - Data Definition Language.

Command	Description
Create	Creates objects in the database (database objects)
Alter	Alters the structures of the database (database project)
Drop	Deletes objects from the database
Truncate	Removes all records from a table Permanently
Rename	Renames an object

Delete everything

Delete records not table.

1) Create Command

```
Create table employees (
emp_id int not null,
first_name varchar(20),
last_name varchar(20),
salary int,
Primary key (emp_id)
);
```

emp_id	first_name	last_name	Salary

★ To see the Table use ↓

```
select * from employees;
```

★ To see the logical structure of table ↓

```
describe employees;
```

→ Alter command (Modify)
Add new column.

alter table employees add column contact int;

→ To change name.

alter table employees rename column contact to Job_code;

→ Truncate Command [To delete records]

TRUNCATE TABLE employees;

→ Drop Command [To delete Table]

DROP TABLE table_name;
DROP TABLE employees;

[DML — Data Manipulation Language]

Command	Description
Insert	Insert data into a table
Update	Updates existing data within a table
Delete	Delete Specific/all records from a table

→ INSERT Command

INSERT INTO employees
(emp_id, first_name, last_name, salary) VALUES
(101, 'Steven', 'King', 10000);

emp_id	first_name	last_name	Salary
101	Steven	King	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

```
INSERT INTO employees
(emp_id, first_name, last_name, salary) VALUES
(102, 'Edwin', 'Thomas', 15000);

INSERT INTO employees
(emp_id, first_name, last_name, salary) VALUES
(103, 'Harry', 'Potter', 20000);
```

DML- UPDATE Command

1.
```
UPDATE employees
SET last_name = 'cohen'
WHERE emp_id = 101;
```

emp_id	first_name	last_name Change	Salary
101	Steven	cohen	10000

2. `update employees set last_name = 'cohen' where emp_id=101;`

DML - DELETE Command

```
DELETE FROM employees WHERE emp_id IN
(101, 103);
```

DCL - DATA Control Language

Command	Description
Grant	Gives access privileges to database
Revoke	withdraws access privileges given with the Grant command.

GRANT <Privilege list> ON <Relation Name> TO <USER>

REVOKE <Privilege list> ON <Relation Name> TO <USER>

TCL → Transaction Control

Command	Description
Commit	Save the work done
Rollback	Restores database to origin state since the last commit
Savepoint	Identify a point in a transaction to which you can roll back later

SQL Operator - Filter

WHERE Clause -

- Used to specify a condition while fetching the data from a single table or by joining with multiple tables
- Not only used in the SELECT Statement, but it is also used in the UPDATE, DELETE Statement, etc.

e.g.

```
SELECT * FROM employees WHERE
emp_id = 101;
```

The example mentioned above extracts all the columns from the table 'employes' whose emp_id = 101

SQL- operator Logical

	Operator	illustrative Example	Result
Both true	AND	(5<2) AND (5>3)	FALSE
anyone true	OR	(5<2) OR (5>3)	TRUE
	NOT	NOT (5<2)	TRUE

Sample Queries:-

```
SELECT * FROM employees WHERE first_name = 'Steven'
  and Salary = 15000;

SELECT *FROM employees WHERE first_name = 'Steven'
   OR Salary = 15000;

SELECT *FROM employees WHERE first_name ='Steven'
  and Salary != 10000;
```

SQL- COMPARISON Operator

Sample Queries;

```
SELECT * FROM employees WHERE first_name ='Steve
  AND Salary <= 10000;

SELECT *FROM employees WHERE first_name='Stev
  OR Salary >= 10000;
```

SQL - Special operators

Between	checks an attribute value within range
Like	checks an attribute value matches a given string pattern
is NULL	checks an attribute value is NULL
IN	checks an attribute value.
DISTINCT	Limits values to unique values.

Sample Queries :-

SELECT * FROM employees WHERE salary between 10000 and 20000;

SELECT * FROM employees WHERE first_name like 'Steven';

SELECT * FROM employees WHERE salary is null;

SELECT * FROM employees where salary in (10000, 12000, 20000);

SELECT DISTINCT (first_name) from employees;

SQL Aggregations

AVG()	Returns the average value from Specified columns
Count()	Returns number of table rows. (Total no. of records)
Max()	Return largest value among the records
Min()	Return Smallest value among the records
Sum()	Returns the sum of Spical column values.

Sample Queries;

SELECT avg (salary) FROM employees;

SELECT count (*) FROM employees;

SELECT min (salary) FROM employees;

SELECT Max (salary) FROM employees;

SELECT SUM (salary) FROM employees;

SQL GROUP BY Clause

Arrange identical data into groups

e.g. SELECT max (salary), dept_id
FROM employees
GROUP BY dept_id

SQL HAVING Clause

- Used with aggregate functions due to its non-performance in the WHERE clause.
- Must follow the Group by clause in a query and must also precede the order by clause if used.

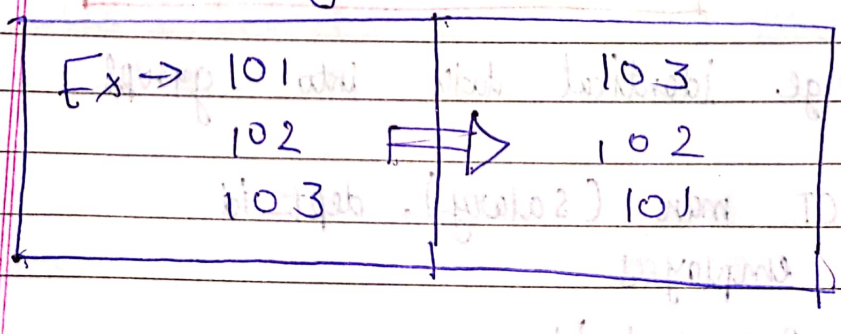
e.g. SELECT AVG (salary), dept_id
FROM employees
GROUP By dept_id
HAVING Count (dept_id) >= 2

Ex- Select first_name, max (salary), dept from employ
group by dept having Count (dept) >= 2;

SQL ORDER BY Clause

- Used to sort output of SELECT Statement
- Default is to sort in ASC (Ascending)
- Can Sort in Reverse (Descending) order with "DESC" after the column name.

e.g., SELECT * FROM employees ORDER BY Salary DESC;



SQL UNION

- Used to combine the result-set of two or more SELECT Statement removing duplicates.
- Each SELECT Statement within the Union must have the same number of columns.
- The selected columns must be of similar data types and must be in the same order in each SELECT Statement.

Product 1

Category-ID	Product_Name
1	Nokia
2	Samsung
3	HP
6	Nikon

Product 2

Category-ID	Product_Name
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	playstation

e.g.

```
SELECT Product_name FROM Product1
UNION
SELECT Product_name FROM Product2;
```

Product_name
Nokia
Samsung
HP
Nikon
LG
Dell
Apple
PlayStation

SQL UNION ALL

- Used to combine the results of two SELECT statement including duplicate rows.
- The same rules that apply to the UNION clause will apply to the UNION ALL operator

SYNTAX : SELECT col1, col2 --- FROM table1
UNION ALL
SELECT col1, col2 --- FROM table2;

P1

category_id	Product_Name
1	Nokia
2	Samsung
3	Hp
6	Nikon

P2

category_id	Product_name
1	samsung
2	LG
3	HP
5	Dell
6	Apple
10	playstation

Product_Name
Nokia
samsung
HP
Nikon
Samsung
LG
HP
Dell
Apple
Playstation

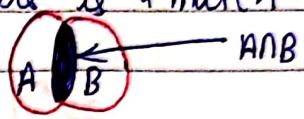
e.g.

```
SELECT product_name FROM Product1
UNION ALL
SELECT Product_name FROM
product2;
```

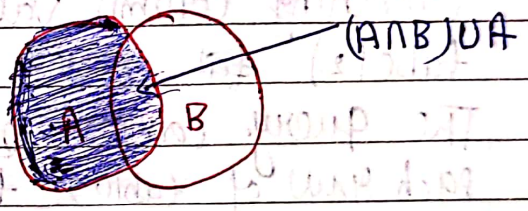
SQL JOINS

Combine rows / columns from two or more tables, based on a related column between them in a database.

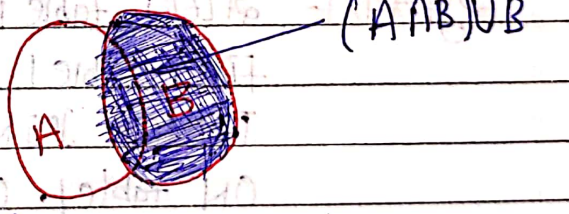
- INNER JOIN :- Returns rows when there is a match in both tables.



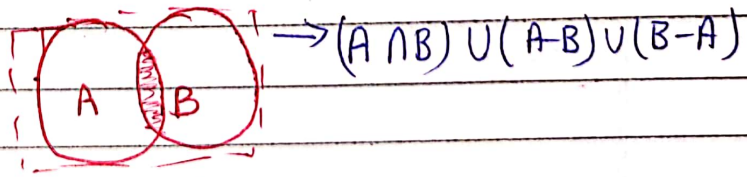
- LEFT JOIN :- Return all rows from the left table, even if there are no matches in the right table.



- Right Join :- Return all rows from the right table even if there are no matches in the left table.

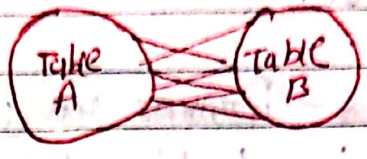


- Full outer Join :- Return rows when there is a match in one of the table.



- Self Join :- 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 (CROSS JOIN) Returns the cartesian product of the set of records from the two or more joined tables



SQL INNER JOIN

The INNER JOIN Creates a new result table by combining column values of two tables (table1 & table2) based upon the Join-predicte. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join predicte.

Syntax:-

```
SELECT table1.col1, table2.col2, ..., table1.col
FROM table1
INNER JOIN table2
ON table1.common_field = table2.commonfield
```