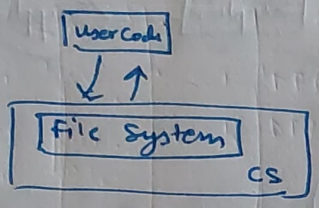
DBMS

A database management system is a collection of software that provides you a quick way to access or modify the data.

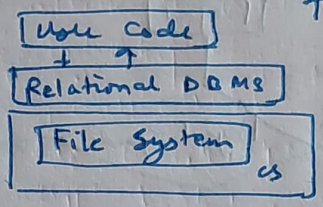
Ex: Oracle, MySQL, SQL server, Etc

* Evaluation of DBMS:

- File Based



- Relational DBMS (1960 +) → Stores data in the form of tables



- NoSQL: Key, value pair / object.
  MongoDB, Dynamo DB

* Entity Relationship Model (ER Model)

- Entity Set: Student, Teacher, Course
- Relationship Set: Teaches, gives, joins
- Attributes: name, date, etc

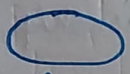
* ER Diagram: Diagram which represents ER model.

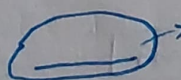
- Entity Set: Rectangle

 → Weak Entity Set

- Relationship Set: Diamond


 → Weak Relationship Set

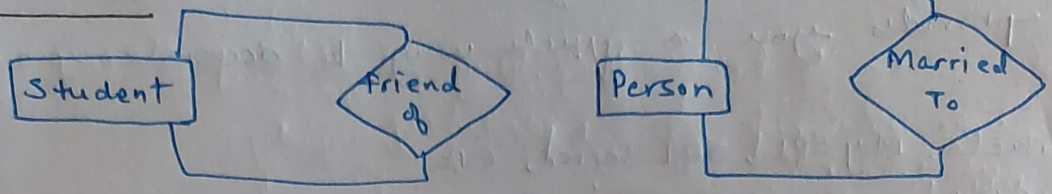
- Attributes:  oval

  - Composite → Name (First Name, Last Name)
    Address (S.No, city, country)
  - Multivalued → Mobile Number 
  - Derived → D.O.B (Age can be derived)
  - Key Attribute → Like enrollment No for Student  → Underline

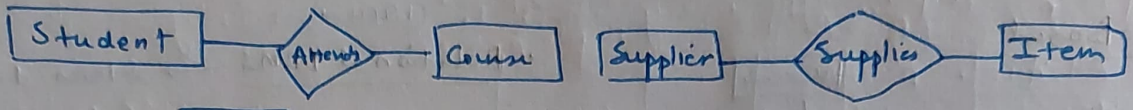
* ER-Diagram (Relationships) :-

- **Degree of Rel^n set**

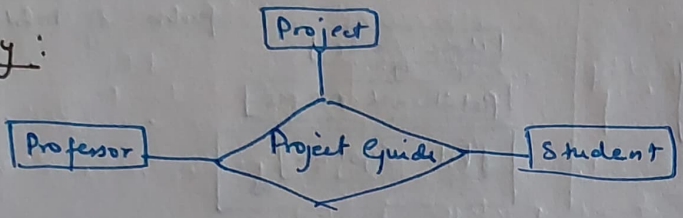
1. **Unary :**



2. **Binary :**



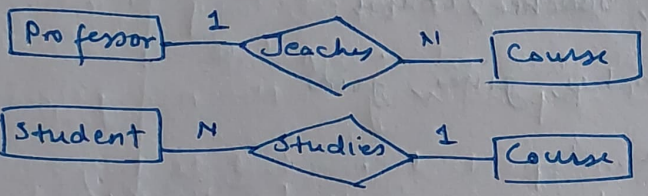
3. **N-ary :**



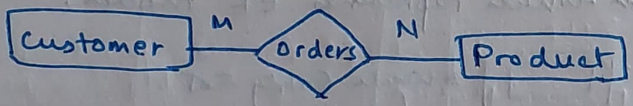
- **Cardinality :**

1. **One to One :**

2. **One to Many :** or **Many to One**



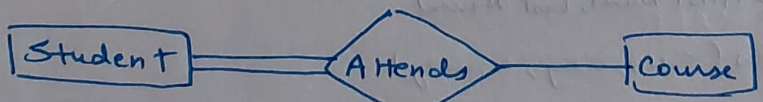
3. **Many to Many :**



* ER Diag (Participation & Weak Entity Set)

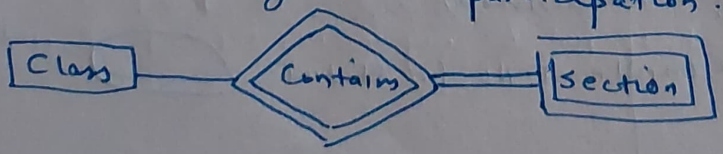
- **Total :** Every entity of one side participates in the relationship.
  Represented by ═══

Ex



- **Weak Entity set :** Does not have their own primary key
  Always total participation.

Ex



ys (Candidate Key, Super Key, Primary Key)

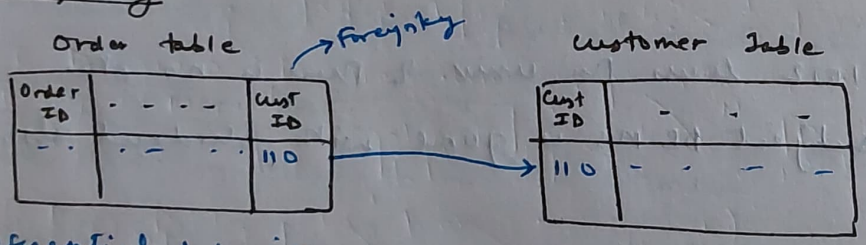
- **Primary Key**: It is the first key which is used to identify one & only one instance of an entity uniquely.
- **Candidate Key**: It is an attribute or set of an attribute which can uniquely identify a tuple.
- **Super Key**: It is a set of an attribute which can uniquely identify a tuple. It is a superset of a candidate key.
- **Foreign Key**: Foreign keys are the column of the table which is used to point the primary key of another table.

1. All attributes which can uniquely denies is super key
2. Among candidate key we chose 1 as primary & rest are alternate keys

* **Armstrong Axioms**:

- **Reflexivity**: $A \rightarrow A$
- **Transitivity**: $A \rightarrow B$, $B \rightarrow C$ $\therefore A \rightarrow C$
- **Augmentation**: $X \rightarrow Y$ $\therefore XZ \therefore YZ$

* **Foreign key**




```
CREATE TABLE Order (
    Orderid Integer primary key,
    Customer.id Integer REFERENCES
    Customer (Customer.ID)
)
```

Referential Integrity

If I delete 110 from customer table, it will get deleted from order table also. (ON DELETE CASCADE)

* **Data Normalization**:

- **Data Redundancy**: ↓ Having same data at multiple places.
- **Data Integrity**: ↑ There should not be any aronious data & it should follow Business Rules.

* Objectives of good database design :-

- No updation, insertion, & deletion anomalies.
- Easily extendible
- Good performance for all query sets
- More informative

* Anomalies

1. Updation Anomaly :

Student ID	Name	Subject ID	Sub Name
1001	ABC	CS101	DBMS
1002	XYZ	CS101	DBMS
1001	ABC	CS102	OS
1003	BCD	CS102	OS

Now, there it is data redundancy. If someone updates one DBMS, other will not change or else he has to change all the occurences which may lead to an error.

Sol → Store redundant data in another table

2. Insertion Anomaly :

Now, if I have to insert a new student in the above table, we can't just insert it without subject ID & name (if they are mandatory)

3. Deletion Anomaly :

Suppose student 1001 & 1002 leave the univ & there is no other subject name DBMS. So subject DBMS is gone with student also.

* Functional Dependency :

A → B, then B is functionally dependent on A.

Enroll No	Name	Addr	No

Enroll No → Name, Addr, No.

Subject ID	Student ID	Marks	Grade

Subject ID, Student ID → Marks, Grade.

A	B
x	1
y	1
z	2

→ A → B ✓

B → A ✗

To reduce redundancy we use functional dependency.

Function Dependency

- Trivial
  - $AB \rightarrow A$
  - $A \rightarrow A$
- Non-Trivial
  - $A \rightarrow B$
  - $A \rightarrow C$

* Database Normalization :-

- **First Normal form**: If every attribute contains only single value. i.e. **Atomic**.

Eg

ID	Name	No
1	A	12, 34
2	B	12
3	C	1234

→ multiple.. ∴ Not Atomic
Not in 1NF

To solve add one more column or have multiple tables

- **Second Normal Form (2NF)**

→ **Candidate key**: Minimal key that define all attributes

→ **Prime attribute**: Any attribute which is part of any candidate key.

2NF → **No Partial Dependency** – No non-prime attribute should depend upon partial candidate key. $(P \rightarrow NP)$ X

- **Third Normal Form**: 3NF

Non-Prime → Non-Prime Not Allowed $(NP \rightarrow NP)$ X

- **BCNF**:

Only Super keys on the left hand side

$P/NP \rightarrow P$ X

SQL & DBMS

- Structured Query Language
- Primary key - Uniquely identifies each row in a table.
- SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column b/w the two.
- (INNER) JOIN - Retrieves records that have matching values in both tables involved the join

```
Select *
from table_A
Join table_B
```

- LEFT (OUTER) JOIN - Retrieves all the records/rows from the left and the matched records/rows from the right table.

```
Select *
from table_A A
Left Join table_B B
on A.col = B.col
```

- RIGHT (OUTER) JOIN - Retrieves all the records/rows from the right & the matched records/rows from the left table.
- FULL (OUTER) JOIN - Retrieves all the records where there is a match in either the left or right table.