DATABASE SHORT NOTE

Data vs Information

Data	Information
Raw, unprocessed facts	Processed and meaningful data
No context or interpretation	Provides context and value
Requires processing	Already processed and useful
Example: "45, 78, 32"	Example: "Average score: 51%"

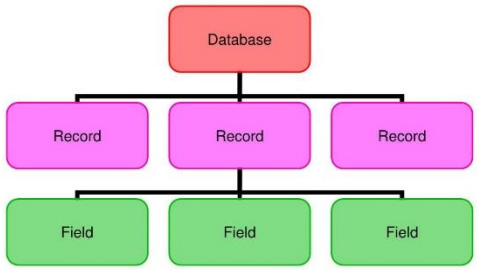
Structured data vs Unstructured data

Structured Data	Unstructured Data
Highly organized and formatted	No predefined format or structure
Easily searchable in databases	Difficult to search and analyze
Stored in rows and columns	Stored in varied formats (e.g., text, images)
Example: Spreadsheets, SQL databases	Example: Emails, social media posts

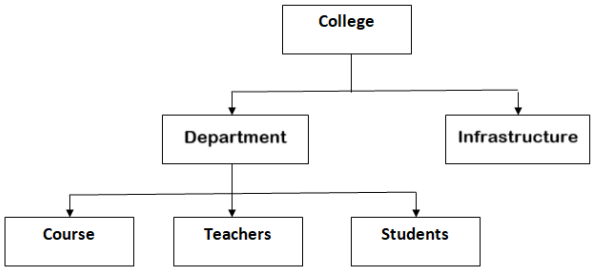
Database: an organized collection of structured data, usually stored on a computer. It is managed by software called a database management system (DBMS).

DBMS Models

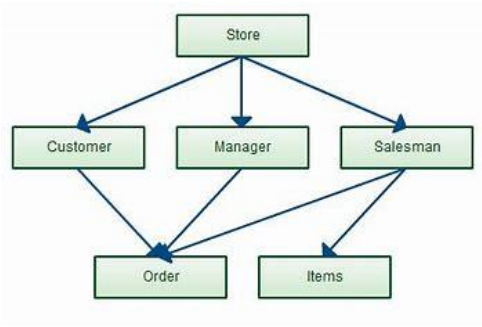
Flat file database -A large collection of data where tables & records are not connected. It have just one table.



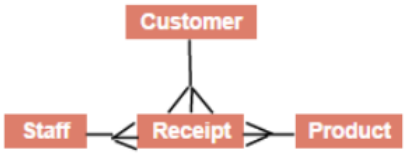
Hierarchical model - Data are organized into a tree-like structure. Presents parent: child relationship (1:M)



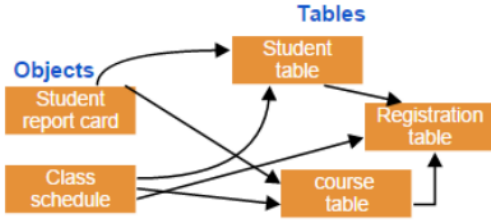
Network Model – Extended version of hierarchical model. Supports more than one parent relationship (M:M)



Relational Model - stores data in the form of relations (tables).



Object relational model - combination of a Object oriented database model and a Relational database model. it supports objects, classes, inheritance etc.



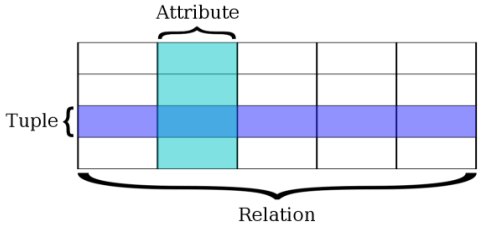
Main components of the relational model

Record/ Tuple: Each row of a table is known as record. It is also known as tuple.

Attributes/Columns : Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, Name, Address.

Degree :The total number of attributes which in the relation is called the degree of the relation.(Number of columns)

Cardinality :Total number of rows present in the Table



Constraints - rules that ensure data consistency when changes are made to the database:

- **NOT NULL Constraint**: Ensures a column cannot have an empty (null) value.
- **Unique Constraint**: Ensures all values in a column are different.
- **Primary Key Constraint**: Uniquely identifies each record in a table. Must be unique and not null.
- **Foreign Key Constraint**: Links a column in one table to the primary key in another, maintaining relationships.
- **Check Constraint**: Limits the range of values that can be entered in a column.

Keys

Candidate key ← Combination of attributes used to identify a record. One of them becomes the primary key

Primary key ← Uniquely identifies each row in a relation

Alternate key ← Candidate key which is not a primary key.

Composite key ← Primary Key with a combination of 2 or more columns

Foreign key ← provides a link between data in two tables

Dependencies

Functional Dependency ← one or more attributes uniquely identify other attributes in a table.

Full Dependency ← All attributes depend on a single attribute.

Partial Dependency ← A non-prime key functionally depend on a part of candidate key.

Transitive Dependency ← Non-prime attribute fully depend On another non-prime attribute.

Data Consistency – Ensures any data is written to a database Must be valid according to defined rules.

Integrity – Maintains the data accuracy and consistency.

i. Integrity constraint – Primary key, Not null, Unique, Foreign key, check, default

ii. Domian Integrity – Ensures all data in a field are within a valid range. Prevents invalid data being stored.

iii. Entity Integrity – Ensures each row in a table uniquely identified by using the primary key.

iv. Referential Integrity – Ensures value in one table references an existing value in another table.

Data atomicity – ensures that all operations within a data entry are either fully completed or none if one part fails.

Data Anomalies

- **Insertion anomaly** –Inability to add data without having some required existing data
- **Update anomaly** - Issues when changing data in one place but not in others, causing inconsistencies.
- **Deletion anomaly** – When deleting a data, losing some other important data

---

Common errors in a database

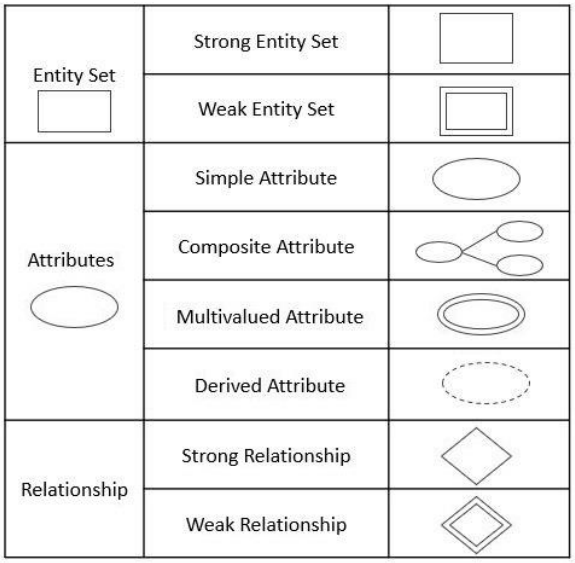
- Data redundancy
- Having null data
- Having data beyond the range
- Data type mismatching

Cardinality/Relationships

- One to one(1:1)
- One to many(1:M)
- Many to many(M:N)

Entity Relationship (ER) Diagrams

ER diagram is a blueprint(plan) we should prepare before creating a physical data storage

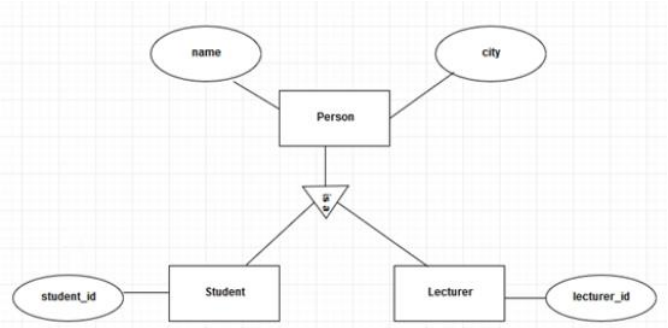


Extended Entity Relationship (EER) diagram is an extention of ER diagrams.

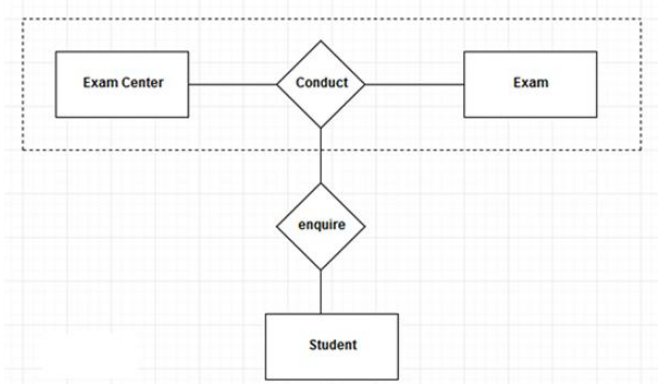
- Generalization
- Specialization
- aggregation.

Generalization- Lower level entities can be combined to produce a higher level entity.(bottom to top approach)

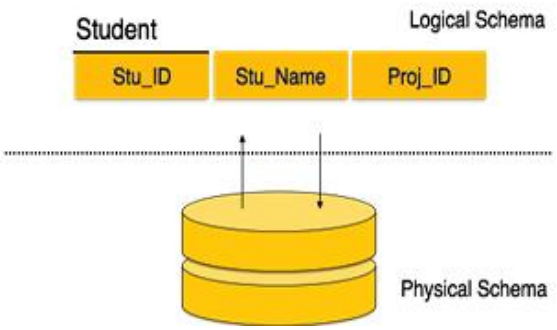
Specialization - high-level entities can be divided into lower level entities. (top to bottom approach)



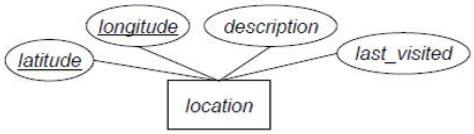
Aggregation - relation between two entities is a treated as a single entity.



Physical schema & Logical schema

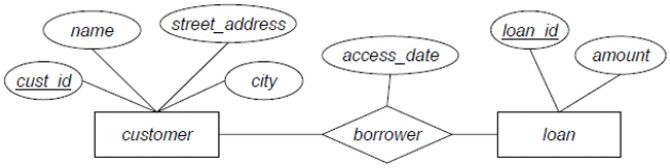


Converting ERD to Logical Schema



location(latitude, longitude, description, last_visited)

M:N ERD to logical schema



customer(cust_id, name, street_address, city)
loan(loan_id, amount)
borrower(cust_id, loan_id, access_date)

Normalization

2 main Purposes:

- Eliminate redundant (useless) data.
- Ensure data dependencies make sense. (data is logically stored)

0NF – Zero Normalization Form

- Extra memory usage
- Anomalies occur

1NF – 1ST Normalization Form

- Removes duplicate records.
- Separate tables for multi-valued attributes.
- Establishes connections using PK & FK

* All the non-key attributes functionally depend on primary key

---

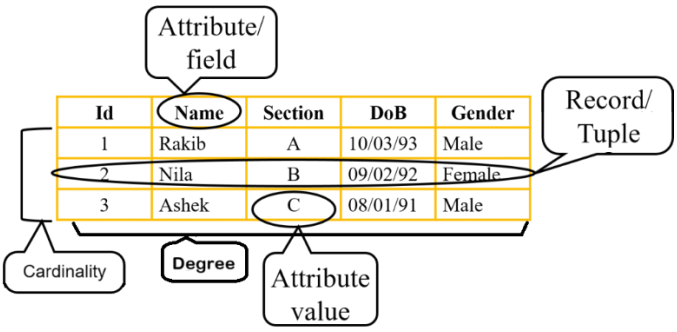
2NF – 2nd Normalization Form

- Removes partial dependency

3NF – 3rd Normalization Form

- Removes Transitive dependency

SQL CHEATSHEET



Constraints

- NOT NULL Constraint → (st_id int(5) **NOT NULL**)
- Unique Constraint → (st_id int(5) **UNIQUE**)
- Primary key constraint → (st_id int(5) **PRIMARY KEY**)
- foreign key constraint → (**FOREIGN KEY**(st_id) **REFERENCES** Students(st_id))
- check constraint → (Age int(2) **check (Age>=18)**)
- default constraint → (Gender varchar(10) **default** 'Male')

SQL → DDL- **Data Definition Language**
DML- **Data Manipulation Language**
DQL- **Data Query Language**
DCL- **Data Control Language**
TCL- **Transaction Control Language**

Data Types → VARCHAR, CHAR, INT, DATE

DDL Commands

- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT

CREATE

- CREATE DATABASE School;
- CREATE TABLE student(
  St_no int(5) PRIMARY KEY,
  Name varchar (20)NOT NULL,
  Sub_id int(3),
  FOREIGN KEY(Sub_id) REFERENCES SUBJECT(Sub_id),
)

ALTER

- ALTER TABLE STUDENT DROP PRIMARY KEY;
- ALTER TABLE ADD Address varchar(30) NOT NULL;
- ALTER TABLE DROP COLUMN Address;
- ALTER TABLE DROP FOREIGN KEY(Sub_id);

DROP

- DROP DATABASE School;
- DROP TABLE Teacher;

DML Commands

- SELECT
- DELETE
- UPDATE
- INSERT

QUERY Commands – Used to carry out tasks
i. SELECT ii. FROM iii. WHERE

- **INSERT INTO Employee**(Empid, Empname, Salary) **Values**(007, "Kasun", "45000") OR

- **INSERT INTO Employee VALUES** (007, "Kasun", "45000");

UPDATE Employee SET Empname= "Kamal" **where** Empid= "007"

DELETE FROM Employee **WHERE** Empid= '007'

Select Statement

- **SELECT** Empid, Salary From Employee
- **(*) -All data → SELECT** * FROM Student (Display all data)

(WHERE) – To Filter Records
SELECT * FROM Employee **WHERE** Department='IT';

Operator	Description	Example
=	Equal	SELECT * FROM Products WHERE Price = 18;
>	Greater than	SELECT * FROM Products WHERE Price > 30;
<	Less than	SELECT * FROM Products WHERE Price < 30;
>=	Greater than or equal	SELECT * FROM Products WHERE Price >= 30;
<=	Less than or equal	SELECT * FROM Products WHERE Price <= 30;
<>	Not equal	SELECT * FROM Products WHERE Price <> 18;
BETWEEN	Between a certain range	
LIKE	Search for a pattern	
IN	To specify multiple possible values for a column	

SELECT * FROM Employee WHERE
Salary **BETWEEN** 50000 **AND** 100000;

SELECT * FROM Employee WHERE
Empname **LIKE** 'a%'**; ← starting with "a"**

SELECT * FROM Employee WHERE
Empname **LIKE** '%an%'**; ← have "an" in the middle**

SELECT * FROM Employee WHERE
Empname **LIKE** '%s'**; ← ends with "s"**

SELECT * FROM Employee WHERE
Empname **LIKE** 'Jo_ _'**; ← starts with 'Jo' and exactly has 2 more characters**

SELECT * FROM Employee WHERE Department
IN ('IT, 'Accounts', 'HR') **← Selects customers from mentioned departments**

SELECT * FROM Employee WHERE
NOT Department='IT'; ← All records except ones from IT

SELECT * FROM Employee WHERE
Department='IT' **AND** Branch='Colombo'; **← select staments where both statements are true**

SELECT * FROM Employee WHERE
Department='IT' **OR** Department= 'HR'; **← select staments where one of the statements are true**

SELECT * FROM employee **ORDER BY** ASC/DESC

Functions

AVG(), SUM(), MAX(), MIN(), COUNT()

SELECT MIN(Salary) FROM Employee;

SELECT SUM(Salary) **as Total** from Employee;

Total

Select Department, sum(Salary) From Employee
GROUP BY Department; ← Group elements

Select Distinct(Department) From Employee;
← Limits Repeated data only to be displayed once

Grouping 2 Tables

SELECT * FROM EMPLOYEE,PROJECT WHERE
EMPLOYEE.EMPNO=PROJECT.EMPNO; ← Display everything in both tables where Empno in both are same

---

SELECT **employee.Empno,** name, SUM(Hours) FROM
Employee,Project WHERE EMPLOYEE.EMPNO=PROJECT.EMPNO;
← If same attributes exists in both tables, should mention from which table we get to query.

Inner Join - selects records that have matching values in both tables
Select Employee.Empno, Salary, Hours From Employee **INNER JOIN Project on** Employee.Empno=Project.Empno;

Multiple Statements

SELECT name FROM city WHERE
country_id IN **(**
SELECT country_id FROM country
WHERE population > 20000000);
←This query finds cities in countries that have a population above 20M: