Database:

* Any collection of related information ex: phonebook, shopping list, et·c
* Databases can be stored in different ways i.e., on paper, in your mind, on a computer.

Database management systems:

A special software program that helps users create and maintain a database.

* makes easy to manage large amounts of information
* handles security
* backup's

Create · Read · update · Delete [CRUD]

4 main operations doing with database.

2 Types of databases:

Relational databases (SQL)

* Organise data into one (or) more tables
  - each table has colums and rows
  - A unique key identifies each row

nonrelational (no SQL / not Just SQL)

* Organise data is anything but a traditional table.

RDBMS (Relational database management system)

Helps users to create and maintain a relational database

SQL (Structured query language)

* Standardised languge for interacting with RDBMS
* Used to produce CRUD operations as well as other tasks (user management, security..)
* used to define tables and structures
* SQL code used on one RDMS is not always portable to another without modification.

Database queries:

* Queries are requests made to the database management system for specific information.
* As the database's structure become more & more complex it becomes more difficult to get the specific pieces of information we want.
* A Google search is a query.

Structured Query language (SQL)

Is alanguage used for interacting with RDMS

Create, retrieve, update & delete

Create & manage databases

Design & create database tables

perform administration tasks

SQL (hybrid language) Basically 4 types of languages in one:

Data query language: * used to query the database for information
* get information that is already stored there.

Data definition language: used for defining data base schemas.

Data control language: control acces to the data in the database
user & permissions management

Data manipulation language: use for inserting, updating, deleting data from the database.

Query: a set of instructions by SQL given to the RDBMS to get the required information

```
select   employee.name, employee.age
From     employee
where     employee.salary > 30000 ;
```

Datatypes

INT	whole numbers
DECIMAL (M,N)	Decimal numbers - exact value.
VARCHAR (1)	String of ~~leng~~ text of length 1.
BLOB	Binary large object, stores large data
DATE	'YYYY-MM-DD'
TIMESTAMP	'YYYY-MM-DD HH:MM:SS'

```
CREATE TABLE Student (
    student-id INT PRIMARYKEY,
    name       VARCHAR(20),
    major      VARCHAR(20),
    );
                                        PRIMARYKEY (student-id)

DESCRIBE student;

DROP TABLE student;

~~Alte~~ ALTER TABLE student ADD gpa DECIMAL (3,2);

SELECT * ~~stude~~ FROM student;
INSERT INTO student VALUES( 1, 'Jack', biology');

INSERT INTO student(student_id,name) values (4, 'claire');

~~CRE~~ INT AUTO-INCREMENT

  INSERT INTO student(name,major) values ('Jack', 'biolology');


  UPDATE student
   SET major = 'Bio'
  WHERE major = 'Biology ;
  SET major = 'comp sci'
  WHERE major = 'computerscience;


  UPDATE student
  SET major = 'Biochemistry'
  WHERE major = 'Bio' OR major= 'chemistry';

 UPDATE student
 SET major = 'undecided' , name = 'Tom'
```

```
DELETE FROM student
WHERE student_id = 5;
WHERE name = 'Tom' AND major = 'undecided;
```

```
       → All
SELECT * FROM student;
```

```
SELECT name, major
FROM student;
```

```
SELECT student.name, student.major
FROM student                          } alphabetical order
ORDER By name; → ascend
```

```
ORDER By name DESC;
```

```
SELECT *
FROM student
ORDER By student_id ASC ;
```

```
* SELECT *
  FROM student
  ORDER By major, student_id. ;
```

St.Id	name	major
4	Jack	Bio
1	Jack	Bio
3	Claire	chem
5	mike	C.S
2	kate	sToi

```
ORDER By major, student_id DESC;
```

1	Jack	Bio
4	Jack	Bio
3		
5		
2		

```
* SELECT *
  FROM student
  ORDER By student_id DESC
  LIMIT 2;
```

```
SELECT *
FROM student
WHERE major = 'biology';
```

```
SELECT name, major
FROM student
WHERE major = 'Chemistry' OR major = 'Biology';
```

```
SELECT *
FROM student                                notequalto.
WHERE student_id < 3 AND name <> 'Jack';
```

```
SELECT *
FROM student
WHERE name IN ('claire','kate', 'Mike');
```

```
SELECT *
FROM student
WHERE major IN ('Biology','chemistry') AND student_id > 2;
```

```
CREATE TABLE employee (
    emp_id INT PRIMARYKEY,
    first_name VARCHAR(20),
    last_name VARCHAR(20),
    Birth_day DATE,
    SEX      VARCHAR(1),
    salary   INT,
    super_id  INT,
    Branch_id INT
);
```

```
CREATE TABLE Branch (
  branch_id INT PRIMARYKEY,
  branch_name VARCHAR(40),
  mgr_id INT,
  mgr_start_date DATE,
  ForeignKEY (mgr_id) REFERENCES
          employee(emp_id)
          ON DELETE SET NULL
);
```

```
ALTER TABLE employee
ADD FOREIGN KEY (branch_id)
REFERENCE Branch (branch_id)
ON DELETE SET NULL;
```

```
ALTER TABLE employee
ADD FOREIGN KEY (branch_id)
REFERENCE employee(branch_id)
ON DELETE SET NULL
```

```
CREATE TABLE client (

    client-id INT PRIMARY KEY,
    client-name VARCHAR(20),
    Branch-id INT,
    FORIEIGN KEY (branch-id) REFERENCE Branch(Branch-id) ON DELETE SET NULL
    );


CREATE TABLE works_with (
    emp-id INT ,
    client-id INT,
    total-sales INT,
    PRIMARY KEY (emp-id, client-id) ,
    FOREIGN KEY (emp-id) REFERENCES employee (emp-id) ON DELETE SET CASCADE,
    FOREIGN KEY (client-id) REFERCENCES client (client-id) ON DELETE SET CASCADE
    );
CREATE TABLE branchsupplier (
    branch-id INT,
    suppliername VARCHAR(40),
    supply-type VARCHAR (40),
    PrimaryKEY (branch-id, supplier-name),
    Foreign KEY (branch-id) REFERENCES branch(branch-id) ON DELETE SET CASCADE
  );


INSERT INTO employee VALUES (100, 'David','Wallace', 1916-11-17', 'M', 25000, NULL, NU
INSERT INTO branch VALUES (2, 'scranton', 102, '1992-04-06');
    UPDATE employee
    SET    branch-id = 1
    WHERE  emp-id = 100
```

-Find all employee

```
SELECT *
FROM employee;
```

-Find all clients

```
SELECT *
FROM client;
```

-Find all employee ordered by salary

```
SELECT *
FROM employee
ORDER BY salary;   (ASC)   / ORDER BY salary DESC; { Descending
```

-Find all employee ordered by sex then name

```
SELECT *
FROM employee
ORDER By sex, First_name, last_name
```

-Find first 5 employees in the table.

```
SELECT *
FROM employee
LIMIT 5;
```

-Find the First and last names of all employees

```
SELECT first_name, last_name
FOM employee;
```

- Find the Forename and surnames of all employees

```
SELECT first_name AS fore_name, last_name AS surename
FROM employee;
```

- Find out all the different genders

```
SELECT DISTINCT sex      }  sex
FROM employee;           }  M
                            F

SELECT DISTINCT branch_id   }  branch_id
FROM employee;              }  1
                               2
                               3
```

1- Find the no·of employees

```
SELECT COUNT(emp-id)
FROM employee;
```

- find no·of employee has super-id

```
SELECT COUNT(super-id)
FROM employee;
```

- Find the no·of female employees born after 1970

```
SELECT COUNT(emp-id)
FROM employee
WHERE sex='F' AND Brith-date > '1970-01-01';
```

- Find the average of all employe's salaries

```
SELECT AVG(salary)
FROM employee;
```

averge of all male employe's salaries

```
SELECT AVG(salary)
FROM employee.
WHERE sex='M';
```

- Find the sum of all employe's salaries

```
SELECT SUM(~~em~~salary)
FROM employee;
```

- Find out how many males and females

```
SELECT count(sex)
FROM employee;
```

} all

```
SELECT count(sex), sex
FROM employee
GROUP BY sex;
```

Count(sex)	sex
3	F
6	M

- Find the total sales of each sales-man

```
SELECT SUM(Total-sales), emp-id
FROM ~~emp~~ works-with
GROUP BY emp-id;
```

SUM(Total-sales)	emp-id
282000	102
:	:

- each client

```
SELECT SUM(Total-sales), client_id
FROM works-with
GROUP BY client_id;
```

wildcards

% = any # characters

_ = one character

-Find my client's who are in LLC

```
SELECT *
FROM client
WHERE client_name LIKE '%LLC';
```

-Find my branchsupplier who are in label business

'% label%';

```
SELECT *
FROM branch-supplier
WHERE ~~bran~~ suppliername LIKE '%Label%'
```

-Find my employee born in october.

```
SELECT *
FROM employee
WHERE birth-date LIKE '____-10%';
```

-Find my clients whare are schools

```
SELECT *
FROM client
WHERE client_name LIKE '% school%';
```

UNIONS:

- Find a list of employee and branch names

```
SELECT first_name AS Company-~~N~~ames
FROM employee;
~~UNION~~
SELECT branch_name
FROM branch;
UNION
SELECT client_name
FROM client;
```

- Find a list of all clients & branch supplier's names

```
SELECT client_name, branch_id → client.branch_id.
FROM client
UNION
SELECT supplier_name, branch_id → branch-supplier.branch_id
FROM branch-supplier;
```

- Find the list of all money spent(or) earned by the company

```
SELECT salary, employee.~~branch~~emp-id
FROM employee
UNION
SELECT total-sales, ~~emp~~ workswith.emp-id
FROM works-with;
```

JOINS

Insert INTO branch Values ('4', 'Buffalo', NULL, NULL)

1- Find all branches and the names of their managers

```
SELECT employee.emp-id, employee.first_name, branch.branch_name
 FROM employee
JOIN branch
  ON employee.emp-id = branch.mgr-id ;
```

```
LEFT JOIN branch
  ON employee.em
```

all the employee-id with come.

left side is employee (so all the employ id will come)

NESTED QUERIES:

- Find names of all employees who have sold over 30,000 to a single client

```
SELECT emp-id
 FROM  works_with
WHERE  total sales > 30,000
```

```
SELECT employee.first_name, employee.last_nam
 FROM employee
 WHERE employee.emp_id IN(

 );
```

- Find all clients who are handled by the branch that michael scott manages

SELECT brnch.branch-id
FROM brnch
WHERE branch.~~branch~~mgr-id = 102
LIMIT 1

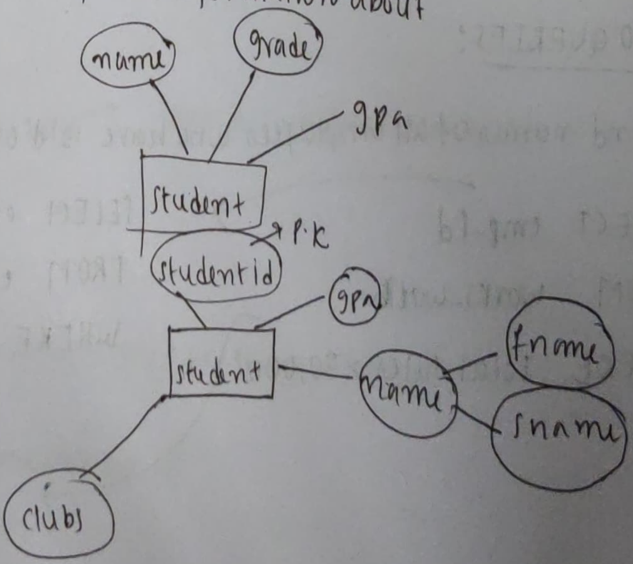
SELECT clientname
FROM client
WHERE client.brnch-id = (

);

```
CREATE TABLE Branch(

branch_id INT

branch_name VARCHAR(20)

mgr_id INT

mg-start-date DATE,

 Foreign KEY (mgr-Id) REFERENCES employee(emp-Id)  ON DELETE SET NULL

     );

 DELETE FROM Employee
 WHERE emp-Id=102;

  SELECT * From branch;
```

---

entity- an object we want to model & store information about


name
grade
gpa
Student
P.K
studentid
gpa
student
fname
name
sname
clubs