* Introduction to the relational modul:-

What is Relational Modul:-

The Relational modul represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table dinote a real World Entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational modul, data are stored as tables However, the physical storage of the data is independent of the way the data are logically organized,

Some popular Relational database management systems are:

→ DB2 and informix Dynamic Server - IBM
→ Oracle and RDB - Oracle
→ SQL server and Access - Microsoft

Relational Modul Concepts:-

1. Attribute:- Each column in a table. Attributes are the properties which define a relation. ex:- Stu-Rollno, Name etc.

2. Tables:- In the Relational modul the relations are saved in the table format. It is stored along with its Entities. A table has two properties rows and Columns. Rows represent records and columns represent attributes

3. Tuple:- It is nothing but a single row of a table, which contains a single record. tuple also called Record.

4. Relation schema:- A relation schema represents the name of the relation with its attributes.

5. Degree:- The total number of attributes which in the relation is called the degree of the relation. The degree, also called arity, of a relation is the number of fields.

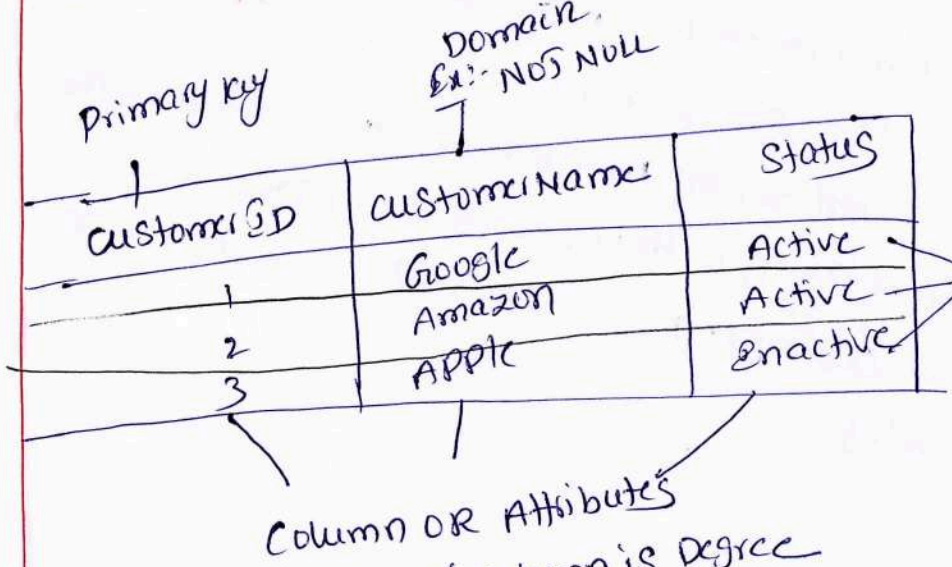
6. Cardinality:- Total number of rows present in the table.

7. Column:- The column represents the set of values for a specific attribute.

8. Relation instance:- Relation instance is a finite set of tuples in the RDBMS System. Relation instances never have duplicate tuples.

9. Relation key:- Every row has one, two or multiple attributes, which is called relation key.

10) Attribute domain:- Every attribute has some pre-defined value and scope which is known as attribute Domain.



A relational database is a collection of relations with distinct relation names. A relational database schema is the collection of schemas for the relations in the database.

Ex:- A university database with relations called Students, Faculty, Courses, Rooms, Enrolled, Teaches, and Meets_In. An Instance of a relational database is a collection of relation instances, one per relation schema in the database schema; Of course, each relation instance must satisfy the domain constraints in its schema.

Advantages:-

1. Simplicity:- A relational data modul is simpler than the hairarchical and network modul.

2. Structural Independence:- The relational database is only concerned with data and not with a structure. This can improve the performance of the modul.

3. Easy to use:- The relational modul is easy as tables consisting of rows and columns is quite natural and simple to understand

4. Query capability:- It makes possible for a high-level query language like SQL to avoid complex database navigation.

5. Data Independence:- The structure of a database can be changed without having to change any application.

6. Scalable:- Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages:-

1. Few relational databases have limits on field lengths which can't be exceeded.
2. Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
3. Complex relational database systems may lead to isolated databases where the information can not be shared from one system to another.

* Integrity Constraints Over Relations:-

An integrity constraint (IC) is a condition specified on a database schema and restricts the data that can be stored in an instance of the database. If a database instance satisfies all the integrity constraints specified on the database schema, it is a legal instance. A DBMS enforces integrity constraints, in that it permits only legal instances to be stored in the database.

Integrity constraints ensure that changes (update, deletion, insertion) made to the database by authorized users do not result in a loss of data consistency.

Integrity constraints are specified and enforced at different times:

1) when the DBA or End user defines a database schema, he or she specifies the ICs must hold on any instance of this database

2) when a database application is run, the DBMS checks for violations and disallows changes to the data that violate the specified ICs. When integrity constraints are checked relative to the statement that causes the change in the data and the transaction that it is part of.

Constraints enforce limits to the data or type of data that can be inserted / updated / deleted from a table. The whole purpose of constraints is to maintain the data integrity during an update / delete / insert into a table.

There are 6 different types of constraints:

1) Domain Constraint
2) Tuple uniqueness Constraint
3) key constraint
4) Entity Integrity Constraint
5) Referential Integrity Constraint
6) General constraints.

The mainly discuss with 3 constraints:

1) key constraint 2) Referential Integrity constraint
3) General constraints.

1) Key constraint:- Key constraint specifics that in any relation

→ All the values of primary key must be unique.

→ The values of primary key must not be null.

Ex:-

Stu_ID	Name	Age
S001	Akshay	20
S001	Abhishek	21
S003	Shashank	20
S004	Rahul	20

∴ This relation does not satisfy the key constraint as here all the values of primary key are not unique.

Unique key:- 1) No duplicate values
2) but null values.

Ex:- 1) Create table students (sid char(20), name char(30), login char(20), age Integer, gpa Real, Unique (name, age), constraint studentskey primary key (sid))

∴ this table Ex:- 2)

sid	sname	branch
101	sai	CSE
102	shiva	CSE
-	sri	CSE

A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple. A set of fields that uniquely identifies a tuple according to a key constraint is called a candidate keys for the relation.

→ Two distinct tuples in a legal instance (an instance that satisfies all ICs including the key constraint) cannot have identical values in all the fields of a key.

→ No subset of the set of fields in a key is a unique identifier for a tuple.

2) Referential Integrity constraint:- This constraint is enforced when a foreign key references the primary key of a relation.

It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

There are two different types of keys:

1) primary key 2) foreign key constraint:-

1) primary key constraint:- The combination of unique and NOT NULL.

1) No duplicate values.

2) No null values.

Ex:

Sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@cs	19	3.2
53650	Madayan	madayan@music	11	3.8

Primary key

Students table.

Create table Students (sid int primary key, name char(15),
login char(20), age number(2), gpa real);

2) Foreign key constraint:- The information stored in a relation is linked to the information stored in another relation. If one of the relation is modified, the other must be checked, and perhaps modified, to keep the data consistent. The most common pc involving two relations is a foreign key constraint.

(1)

Ex:

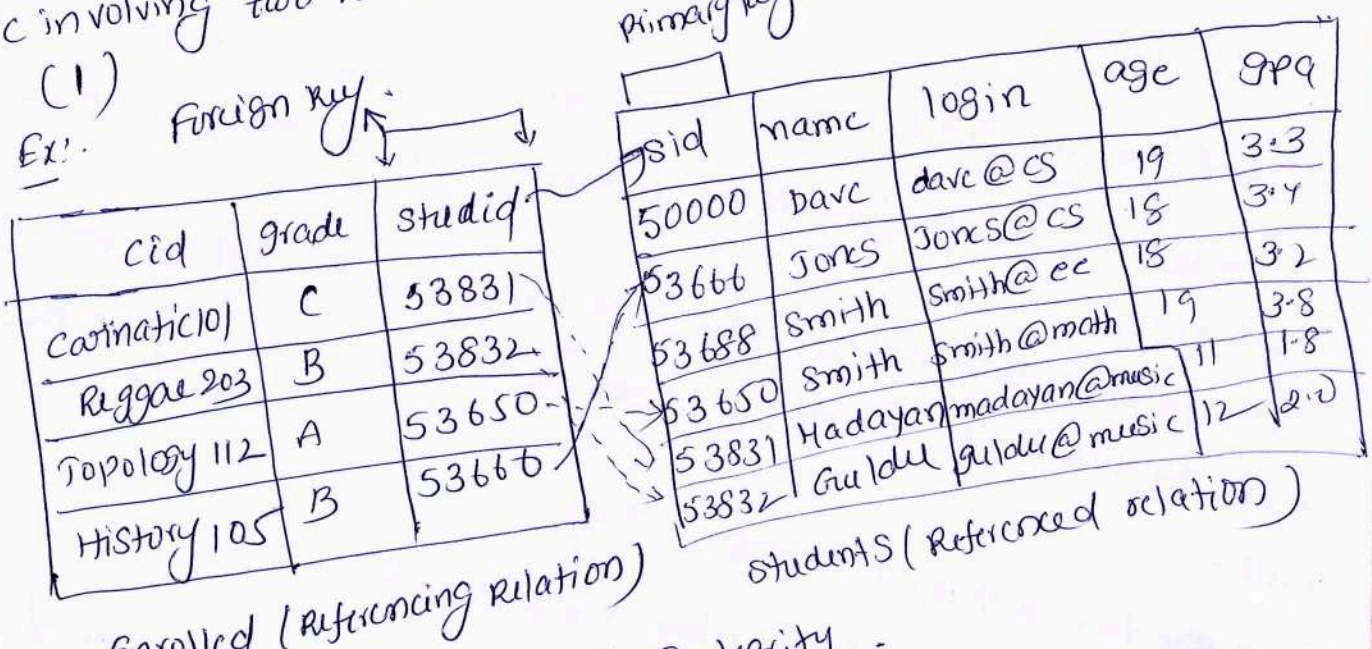


Fig:- Referential Integrity.

Ex: i) students table creation:-

Create table students (sid int primary key, name char(15),
login char(20), age number(2), gpa real);

ii) Enrolled table creation: using primary key and foreign key:-

create table Enrolled (studid char(20), cid char(20), grade
char(10), primary key (studid, cid),
Foreign key (studid) References students);

Ex:-(2) Dept table

Primary key.

Dept_id	Dname
10	CSE
20	ECE
30	CIVIL

Primary key Employee table

Foreign key

Emp_id	E_name	sal	dept_id
1	A	10000	10
2	B	20000	20
3	C	30000	30.

To create Dept table:-

```
Create table Dept (Dept_id number(10) primary key,
                   Dname char(10));
```

To create Employee table:-

```
Create table Employee (Emp_id number(10) primary key,
E_name char(10), sal real, foreign key(
dept_id) References Dept(Dept_id));
```

NULL:- The use of null in a field of a tuple means that value in that field is either unknown or not applicable.

The appearance of null in a foreign key field does not violate the foreign key constraint. However, null values are not allowed to appear in a primary key field (because the primary key fields are used to identify a tuple uniquely).

Sid	Sname	age
101	A	16
102	B	17
-	C	18

null value

(or)

Sid	Sname	age	Phoneno
1	A	10	1234567
2	B	20	145678
3	C	30	-

null

3) General Constraints:-

current relational database systems support such general constraints in the form of table constraints and assertions.

Table constraints:- Table constraints are associated with a single table and checked whenever that table is modified.

Assertions:- Assertions involve several tables and are checked whenever any of these tables is modified.

Both table constraints and assertions can use the full power of SQL queries to specify the desired restrictions.

* Operations in Relational Model:-

Four basic update operations performed on relational database model are insert, update, delete and select.

- insert is used to insert data into the relation
- delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert operation:- The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

Customer ID	customer name	Status
1	Google	Active
2	Amazon	Active
3	APPle	Inactive

insert

customer ID	customer name	Status
1	Google	Active
2	Amazon	Active
3	APPle	Inactive
4	Alibaba	Active

Update operation:- We can see that in the below-given relation table customerName = 'APPle' is updated from Inactive to Active

Cust_ID	cust_name	Status
1	Google	Active
2	Amazon	Active
3	APPle	Inactive
4	Alibaba	Active

Update

cust_ID	cust_name	Status
1	Google	Active
2	Amazon	Active
3	APPle	Active
4	Alibaba	Active

Delete operation:- To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

CUST_ID	cust_name	Status
1	Google	Active
2	Amazon	Active
3	APPle	Active
4	Alibaba	Active

Delete

cust_ID	cust_name	status
1	Google	Active
2	Amazon	Active
3	Alibaba	Active

In the above-given Example, CUST_Name = "APPle" is deleted from the table

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

Select operation:-

CUST_ID	CUST_NAME	STATUS
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

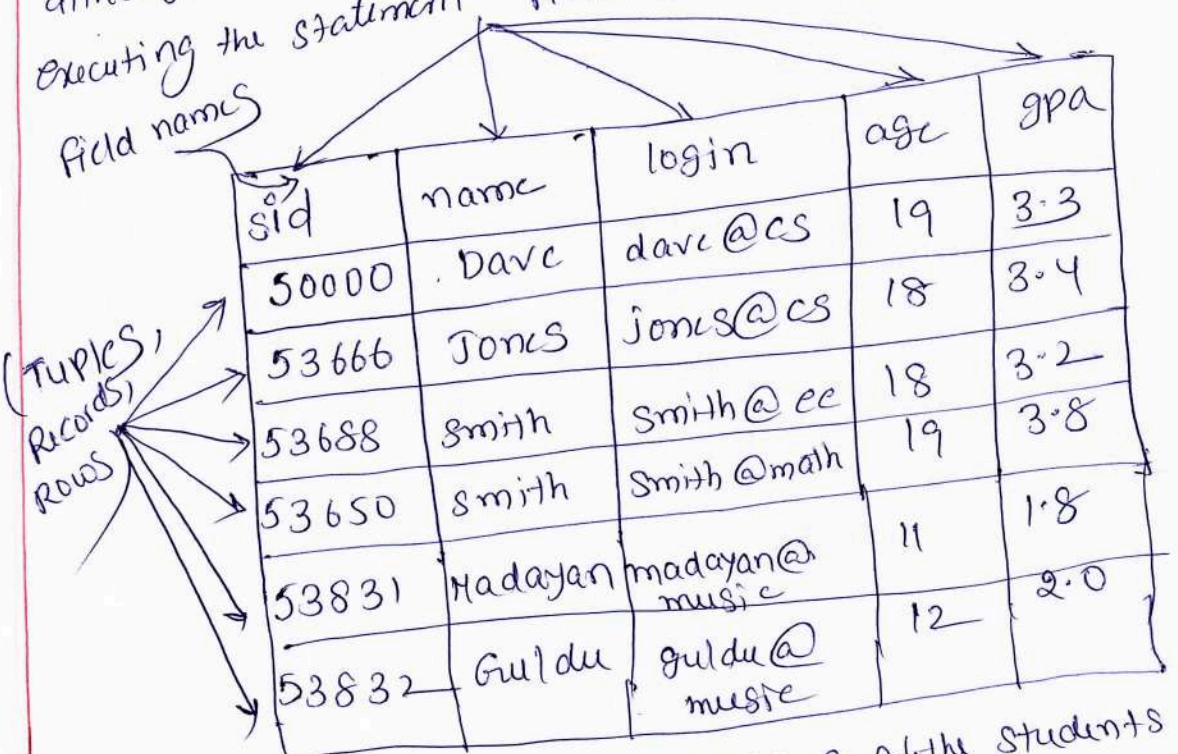
Select

CUST_ID	CUST_NAME	STATUS
2	Amazon	Active

In the above given example, CUST_NAME = "Amazon" is selected.

* Enforcing Integrity Constraints:-

Integrity constraints are specified when a relation is created and Enforced when a relation is modified. The impact of domain, primary key, and unique constraints is straightforward: If an insert, delete, or update command causes a violation, it is rejected. Every potential integrity constraint violation is generally checked at the end of each SQL statement execution, although it can be deferred until the end of the transaction executing the statement.



sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Fig: An instance S1 of the Students Relation

The following insertion violates the primary key constraint because there is already a tuple with the sid 53688, and it will be rejected by the DBMS:

```
INSERT
  INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Mike', 'mike@ee', 17, 3.4)
```

The following insertion violates the constraint that the primary key can not contain null:

```
INSERT
  INTO Students (sid, name, login, age, gpa)
VALUES (null, 'Mike', 'mike@ee', 17, 3.4)
```

Deletion does not cause a violation of domain, primary key or unique constraints. However, an update can cause violations, similar to an insertion:

```
UPDATE Students S
  SET  S.sid = 50000
WHERE  S.sid = 53688;
```

This update violates the primary key constraint because there is already a tuple with sid 50000.

We discuss the referential integrity enforcement steps taken by the DBMS in terms of our Enrolled and Students table, with the foreign key constraint that Enrolled.sid is a reference to Students.

With the References of Referential Integrity figure:

Deletion of Enrolled tuples do not violate referential integrity, but insertion of Enrolled tuples could. The following insertion is illegal because there is no Students tuple with sid 51111:

```
INSERT
INTO Enrolled (cid, grade, studid)
VALUES ('Hindi101', 'B', 5111);
```

Consider students and Enrolled tables:

→ sid in Enrolled is a foreign key that references the student table

→ what should be done if an Enrolled tuple with a nonexistent student id is inserted?

- Reject it.

→ what should be done if a student, s tuple is deleted?

- Also delete all Enrolled tuples that refer to it.
- Disallow deletion of a students tuple that is referred to.
- Set sid in Enrolled tuples that refer to it to a default sid.

(. In SQL also: Set sid in Enrolled tuples that refer to it to a special value null, denoting 'unknown' or 'inapplicable')

→ similar if primary key of students tuple is updated.

Specifying behavior on Referential Integrity violation:-

→ Behavior specified at table create

- NO action (Reject action that violates constraint)
- update referring table (update foreign key to the new value)
- Set to NULL (set all foreign keys to a NULL)
- Set to a Default (set all foreign keys to a default value)

Create table Enrolled (sid char(20), cid char(20), grade char(2), primary key (sid, cid), foreign key (sid) References Students ON DELETE CASCADE ON UPDATE SET DEFAULT)

* Querying Relational Data:-

A relational database query (query, for short) is a question about the data, and the answer consists of a new relation containing the result. A query language is a specialized language for writing queries.

SQL is the most popular commercial query language for a relational DBMS.

Ex:-

Sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Ans:- An instance S_1 of the Students Relation.

Query:- 1) Go we can retrieve rows corresponding to students who are younger than 18 with the following SQL query:-

```
SELECT *
FROM Students S
WHERE S.age < 18;
```

The Symbol '*' means that we retain all fields of selected tuples in the result. S as a variable that takes on the value of each tuple in Students, one tuple after the other. The condition S.age < 18 in the WHERE Clause specifies that we want to select only tuples in which the age field has a value less than 18.

Sid	name	login	age	gpa
53831	Madayan	Madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Fig: Students with age < 18 on Instance S_1

The condition S.age < 18 involves an arithmetic comparison of an age value with an integer and is permissible because the domain of age is the set of integers.

Query 2:- We can compute the names and logins of students who are younger than 18.

```
SELECT S.name, S.login
FROM   Students S
WHERE  S.age < 18;
```

name	login
Madayan	madayan@music
Guldu	guldu@music

Fig:- Names and logins of Students under 18

Query 3:- We can also combine information in the Students and Enrolled relations. If we want to obtain the names of all students who obtained an A and the id of the course in which they got an A.

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid = E.studid AND E.grade = 'A';
```

This Query can be understood as follows: "If there is a Students tuples and an Enrolled tuple E such that S.sid = E.Studid (so that S discribes the student who is Enrolled in E) and E.grade = 'A', then print the Student's name and the courscid".

* Logical Database Design: ER To Relational

The ER model is convenient for representing an initial, high-level database design.

1) Entity Sets to Tables:- Each attribute of the Entity set becomes an attribute of the table. The domain of each attribute and the (primary) key of an entity set.

Ex:- consider the Students entity set with attributes sid, sname and age.

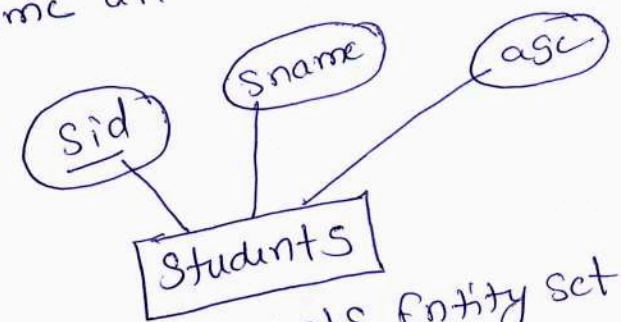


Fig: Students Entity Set.

Table:-

sid	sname	age
101	A	20
102	B	21
103	C	22

Fig:- An instance of the Students Entity Set

Table creation:-
```
create table Students (sid int,
                       sname char(10),
                       age number(10),
                       primary key (sid));
```

2) Relationship Sets (without constraints) to Tables:-

A relationship set, like an Entity set, is mapped to a relational ~~model~~ in the relational model.

The attributes of the relation include:

→ The primary key attributes of each participating entity set, as foreign key fields.

→ The descriptive attributes of the relationship set.

Ex: consider the Works_In2 relationship set. Each Department has offices in several locations and we want to record the locations at which each employee works.

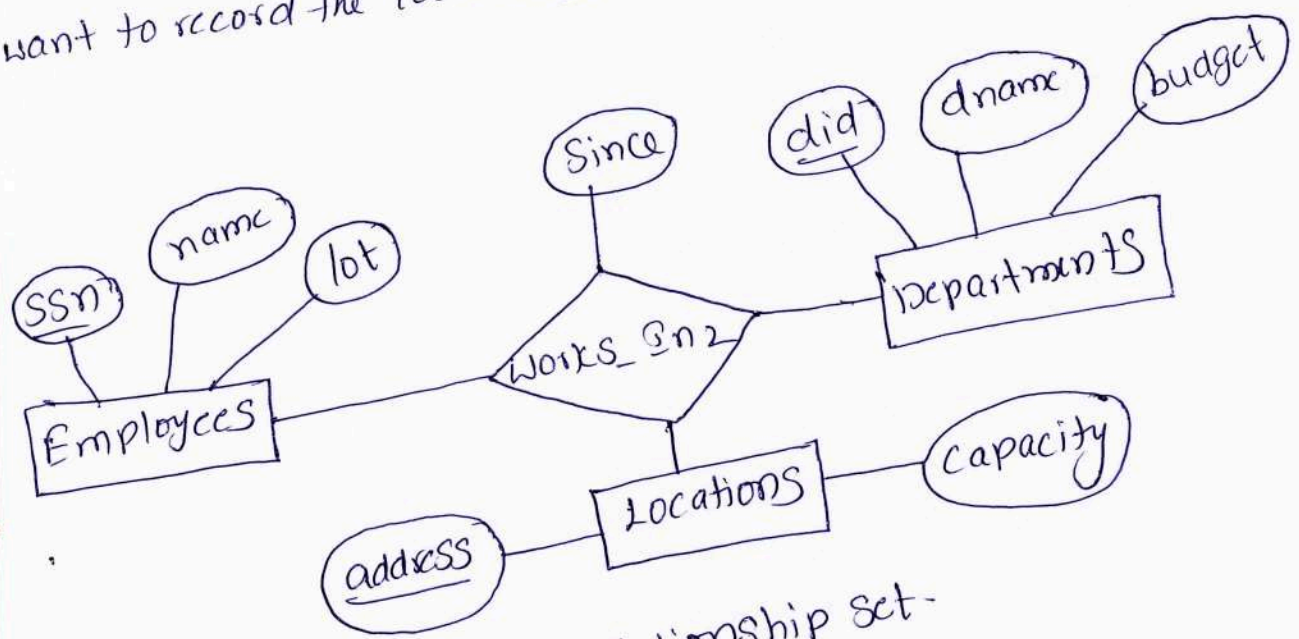


Fig: A Ternary Relationship Set.

```
Create table Works_In2 (SSn char(11),
                        did Integer,
                        address char(20),
                        since Date,
                        Primary key (ssn, did, address),
                        Foreign key (ssn) References Employees,
                        Foreign key (address) Reference Locations,
                        Foreign key (did) References Departments);
```

address, did and ssn fields cannot take on null values. Because these fields are part of the primary key for works_In2, a NOT NULL constraint is implicit for each of these fields.

Consider the Reports_To Relationship set

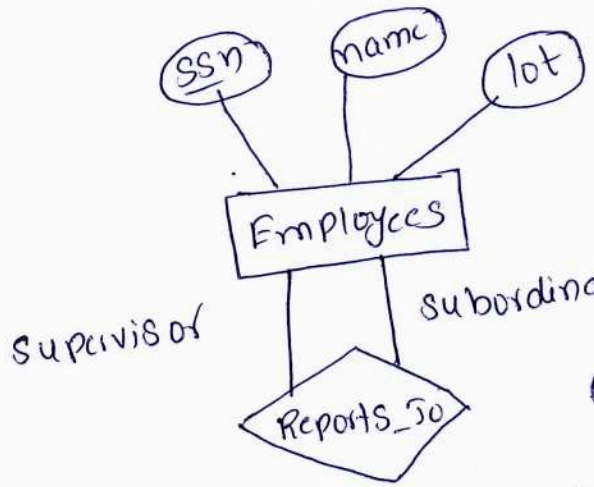


Fig:- The Reports_To Relationship set

The role indicators supervisor and subordinate are used to create meaningful field names in the Create statement for the Reports_To table:

```
Create table Reports_To (supervisor_ssn char(11),
subordinate_ssn char(11), primary key (supervisor_ssn,
subordinate_ssn), foreign key (supervisor_ssn) References
Employee (ssn), foreign key (subordinate_ssn) References
Employees (ssn));
```

Translating Relationship Sets with Key constraints:-

If a relationship set involves n entity sets and some m of them are linked via arrows in the ER diagram, the key for any one of these m entity sets constitutes a key for the relation to which the relationship set is mapped. Hence we have m candidates keys, and one of these should be designated as the primary key.

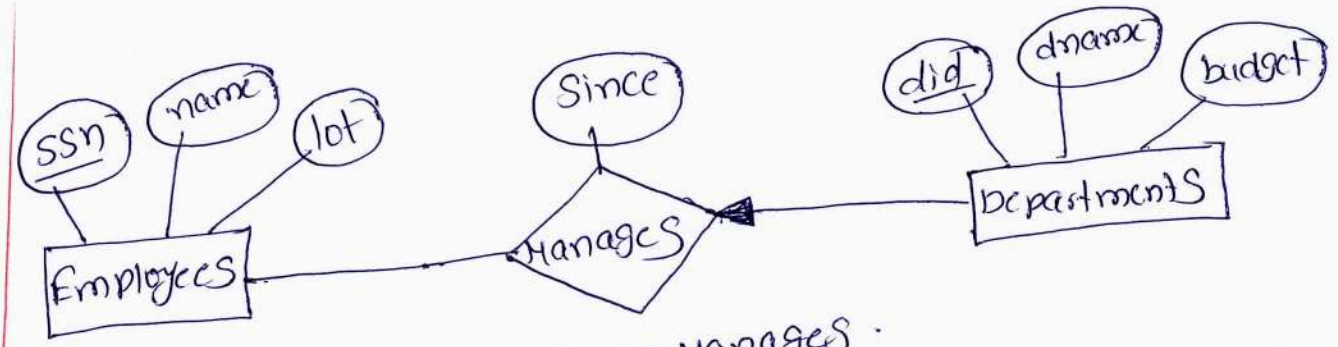


Fig:- Key constraint on Manages.

The table corresponding to Manages has the attributes SSN, did, Since. Each department has at most one manager, no two tuples can have the same did value but differ on the SSN value.

Create table Manages (SSN Char(11), did integer, Since Date, Primary Key (did), Foreign Key (SSN) References Employees, foreign key (did) References Departments);

A second approach to translating a relationship set with key constraints is often superior because it avoids creating a distinct table for the relationship set.

Ex:- A Department has at most one manager, we can add the key fields of the Employees tuple denoting the manager and the Since attribute to the Department tuple.

This approach eliminates the need for a separate Manages relation, and queries asking for a department's manager can be answered without combining information from two ~~tabl~~ relations.

The following SQL statement, defining a Dept_Mgr relation that captures the information in both Departments and Manages.

The second approach to translating relationship sets with key constraints:

```
Create table Dept_Mgr ( did Integer,
                        dname char(20),
                        budget Real,
                        SSN    char(11),
                        since  Date,
                        primary key (did),
                        Foreign key (SSN) References Employees);
```

SSN can take on null values.

Translating Relationship sets with participation constraints:-

Every department is required to have a manager, due to the participation constraint, and at most one manager, due to the key constraint.

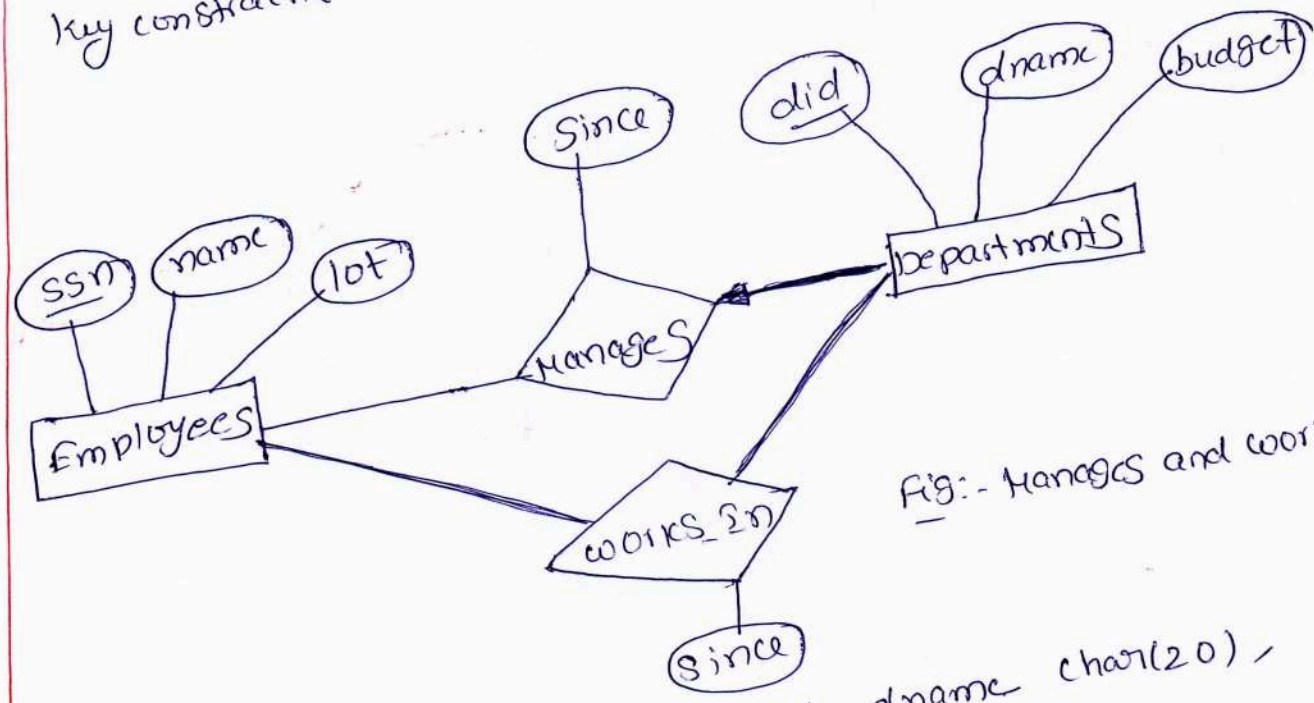


Fig:- Manages and works_In

```
Create table Dept_Mgr ( did Integer, dname char(20),
                        budget Real, SSN char(11) NOT NULL, since Date,
                        primary key (did), foreign key (SSN) References Employee
                        ON DELETE NO ACTION );
```

SSn can not take on null values, each tuple of Dept_Mgr identifies a tuple in Employees. The NO ACTION specification, which is the default and need not be explicitly specified, ensures that an Employees tuple can not be deleted while it is pointed to by a Dept_Mgr tuple. If we wish to delete such an Employees tuple, we must first change the Dept_Mgr tuple to have a new Employee as manager.

Translating Weak Entity Sets:-

A weak Entity set always participates in a one-to-many binary relationship and has a key constraint and total participation.

Consider the Dependents weak Entity set, with partial key pname. A Dependents entity can be identified uniquely only if we take the key of the owning Employees entity and the pname of the Dependents entity, and the Dependents Entity must be deleted if the owning Employees entity is deleted.

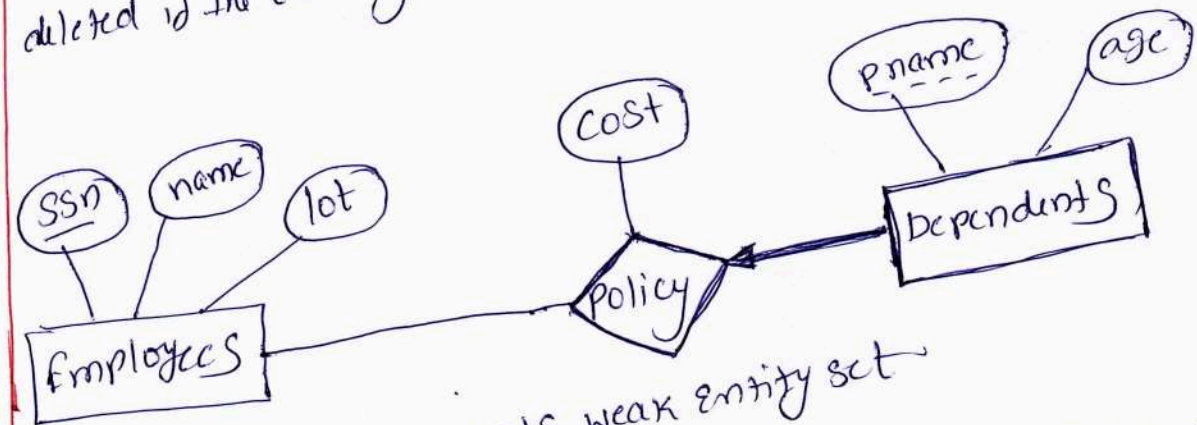


Fig: The Dependents weak Entity set

Create table Dep_policy (pname char(20), age integer, cost Real, SSn char(11), primary key(pname, SSn), foreign key (SSn) References Employees ON DELETE CASCADE)

The CASCADE option ensures that information about an Employee's policy and dependents is deleted if the corresponding Employees tuple is deleted.

Translating class Hierarchies:-

We present the two basic approaches to handling ISA heirarchies

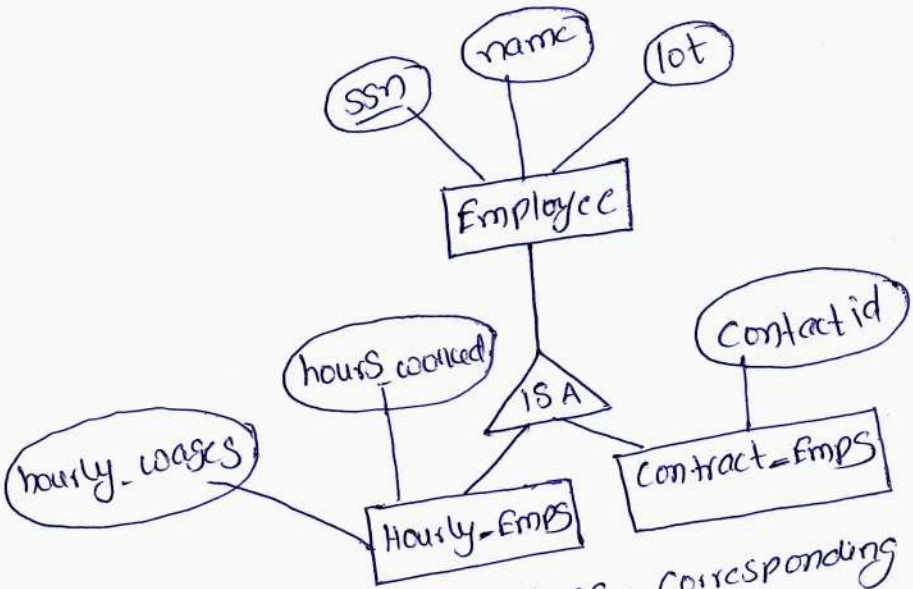


Fig:- class hierarchy.

We can create just two relations, corresponding to Hourly Emps and Contract_Emps. The relation for Hourly_Emps includes all the attributes of Hourly_Emps as well as all the attributes of Employees.

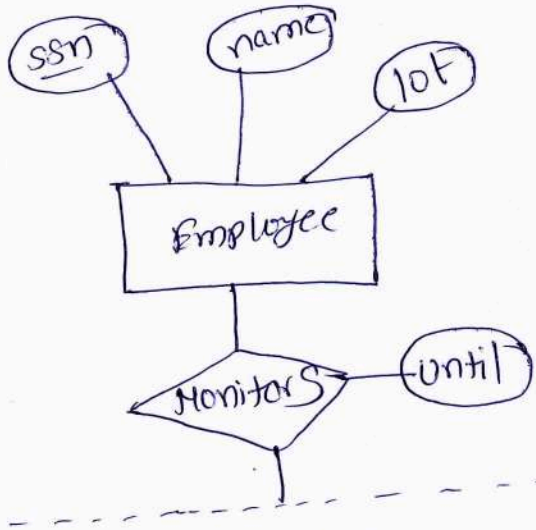
→ All Employees and do not care about the attributes specific to the subclasses are handled easily using the Employees relation.

→ who are neither hourly employees, nor contract Employees, since there is no way to store such employee.

Translating ER Diagrams With aggregation:-

1. We have to record the descriptive attributes (in our example, since) of the sponsors relationship.

2. Not every sponsorship has a monitor, and thus some(pid, did) pairs in the sponsors relation may not appear in the monitors relation).



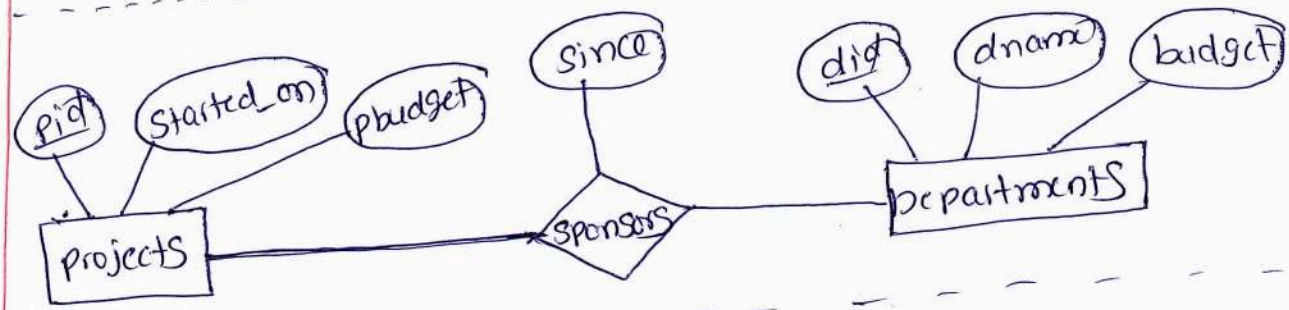
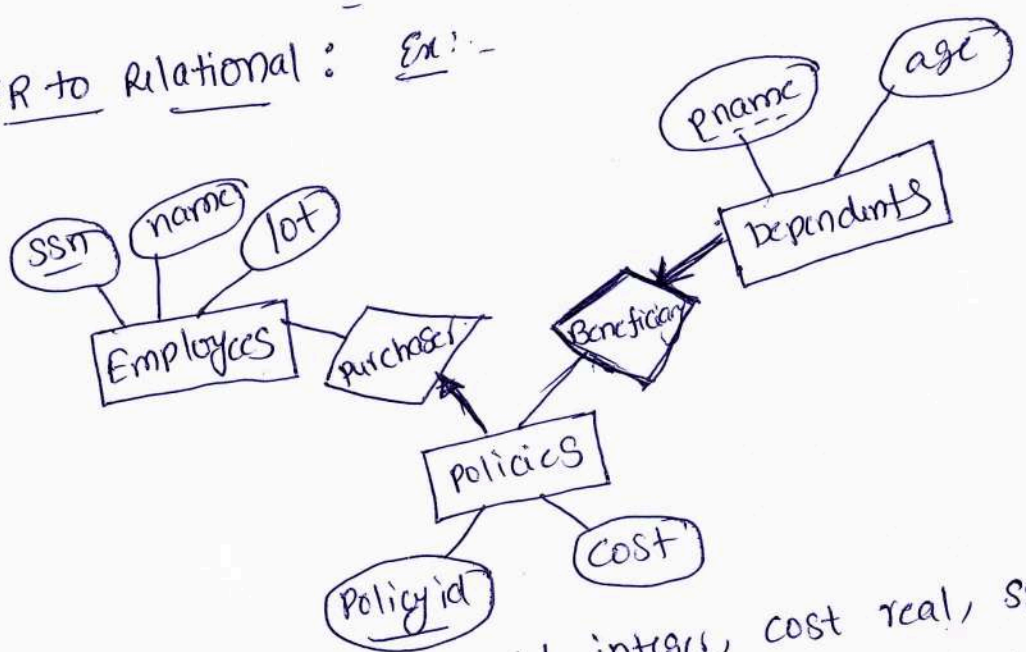


fig: Aggregation

ER to Relational : Ex:-



→ Create table Policies (Policyid integer, cost real, SSN char(11) NOT NULL primary key (Policyid), foreign key (SSN) References Employees ON DELETE CASCADE);

→ Create table Dependents (pname char(20), age integer, policyid int, primary key (pname, policyid), foreign key (policyid) References Policies ON DELETE CASCADE);

* Introduction to views:-

A view in SQL is a logical subset of data from one or more tables.

view is used to restrict data access.

→ Views in SQL are considered as a virtual table. A view also contains rows and columns.

→ To create the view, we can select the fields from one or more tables present in the database.

→ A view can either have specific rows based on certain condition or all the rows of a table.

Syn:- sample table:-

student_detail:-

Stu_ID	Name	Address
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugaram

Student_Marks:

Stud_ID	Name	Marks	Age
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1) Creating view:- A view can be created using the create view statement. We can create a view from a single table or multiple tables.

Syn:-

```
Create view view_name AS
SELECT column1, column2...
from table_name
WHERE condition;
```

2) Creating view from a single table:-

We can create a view named DetailsView from the table Student_Detail.

Query:-

```
Create view DetailsView AS
Select name, Address
from Student_Details
Where Stu_ID < 4;
```

We can query the view to view the data.

```
Select * from DetailsView;
```

O/P:-

Name	Address
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3) Creating view from multiple tables:-

View from multiple tables can be created by simply include multiple tables in the Select statement.

A view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:- Create view MarksView AS
Select Student_Detail.Name, Student_Detail.Address,
Student_Marks.Marks
from Student_Detail, Student_Mark
where student_Detail.Name = Student_Marks.Name;

To display data of view MarksView;

```
Select * from MarksView;
```

Name	Address	Marks
Stephan	Delhi	97
kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting view:- A view can be deleted using the DROP view statement.

Syn:- `Drop view view_name;`

Ex:- If we want to delete the view MarksView.

```
Drop view MarksView;
```

5. Update a view:- Update command for view is same as for tables.

Syn:- `Update view_name Set=value where condition;`

Types of view:- There are two types of view -

→ simple view
→ complex view

simple view	complex view
created from one table	created from one or more table
Does not contain functions	contain functions
Does not contain groups of data.	contains groups of data

* Altering / Destroying Tables And views:-

If we decide that we no longer need a base table and want to destroy it (i.e; delete all the rows and remove the table definition information), we can use the DROP TABLE command.

Ex:- DROP TABLE Students RESTRICT destroys the Students table unless some view or integrity constraint refers to Students; if so, the command fails. If the keyword RESTRICT is replaced by CASCADE, Students is dropped and any referencing views or integrity constraints are (recursively) dropped as well; A view can be dropped using the DROP view command, which is just like DROP TABLE.

ALTER TABLE modifies the structure of an existing table.
To add a column called maiden_name to Students.

```
ALTER TABLE Students
  ADD COLUMN maiden-name char(10);
```

The definition of Students is modified to add this column, and all existing rows are padded with null values in this column. ALTER TABLE can also be used to delete columns and add or drop integrity constraints on a table;

Drop table: Drop table <tablename>;

Drop view: Drop view view-name;