Introduction to database

- Database is a collection of data.
- Data is in the form of tables (structured data) and documents (unstructured data).
- Tables consists of rows and columns.
- Documents consists of key-value pairs.

Background

- Data should be stored permanently. Because, the variables or references we use in JS uses primary memory.
- Primary Memory - RAM (Random Access Memory)
- RAM consists of two parts: heap and stack.
- Heap is used to store data.
- Stack is used to store functions and variables.
- We need a Database to store our data permanently.
- For that, we need a Database Management System (DBMS).
- DBMS is a software that is used to manage the database.
- DBMS is used to create, read, update, and delete data in the database.
- Example of DBMS: MySQL, PostgreSQL, Oracle, MongoDB, SQLite, Redis, Cassandra, Elastic Search, etc.
- Language used to query data: SQL (Structured Query Language).

Queries

- To show all the databases

```sql
SHOW DATABASES;
```

- To create a database

```sql
CREATE DATABASE database_name;
```

- To check the current database

```sql
select database();
```

- To use a database

```sql
USE database_name;
```

- To drop a database

```
DROP DATABASE database_name;
```

- To show all the tables in a database

```
SHOW TABLES;
```

- Example Table: Products

id	name	price	quantity	type	brand	branch
1	Pen	10	100	Stationary	Reynolds	Chennai
2	Pencil	5	200	Stationary	Apsara	Chennai
3	Book	100	50	Stationary	Navneet	Mumbai
4	Laptop	50000	10	Electronics	Dell	Chennai
5	Mobile	20000	20	Electronics	Samsung	Mumbai
6	TV	30000	5	Electronics	Sony	Chennai
7	AC	40000	5	Electronics	LG	Mumbai
8	Fridge	30000	5	Electronics	Whirlpool	Chennai
9	Washing Machine	20000	5	Electronics	Samsung	Mumbai
10	Chair	500	50	Furniture	Godrej	Chennai
11	Table	1000	20	Furniture	Nilkamal	Mumbai
12	Sofa	5000	10	Furniture	Urban Ladder	Chennai
13	Bed	10000	10	Furniture	Pepperfry	Mumbai
14	Cupboard	8000	10	Furniture	Godrej	Chennai
15	Almirah	7000	10	Furniture	Nilkamal	Mumbai

- To create a table

```
CREATE TABLE table_name (
    column1_name column1_datatype column1_constraints,
    column2_name column2_datatype column2_constraints,
    ...
);
```

- To drop a column

```
ALTER TABLE table_name DROP COLUMN column_name;
```

- To add a column

```
ALTER TABLE table_name ADD COLUMN column_name column_datatype
column_constraints;
```

- To rename a column

```
ALTER TABLE table_name CHANGE COLUMN old_column_name new_column_name
column_datatype column_constraints;
```

- To insert data into a table

```
INSERT INTO table_name (column1_name, column2_name, ...)
VALUES (value1, value2, ...);
```

- to view the data in a table

```
SELECT * FROM table_name;
```

- To delete all the data in a table

```
DELETE FROM table_name;
```

- to view the structure of a table

```
DESC table_name;
```

or

```
DESCRIBE table_name;
```

- To modify a column in a table

```
ALTER TABLE table_name MODIFY COLUMN column_name column_datatype
column_constraints;
```

- To modify a column and drop a constraint

```
ALTER TABLE table_name MODIFY COLUMN column_name column_datatype;
```

- Insert data into the products table

```
INSERT INTO products (id, name, price, quantity, type, brand, branch)
VALUES
(1, 'Pen', 10, 100, 'Stationary', 'Reynolds', 'Chennai'),
(2, 'Pencil', 5, 200, 'Stationary', 'Apsara', 'Chennai'),
(3, 'Book', 100, 50, 'Stationary', 'Navneet', 'Mumbai'),
(4, 'Laptop', 50000, 10, 'Electronics', 'Dell', 'Chennai'),
(5, 'Mobile', 20000, 20, 'Electronics', 'Samsung', 'Mumbai'),
(6, 'TV', 30000, 5, 'Electronics', 'Sony', 'Chennai'),
(7, 'AC', 40000, 5, 'Electronics', 'LG', 'Mumbai'),
(8, 'Fridge', 30000, 5, 'Electronics', 'Whirlpool', 'Chennai'),
(9, 'Washing Machine', 20000, 5, 'Electronics', 'Samsung', 'Mumbai'),
(10, 'Chair', 500, 50, 'Furniture', 'Godrej', 'Chennai'),
(11, 'Table', 1000, 20, 'Furniture', 'Nilkamal', 'Mumbai'),
(12, 'Sofa', 5000, 10, 'Furniture', 'Urban Ladder', 'Chennai'),
(13, 'Bed', 10000, 10, 'Furniture', 'Pepperfry', 'Mumbai'),
(14, 'Cupboard', 8000, 10, 'Furniture', 'Godrej', 'Chennai'),
(15, 'Almirah', 7000, 10, 'Furniture', 'Nilkamal', 'Mumbai');
```

- To select all the columns from the products table

```
select * from products;
```

- to select specific columns from the products table

```
select name, price from products;
```

- to select products with branch as Chennai

```
select * from products where branch = 'Chennai';
```

- to select products with branch not as Chennai

```sql
select * from products where branch != 'Chennai';
```

- to select products with price greater than 10000

```sql
select * from products where price > 10000;
```

- to select products with price greater than or equal to 10000

```sql
select * from products where price >= 10000;
```

- to select products with price greater than 10000 and less than 30000 (inclusive)

```sql
select * from products where price > 10000 and price <= 30000;
```

- to select products with branch as 'Chennai' or branch as 'Mumbai'

```sql
select * from products where branch = 'Chennai' or branch = 'Mumbai';
```

- to select products where type is Stationary or Furniture (using or operator)

```sql
select * from products where type = 'Stationary' or type = 'Furniture';
```

- to select products where type is Stationary or Furniture (using in operator)

```sql
select * from products where type in ('Stationary', 'Furniture');
```

- to select products where type is not Stationary or Furniture

```sql
select * from products where type not in ('Stationary', 'Furniture');
```

- to select products where price is between 10000 and 30000 (using between)

```sql
select * from products where price between 10000 and 30000;
```

SQL: Structured Query Language

- DDL: Data Definition Language
    - CREATE DATABASE
    - DROP DATABASE
    - CREATE TABLE
    - DROP TABLE
    - ALTER TABLE
- DML: Data Manipulation Language
    - INSERT INTO
    - DELETE FROM
    - UPDATE
    - SELECT
- DCL: Data Control Language
    - GRANT
    - REVOKE
- TCL: Transaction Control Language
    - COMMIT
    - ROLLBACK
- to update the price of the product with id 1

```sql
update products set price = 15 where id = 1;
```

- to delete the product with id 1

```sql
delete from products where id = 1;
```

- to select distinct branches from the products table

```sql
select distinct branch from products;
```

- to find the sum of the price of all the products

```sql
select sum(price) from products;
```

- to find the min, max, sum, avg, stddev of the price of all the products

```
select min(price), max(price), sum(price), avg(price), stddev(price) from
products;
```

- to find the sum of the price of all the products with respect to the grouped branches

```
select branch, sum(price) from products group by branch;
```

- to order the products based on the price in ascending order

```
select * from products order by price;
```

- to order the products based on the price in descending order

```
select * from products order by price desc;
```

- to limit the number of rows to be displayed

```
select * from products limit 5;
```

- to limit the number of rows to be displayed starting from the 5th row

```
select * from products limit 5 offset 5;
```

Day - 2 Topics

- [x] Normalization
- [ ] Select Queries
- [ ] Joins
- [ ] DB Model Design
- [ ] Pros & Cons of Relational databases
- [ ] Relational Database Vs Non Relational Database

Normalization

- Normalization is a process of organizing the data in the database.
- It is used to reduce redundancy and dependency by dividing the large table into smaller tables and defining relationships between them.
- It is used to eliminate the data anomalies.
- It is used to improve the data integrity (accuracy and consistency of data).

Example Table: Products

id	name	price	quantity	type	brand	branch
1	Pen	10	100	Stationary	Reynolds	Chennai
2	Pencil	5	200	Stationary	Reynolds	Chennai
3	Book	100	50	Stationary	Reynolds	Mumbai
4	Laptop	50000	10	Electronics	Samsung	Chennai
5	Mobile	20000	20	Electronics	Samsung	Mumbai
6	TV	30000	5	Electronics	Samsung	Chennai
7	AC	40000	5	Electronics	Samsung	Mumbai
8	Fridge	30000	5	Electronics	Samsung	Chennai
9	Washing Machine	20000	5	Electronics	Samsung	Mumbai
10	Chair	500	50	Furniture	Nilkamal	Chennai
11	Table	1000	20	Furniture	Nilkamal	Mumbai
12	Sofa	5000	10	Furniture	Pepperfry	Chennai
13	Bed	10000	10	Furniture	Pepperfry	Mumbai
14	Cupboard	8000	10	Furniture	Pepperfry	Chennai
15	Almirah	7000	10	Furniture	Nilkamal	Mumbai

- Splitting Products table into multiple tables

1. Products Table

id	name	price	quantity	type	brand_id	branch_id
1	Pen	10	100	Stationary	1	1
2	Pencil	5	200	Stationary	1	1
3	Book	100	50	Stationary	1	2
4	Lapop	50000	10	Electronics	2	1
5	Mobile	20000	20	Electronics	2	2
6	TV	30000	5	Electronics	2	1
7	AC	40000	5	Electronics	2	2
8	Fridge	30000	5	Electronics	2	1
9	Washing Machine	20000	5	Electronics	2	2
10	Chair	500	50	Furniture	3	1

id	name	price	quantity	type	brand_id	branch_id
11	Table	1000	20	Furniture	3	2
12	Sofa	5000	10	Furniture	4	1
13	Bed	10000	10	Furniture	4	2
14	Cupboard	8000	10	Furniture	4	1
15	Almirah	7000	10	Furniture	3	2

2. Brands Table

id	name
1	Reynolds
2	Samsung
3	Nilkamal
4	Pepperfry

3. Branches Table

id	name
1	Chennai
2	Mumbai

Question: Select the name and price of the products with brand as 'Samsung'

```sql
create table products (
    id int primary key,
    name varchar(255),
    price int,
    quantity int,
    type varchar(255),
    brand_id int,
    branch_id int
)
```

insert data into the products table

```sql
insert into products (id, name, price, quantity, type, brand_id,
branch_id)
values
(1, 'Pen', 10, 100, 'Stationary', 1, 1),
(2, 'Pencil', 5, 200, 'Stationary', 1, 1),
(3, 'Book', 100, 50, 'Stationary', 1, 2),
(4, 'Laptop', 50000, 10, 'Electronics', 2, 1),
(5, 'Mobile', 20000, 20, 'Electronics', 2, 2),
(6, 'TV', 30000, 5, 'Electronics', 2, 1),
(7, 'AC', 40000, 5, 'Electronics', 2, 2),
(8, 'Fridge', 30000, 5, 'Electronics', 2, 1),
(9, 'Washing Machine', 20000, 5, 'Electronics', 2, 2),
(10, 'Chair', 500, 50, 'Furniture', 3, 1),
(11, 'Table', 1000, 20, 'Furniture', 3, 2),
(12, 'Sofa', 5000, 10, 'Furniture', 4, 1),
(13, 'Bed', 10000, 10, 'Furniture', 4, 2),
(14, 'Cupboard', 8000, 10, 'Furniture', 4, 1),
(15, 'Almirah', 7000, 10, 'Furniture', 3, 2);
```

create the brands table

```
create table brands (
    id int primary key,
    name varchar(255)
)
```

insert data into the brands table

```
insert into brands (id, name)
values
(1, 'Reynolds'),
(2, 'Samsung'),
(3, 'Nilkamal'),
(4, 'Pepperfry');
```

create the branches table

```
create table branches (
    id int primary key,
    name varchar(255)
)
```

insert data into the branches table

```
insert into branches (id, name)
values
(1, 'Chennai'),
(2, 'Mumbai');
```

Joins

- Joins are used to combine rows from two or more tables based on a related column between them.
- Types of Joins:
    - Inner Join: Returns the rows when there is a match in both tables.
    - Outer Join
        - Left Join: Returns all the rows from the left table and the matched rows from the right table.
        - Right Join: Returns all the rows from the right table and the matched rows from the left table.
        - Full Join: Returns all the rows when there is a match in one of the tables.

Example

1. Table1

c1	c2	c3
1	1000	x1
2	2000	x2
3	3000	x1

2. Table2

c4	c5
x1	4000
x3	5000

```
create table table1 (
    c1 int,
    c2 int,
    c3 varchar(255)
)
```

```
insert into table1 (c1, c2, c3)
values
(1, 1000, 'x1'),
(2, 2000, 'x2'),
(3, 3000, 'x1');
```

```
create table table2 (
    c4 varchar(255),
    c5 int
)
```

```sql
insert into table2 (c4, c5)
values
('x1', 4000),
('x3', 5000);
```

- Inner Join

```sql
select table1.c1, table1.c2, table2.c5 from table1 inner join table2 on
table1.c3 = table2.c4;
```

- Left Join

```sql
select table1.c1, table1.c2, table2.c5 from table1 left join table2 on
table1.c3 = table2.c4;
```

- Right Join

```sql
select table1.c1, table1.c2, table2.c5 from table1 right join table2 on
table1.c3 = table2.c4;
```

- Full Join

```sql
select table1.c1, table1.c2, table2.c5 from table1 left join table2 on
table1.c3 = table2.c4
union
select table1.c1, table1.c2, table2.c5 from table1 right join table2 on
table1.c3 = table2.c4;
```

DB Model Design

- DB Model Design is a process of designing the database.
- It is used to define the structure of the database.

Example: E-Commerce Website

Features of the Website:

- Users can register and login.
- Users can view the products.
- Users can add the products to the cart.
- Users can place the order.

Tables:

1. Users Table
2. Products Table
3. Cart Table
4. Orders Table

Columns in each table:

1. Users Table
    - id: int Primary Key
    - name varchar(255)
    - email varchar(255)
    - password varchar(255)
    - role (admin, user) varchar(255)
    - created_at datetime
    - updated_at datetime
2. Products Table
    - id int Primary Key
    - name varchar(255)
    - price int
    - quantity int
    - type varchar(255)
    - brand varchar(255)
    - branch varchar(255)
    - created_at datetime
    - updated_at datetime
3. Cart Table
    - id int Primary Key
    - user_id int Foreign Key
    - product_id int Foreign Key
    - quantity int
    - created_at datetime
    - updated_at datetime
4. Orders Table
    - id int Primary Key
    - user_id int Foreign Key
    - product_id int Foreign Key
    - quantity int
    - status (ordered, shipped, delivered) varchar(255)
    - created_at datetime
    - updated_at datetime

Relationships:

1. Users Table and Products Table
    - One User can have multiple products.
    - One Product can have multiple users.
2. Users Table and Cart Table
    - One User can have multiple products in the cart.
    - One Product can have multiple users in the cart.
3. Users Table and Orders Table
    - One User can have multiple orders.