Database Management System

Nitish Sharma

1

Introduction

Database is a collection of related data.

E.g. Set of student information.

Database Management System:

It is a software used to manage and access the DB files in more efficient way.

2

DBMS

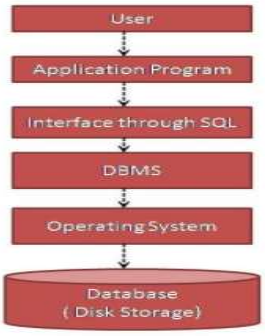
- Collection of interrelated data.
- Collection of set of programs to access the data.
- It acts as a layer of abstraction between the application programs and the file system.
- DBMS provides an environment that is both convenient and efficient to use.
- For interacting with the DBMS we use a Query language called Structured Query Language (SQL)

3

Applications

- **Banking**: all transactions
- **Airlines**: reservations, schedules
- **Universities**: registration, grades
- **Sales**: customers, products, purchases
- **Manufacturing**: production, inventory, orders, supply chain
- **Human resources**: employee records, salaries, tax deductions

4

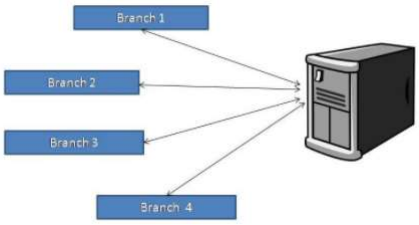


5

Types of Databases

1. Centralized Database

In Centralized database system, all data is stored at a single site. It offers a great control in accessing and updating data. However failure chances are high because the system depends on the availability of resources at the central site.



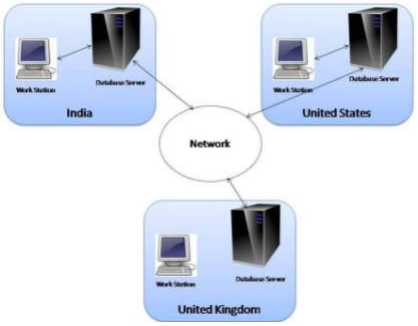
6

2. Distributed Database

In Distributed Database system, the database is stored on several computers.

Computers in a distributed system may communicate with one another through internet/intranet/telephone lines etc.

Most of the distributed systems will be geographically separated and managed. Distributed databases can also separately be administered.



7

Services provided by a DBMS

1. Data management
2. Data definition
3. Transaction support
4. Concurrency control
5. Recovery
6. Security and integrity
7. Facilities to import and export data
8. user management
9. backup
10. performance analysis
11. logging
12. audit
13. physical storage control

8

Users of a DBMS

1. Database Administrator (DBA)
- DBA takes care of the administrative tasks of DBMS.
- Monitoring database performance.

2. Database designers
- Database designers design the database components.

3. Application programmers
- Application programmers write programs to **access/insert/update/delete** data from/to database by making use of the various database components.

4. End users
- End users use DBMS

9

Advantages of a DBMS

- Data independence
- Reduced data redundancy
- Increased security
- Better flexibility
- Effective data sharing
- Enforces integrity constraints
- Enables backup and recover

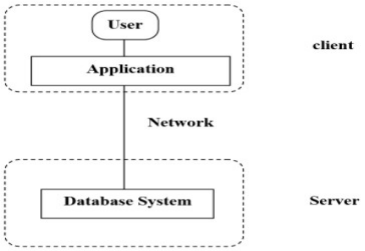
10

DBMS Architecture 2-tier, 3-tier

2 tier architecture:

Two tier architecture is similar to a basic **client-server** model.

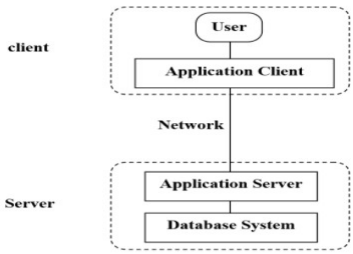
- The application at the client end directly communicates with the database at the server side. **API's like ODBC, JDBC** are used for this interaction.
- The server side is responsible for providing query processing and transaction management functionalities.
- An advantage of this type is that maintenance and understanding is easier, compatible with existing systems. However this model gives poor performance when there are a large number of users.



11

3 tier architecture:

- There is another layer between the client and the server.
- The client does not directly communicate with the server.
- Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place.
- This intermediate layer acts as a medium for exchange of partially processed data between server and client.
- This type of architecture is used in case of large web applications.



12

3 tier architecture:

Advantages:

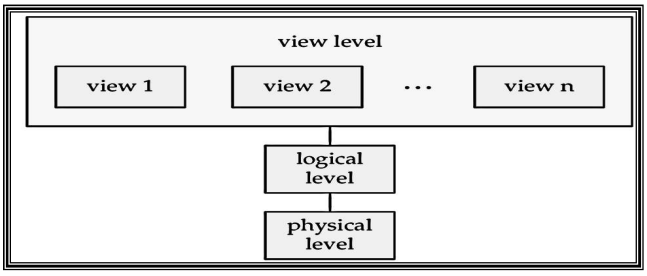
- **Enhanced scalability** due to distributed deployment of application servers. Now, individual connections need not be made between client and server.
- **Data Integrity** is maintained. Since there is a middle layer between client and server, data corruption can be avoided/removed.
- **Security** is improved. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

Disadvantages:
Increased complexity of implementation and communication. It becomes difficult for this sort of interaction to take place due to presence of middle layers.

13

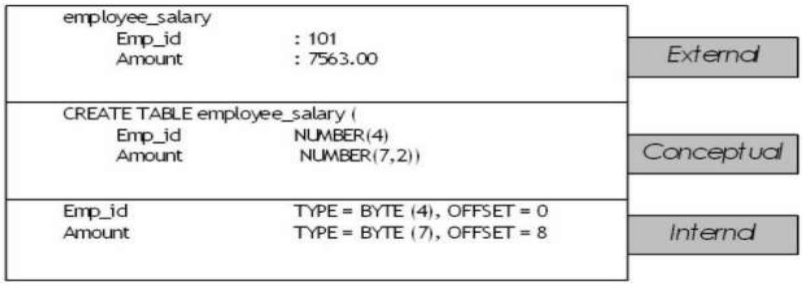
Levels of Abstraction

- Physical level: describes how a record is stored.
- Logical level: describes data stored in database, and the relationships among the data.
- View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes



14

An example of the three levels



15

Data Models

- A collection of tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model
- Relational model

16

ENTITY RELATIONSHIP DIAGRAMS

(A) Diagramatic Representation of DB Design

(B) High level DB design

(C) DB Design Steps :-

High Level DB Design

1. Requirements Gathering
2. Design [ER Model]
   - Tables
   - Relationship b/w Tables

Low Level DB Design

3. Convert ER Model to RDBMS Relation
   Apply Normalization
4. Implement RDBMS Relations [Create table] and Design Indexes [Create Index]
5. Design Security [Create View]
   [Access Rights based on end user need]

ER MODEL

ER Model is used to model the logical view of the system from data perspective.

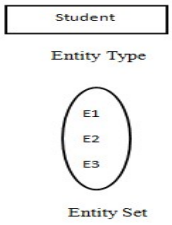
An **Entity** may be an object with a physical existence

– a particular person, car, house, or employee

or

it may be an object with a conceptual existence

– a company, a job, or a university course.



18

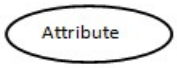
ER MODEL

- **Attribute**

Attributes are the **properties which define the entity type**.

For example, Roll_No, Name, DOB, Age etc.

In ER diagram, attribute is represented by an oval.



19

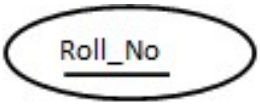
ER MODEL

- **Key Attribute**

The attribute which **uniquely identifies each entity** in the entity set is called key attribute.

For example, Roll_No will be unique for each student.

In ER diagram, key attribute is represented by an oval with underlying lines.



20

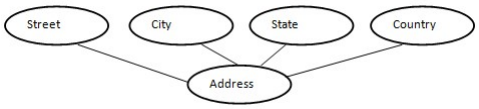
ER MODEL

- **Composite Attribute**

An attribute **composed of many other attribute** is called as composite attribute.

For example, Address attribute of student Entity type consists of Street, City, State, and Country.

In ER diagram, composite attribute is represented by an oval comprising of ovals.



21

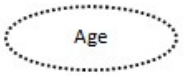
ER MODEL

- **Derived Attribute**

An attribute which can be **derived from other attributes** of the entity type is known as derived attribute.

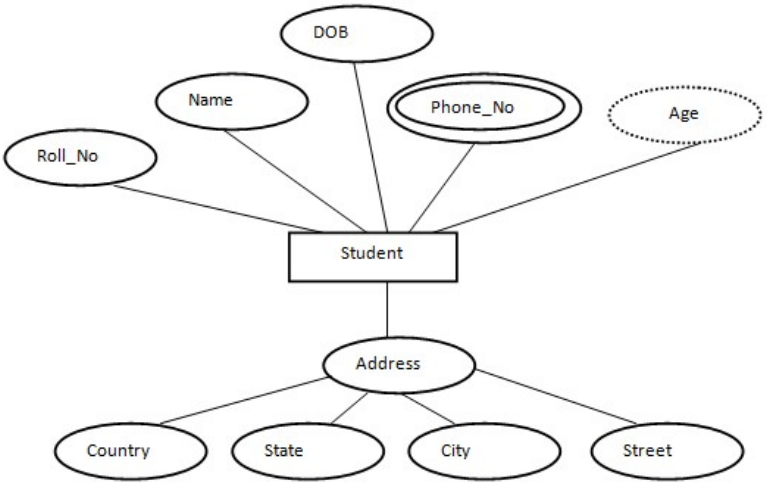
e.g.; Age (can be derived from DOB).

In ER diagram, derived attribute is represented by dashed oval.



22

ER MODEL



23

Relationship Type

A relationship type represents the **association between entity types**.

For example,

'Enrolled in' is a relationship type that exists between entity type Student and Course.

In **ER diagram**, relationship type is represented by a diamond and connecting the entities with lines.

24

Degree of a relationship set

The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.

1. Unary relationship.
2. Binary relationship.
3. Ternary relationship.
4. N-ary relationship.

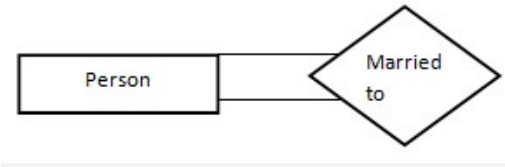
25

Degree of a relationship set

- **Unary Relationship**

When there is **only ONE entity set participating in a relation**, the relationship is called as unary relationship.

For example, one person is married to only one person.



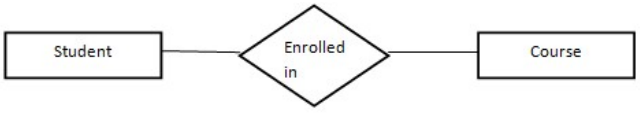
26

Degree of a relationship set

- **Binary Relationship**

When there are two entities set participating in a relation, the relationship is called as binary relationship.

For example, Student is enrolled in Course.

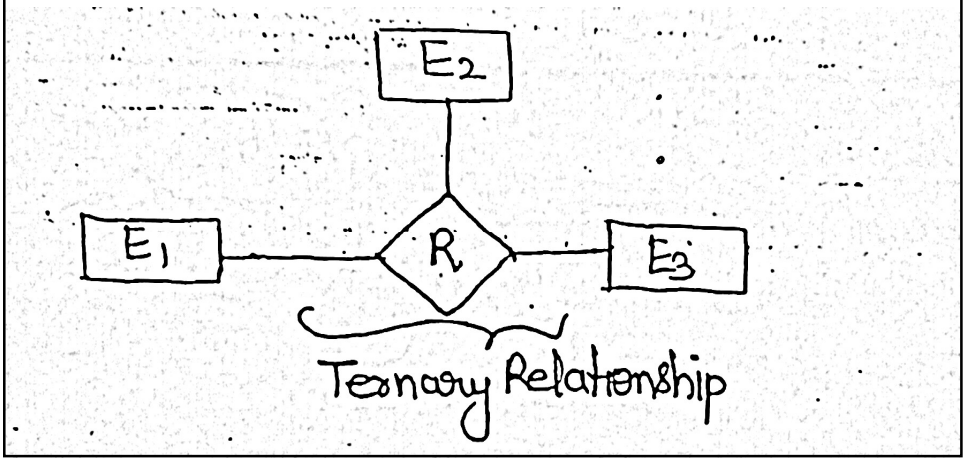


27

Degree of a relationship set

- **Ternary Relationship**

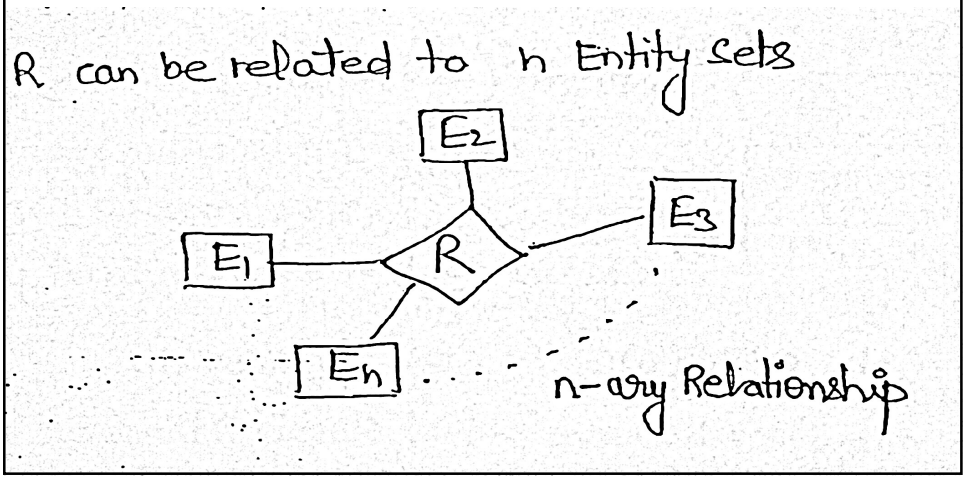
When there are three entities set participating in a relation, the relationship is called as ternary relationship.

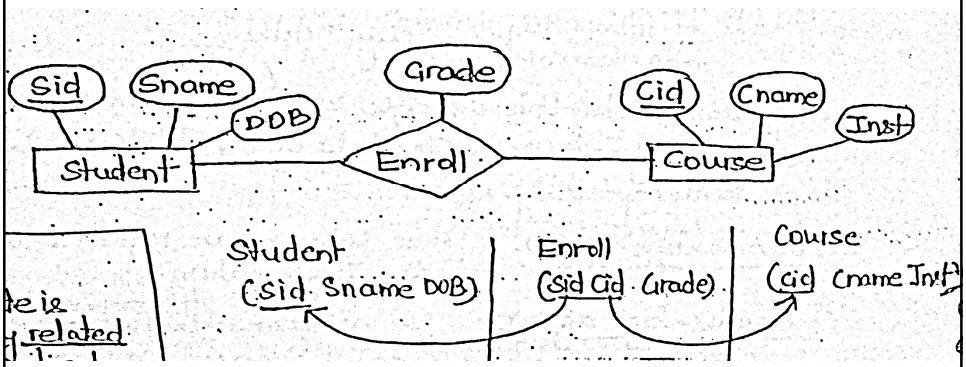


Degree of a relationship set

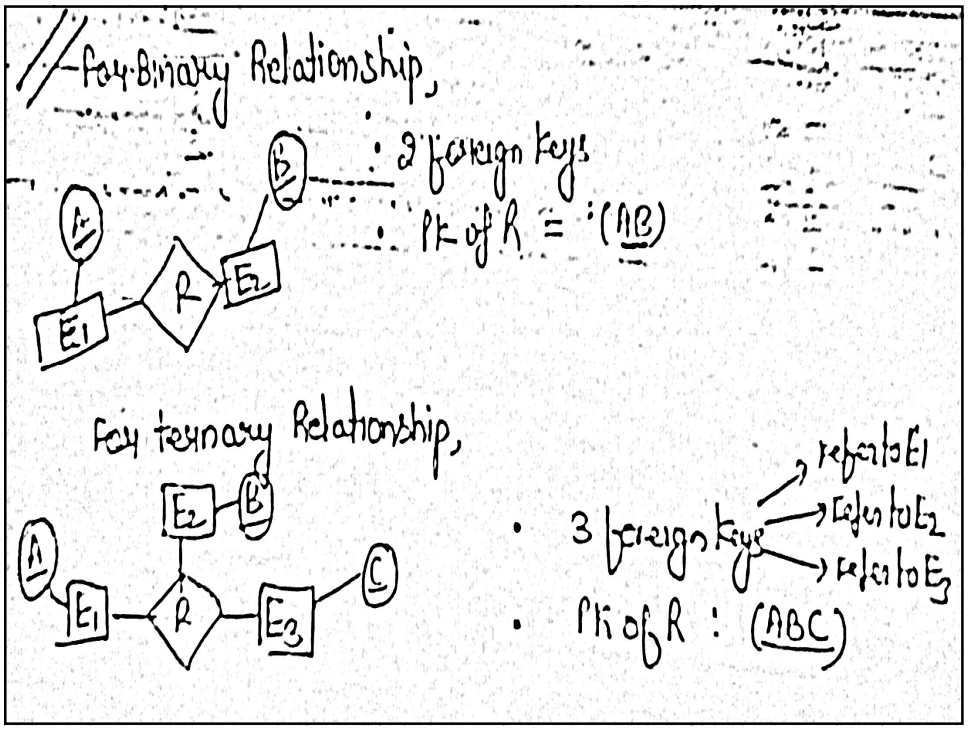
- **n-ary Relationship**

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.





30



Cardinality

The **number of times an entity of an entity set participates in a relationship** set is known as cardinality.

Cardinality can be of different types:

- One to One
- One to many
- Many to Many

32

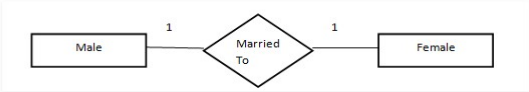
Cardinality

- **One to One**

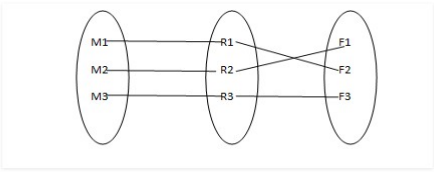
When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one.

Let us assume that a male can marry to one female and a female can marry to one male.

So the relationship will be one to one.



Using Sets, it can be represented as:



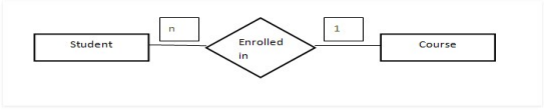
33

Cardinality

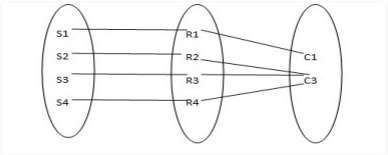
- **many to one:**

When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.

Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1.



Using Sets, it can be represented as:



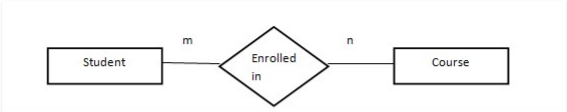
34

Cardinality

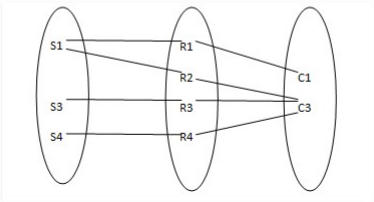
- **many to many:**

When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many.

Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



Using sets, it can be represented as:

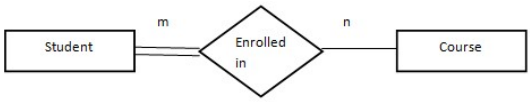


35

Participation Constraint

Participation Constraint is applied on the entity participating in the relationship set.

- **Total Participation:** Each entity in the entity set **must participate** in the relationship. Total participation is shown by double line in ER diagram.
- **Partial Participation:** The entity in the entity set **may or may NOT participat**e in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.



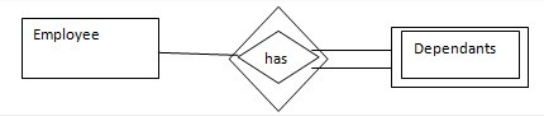
36

Weak Entity Type

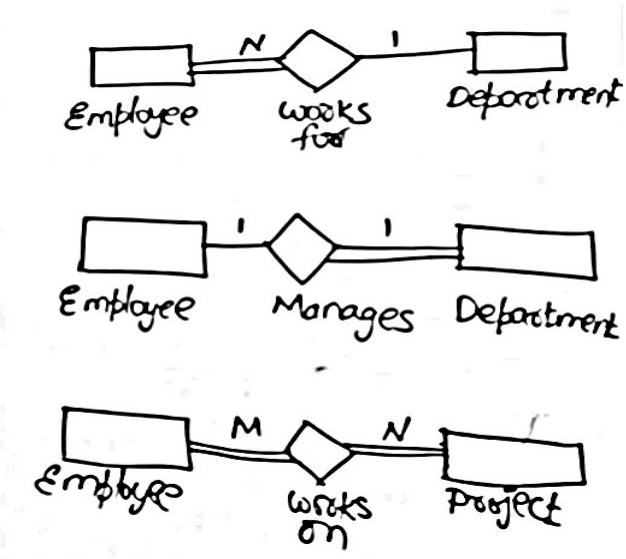
Some entity type for which key attribute can't be defined. These are called Weak Entity type.

- For example,

A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependents don't have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependant.

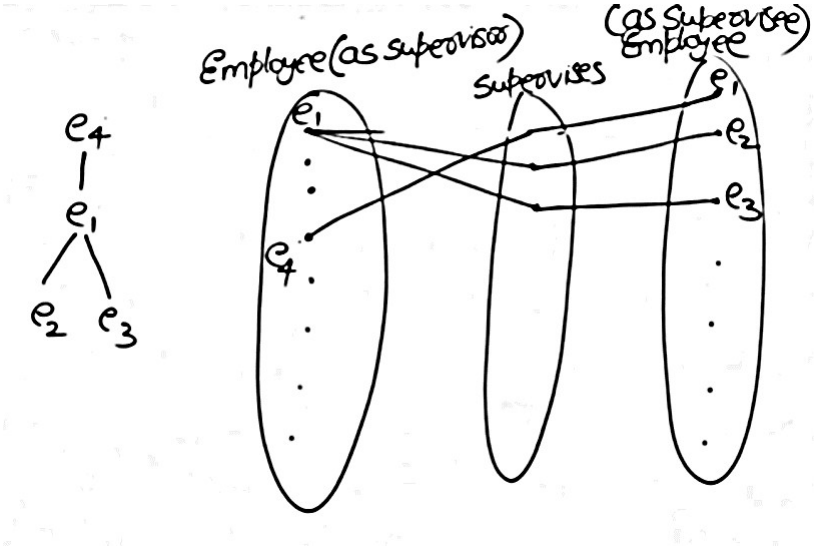


37



38

Recursive Relationships



39

Extended Entity-Relationship Model

The Extended Entity-Relationship Model is a **more complex** and **high-level model** that extends an E-R diagram to include more types of **abstraction**, and clearly express various **constraints**.

All of the concepts contained within an E-R diagram are included in the EE-R model, along with additional concepts that cover more **semantic** information.

These additional concepts include generalization, specialization etc.

40

Extended Entity-Relationship Model

Generalization is a process of defining generic entity from set of specialized entities.

i.e., by using subset entities we are developing a generic entity.

It follows **Bottom-Up** approach.

For example,

The entities **CAR** and **TRUCK** have similar properties and attributes, so they could be generalized into a superclass of **AUTOMOBILE**.

41

Extended Entity-Relationship Model

Specialization is a process of defining two or more sub-types of a super-type forming super-type, sub-type relationship by using generic entity.

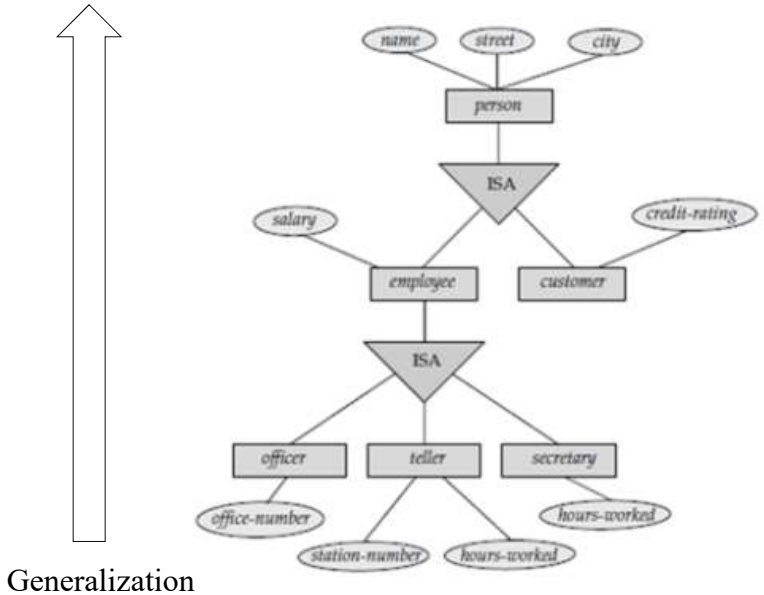
It is a complete reverse process of Generalization technique.

i.e., Specialization follows a **Top down** approach.

In Specialization sub-type is formed based on some different characteristics such as attributes or relationships.

42

Specialization

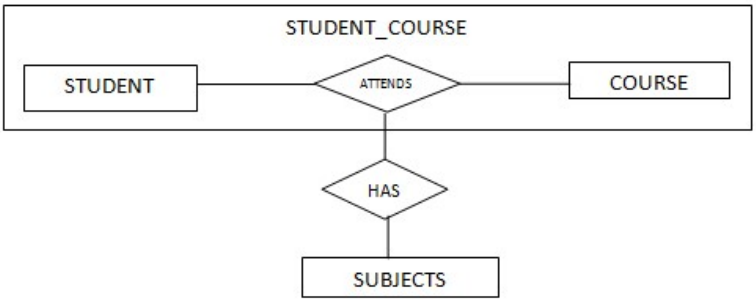


Generalization

43

Aggregation

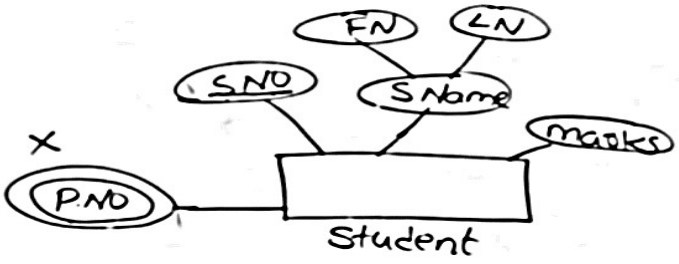
Aggregation is a process when relation between two entity is treated as a single entity.



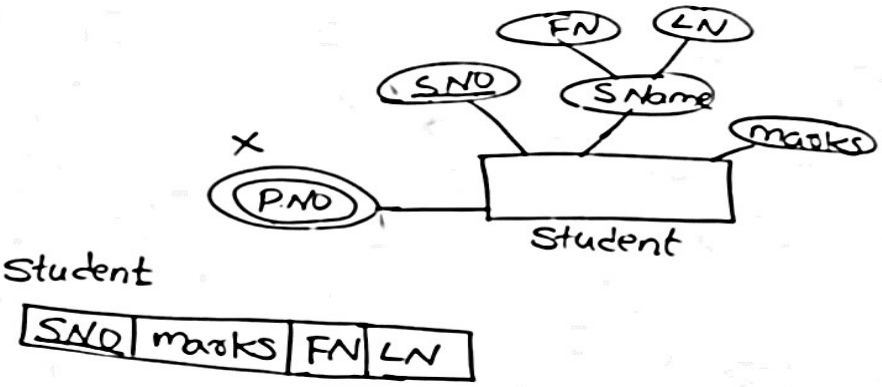
44

CONVERSION OF ER MODEL TO RELATIONAL MODEL

45

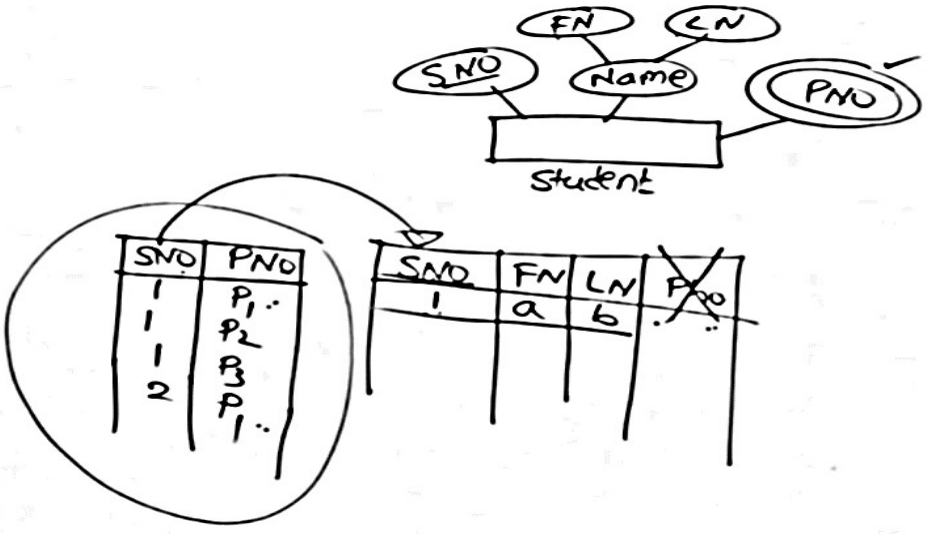


46



Here,
composite attribute is represented
but, multivalued attribute is represented in different relation.

47



Separate relation is made for multivalued attributes by using foreign key.

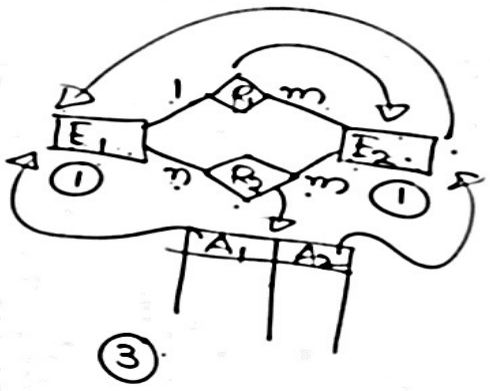
48

Gate 05

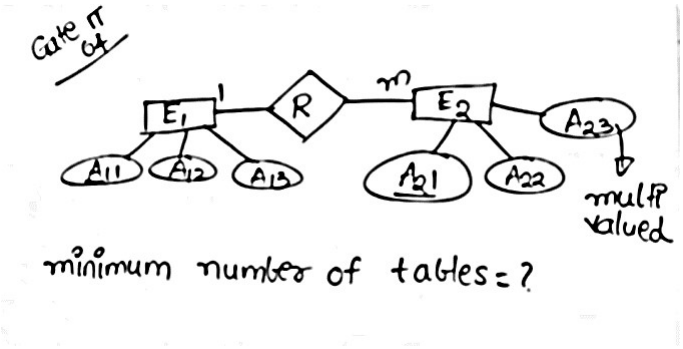
E_1, E_2 – two entities

R_1, R_2 are two relationships b/w E_1 and E_2. R_1 is one-to-many and R_2 is many to many. R_1 and R_2 doesnot have any attributes of their own. what is the minimum number of tables.

49

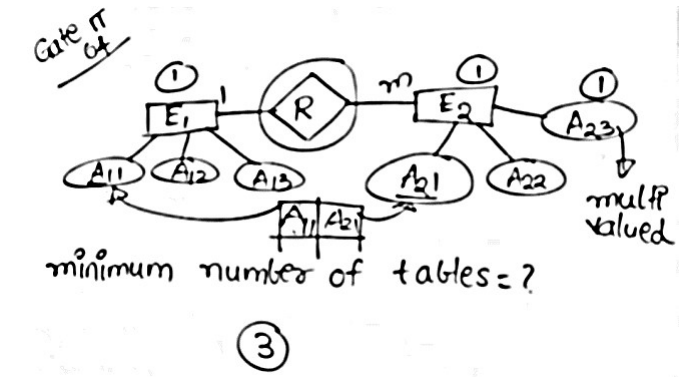


50



minimum number of tables = ?

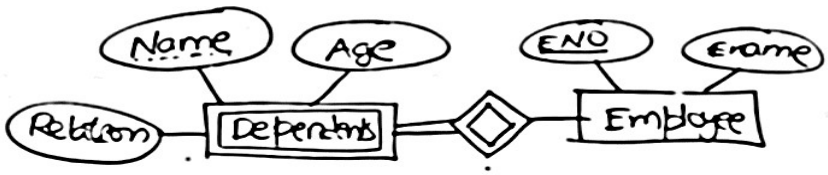
51



minimum number of tables = ?

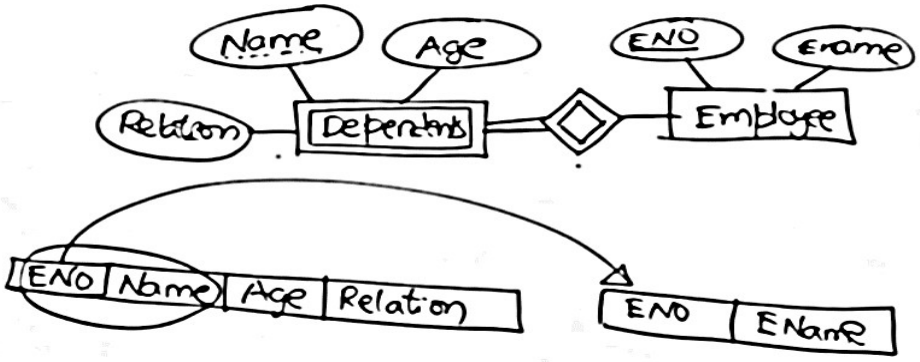
3

52



Weak entity set: Dependents (doesn't have primary key)
Strong entity set: Employee

53



Weak entity set: Dependents (doesn't have primary key)
Strong entity set: Employee

54

ER MODEL	Relational model
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or relation)
M:N relationship type	relationship "relation" + 2 FKS
n-ary relationship type	relationship "relation" + n FKS
Simple attribute	Attribute
Composite attribute	Set of Simple Component attr
Multivalued attribute	Relation and Foreign key
Value set	Domain
Key attribute	Primary key.

55

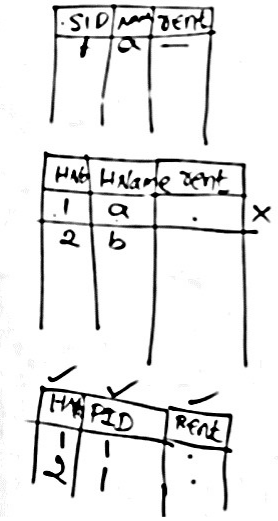
GateIT-05

lodging is a many to many relationship. Rent, payment to be made by person(s) occupying different hotel rooms should be added as an attribute to

a) Hotel b) lodging c) Person
d) none.

56

GateIT-05



lodging is a many to many relationship. Rent, payment to be made by person(s) occupying different hotel rooms should be added as an attribute to

a) Hotel b) lodging c) Person d) none.

57

GateI2

Given the basic ER and relational models, which of the following is INCORRECT?

(a) An attribute of an entity can have more than one value.

(b) An attribute of an entity can be composite.

(c) In a row of a relational table, an attribute can have more than one value.

(d) In a row of a relational table, an attribute can have exactly one value or a NULL value.

58

Gate12

Given the basic ER and relational models, which of the following is INCORRECT?

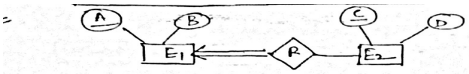
(a) An attribute of an entity can have more than one value.

(b) An attribute of an entity can be composite.

✓(c) In a row of a relational table, an attribute can have more than one value.

(d) In a row of a relational table, an attribute can have exactly one value or a NULL value.

59



a) How many Relational tables for ERD which satisfy 1NF?

a) 1 b) 2 c) 3 d) 4

Here, it is right to combine R and E_2.

But here E_1 can also be merged into RE_2 because the data redundancy is allowed in INF and there is total participation from E_1 to R.

So, E_1RE_2 (A B C.K. C D)

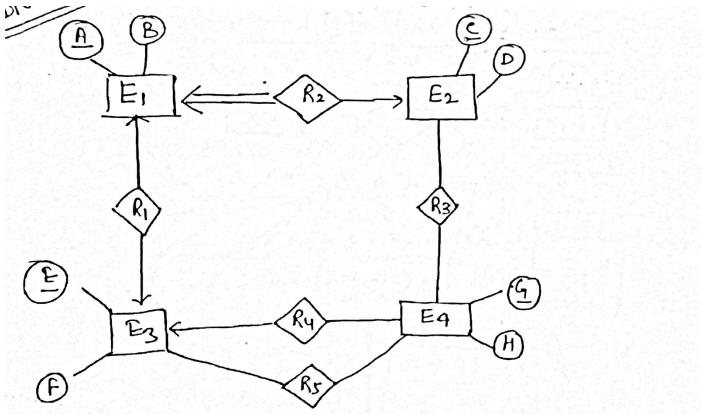
$C \rightarrow DA$

$A \rightarrow B$: data Redundancy (but redundancy is allowed in INF)

Here partial participation can't be failed as in EPD, we are given total participation b/w E_1 & R.

∴ only 1 table ∴ (a).

60



1) How many min Relational tables for given ERD?

61

1) How many min Relational tables for given ERD?

i) E_1 — R_2 — E_2 (1:1; TP, PP)

(ii) E_1 (PP) — E_3 (1:1, P)

(iii) E_3 — R_4 — E_4 (M:1)

(iv) R_3 (M:N) — so seperate table

(v) R_5 (M:N) — seperate table

So, 5 tables.

62

2) How many foriegn keys required for given EPD, RDBMS design?

$E_1 R E_2$ (A B C D)

$R_1 E_3$ (E F A)

$R_4 E_4$ (G H E)

R_3 (A G) #FKs = 6

R_5 (E G)

3) Total # of Attributes in Relational tables of given EPD?

4 + 3 + 3 + 2 + 2 = 14.

63

Database Language

64

Database Language

Database language are categorized as:

1. Data Definition language (DDL)
2. Data Manipulation language (DML)
3. Data Control language (DCL)
4. Transaction control language (TCL)

65

Database Language

1. Data Definition Language

Data Definition Language (DDL) statements are used to classify the database **structure** or **schema**.

It is a type of language that allows the DBA or user to depict and name those entities,
attributes,
and relationships that are required for the application

along with any **associated integrity** and **security constraints**.

66

Here are the lists of tasks that come under DDL:

CREATE – used to create objects in the database

ALTER – used to alters the structure of the database

DROP – used to delete objects from the database

TRUNCATE – used to remove all records from a table, including all spaces allocated for the records are removed

RENAME – used to rename an object

67

Database Language

2. Data Manipulation Language

A language that offers a **set of operations** to support the fundamental data manipulation operations on the data held in the database.

Data Manipulation Language (DML) statements are used to **manage** data within schema objects.

68

Here are the lists of tasks that come under DML:

INSERT – It inserts data into a table
UPDATE – It updates existing data within a table
DELETE – It deletes all records from a table, the space for the records remain
CALL – It calls a PL/SQL or Java subprogram
EXPLAIN PLAN – It explains access path to data
LOCK TABLE – It controls concurrency

69

Database Language

3. Data Control Language

- The Data Control Language (DCL) is used to control **privilege** in Database.
- To perform any operation in the database, such as for creating tables, sequences or views we need privileges.

Privileges are of two types:
System – creating a session, table etc. are all types of system privilege.
Object – any command or query to work on tables comes under object privilege.

DCL is used to define two **commands**:
Grant – It gives user access privileges to a database.
Revoke – It takes back permissions from the user.

70

Database Language

4. Transaction Control Language (TCL)

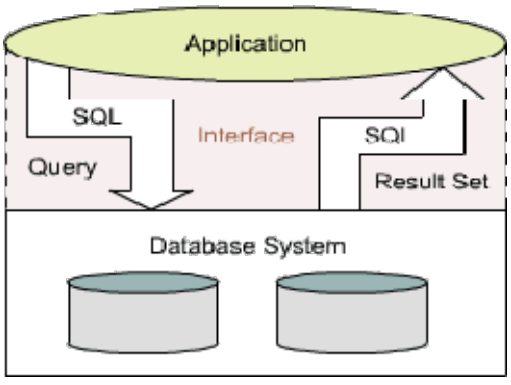
Transaction Control statements are used to run the changes made by DML statements.

It allows statements to be grouped together into logical transactions.

COMMIT – It saves the work done
SAVEPOINT – It identifies a point in a transaction to which you can later roll back
ROLLBACK – It restores database to original since the last COMMIT

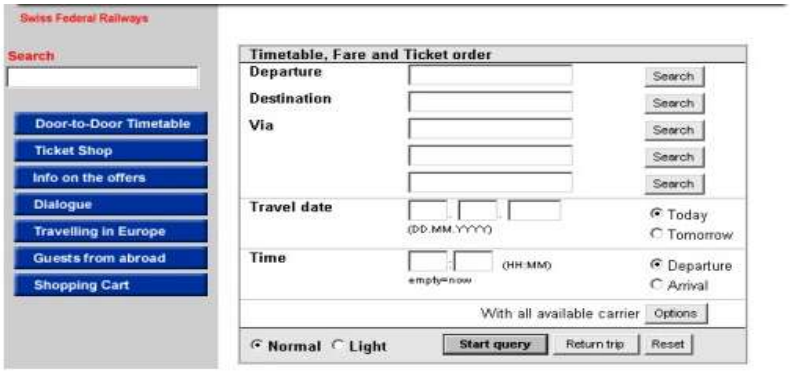
71

Database Interface



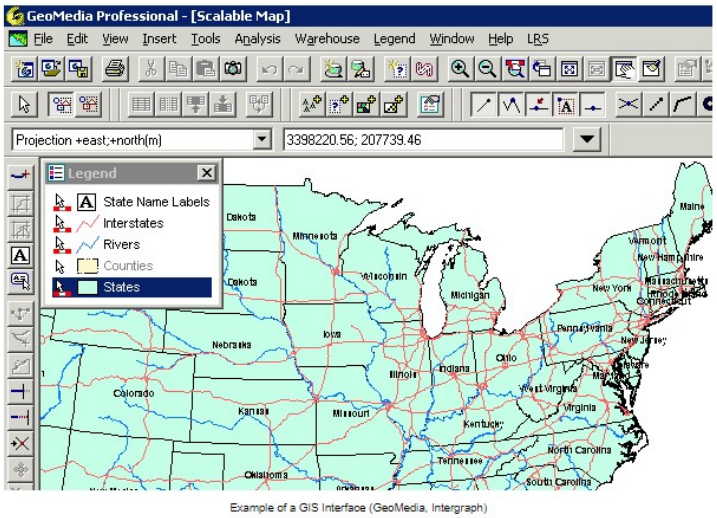
72

Database Interface



73

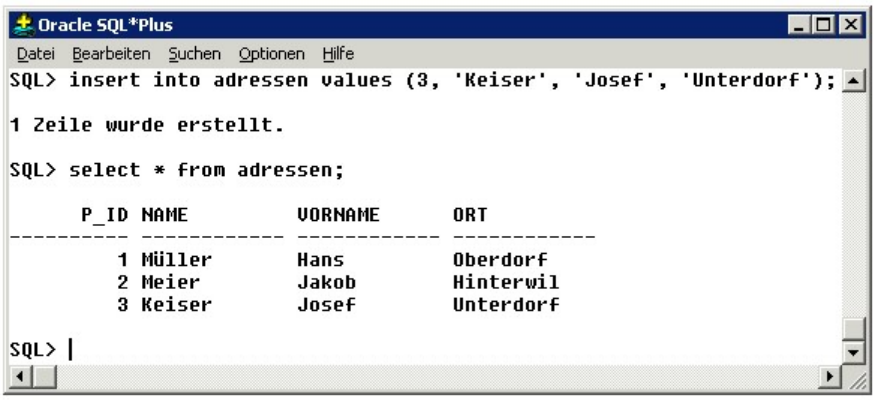
Database Interface



Example of a GIS Interface (GeoMedia, Intergraph)

74

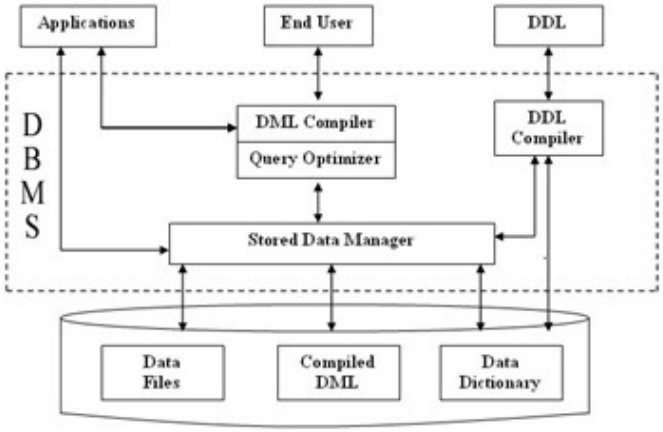
Database Interface



Example of a Text-base User Interface

75

DBMS Structure



76

SQL CONSTRAINTS

77

SQL Constraints

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into following two types,

- **Column level constraints :** limits only column data
- **Table level constraints :** limits whole table data

Constraints are used to make sure that the **integrity of data** is maintained in the database. Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

78

SQL Constraints

1. NOT NULL Constraint

- NOT NULL constraint restricts a column from having a NULL value.
- Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.

```
CREATE table Student(s_id int NOT NULL, Name varchar(60), Age int);
```

2. UNIQUE Constraint

- UNIQUE constraint ensures that a field or column will only have unique values.
- A UNIQUE constraint field will not have duplicate data.

```
CREATE table Student(s_id int NOT NULL UNIQUE, Name varchar(60),
Age int);
```

79

SQL Constraints

3. Primary Key Constraint

- Primary key constraint uniquely identifies each record in a database.
- A Primary Key must contain unique value and it must not contain null value.

```
CREATE table Student (s_id int PRIMARY KEY, Name varchar(60) NOT
NULL, Age int);
```

80

SQL Constraints

4. Foreign Key Constraint

Customer_Detail Table :

c_id	Customer_Name	address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

Order_Detail Table :

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

CREATE table Order_Detail(order_id int PRIMARY KEY, order_name varchar(60) NOT NULL, c_id int **FOREIGN KEY** REFERENCES **Customer_Detail**(c_id));

81

SQL Constraints

5. Check Constraint

- CHECK constraint is used to restrict the value of a column between a range.
- It performs check on the values, before storing them into the database.
- Its like condition checking before saving data into a column.

create table Student(s_id int NOT NULL **CHECK(s_id > 0)**, Name varchar(60) NOT NULL, Age int);

82

INTEGRITY CONSTRAINTS FOR RDBMS

83

Relational DBMS designed by Dr. E.F. Codd

Dr. E.F. Codd is an IBM researcher who first developed the relational data model in 1970.

In 1985, Dr. E.F. Codd published 12 rules to design RDBMS

84

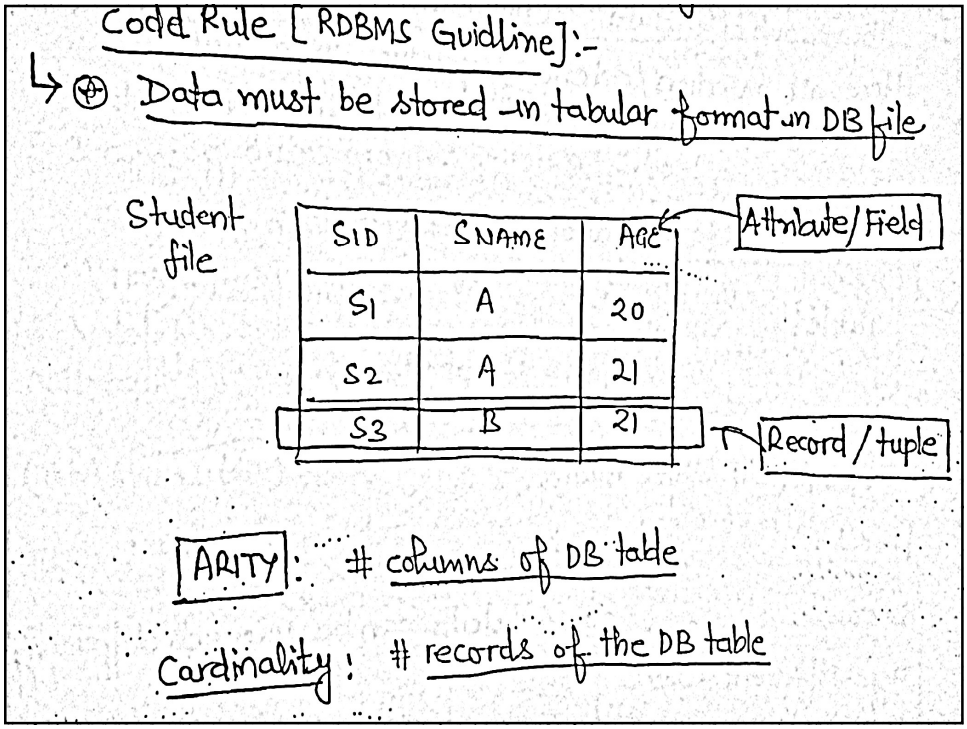
Integrity Constraints:

It is a condition that can be applied on a database schema to restrict the data according to the need.

1. Entity Integrity Constraint.
   - No Primary Key value can be null.

2. Referential Integrity Constraint**.**
   - It is a condition which ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation. (Foreign Key concept)

85



↳ Relation Schema:

DB. Table Definition (Abstract details of the table)

eg. Student (sid sname Age) — schema

& key constraints.

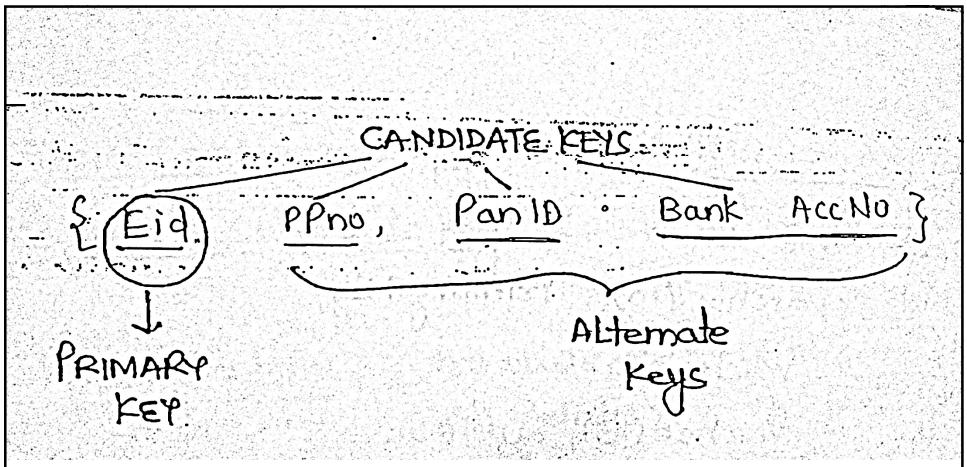
↳ Relational Instance: (SNAPSHOT)

It is the Record Set of DB table at particular instance of time.

Properties of Relational Tables:

1. Values are atomic.
2. Column values are of same kind.
3. Each row is unique.
4. Each column has a unique name.
5. The sequence of rows/columns is insignificant.

88



Candidate key is a minimal set of attributes used to differentiate records of the relation uniquely.

89

- Any one of the candidate key must be chosen as Primary key.

↳ Primary key –

Any one candidate key with field values must be "NOT NULL".

NULL: Unknown values / UnExisted values

↳ Primary key Constraints –

1) All candidate key constraints are primary key constraints.
2) NULL values are not allowed over primary key fields.
3) Almost only one primary key is allowed for any DB table.

↳ Alternative key -

All candidate keys of the Relation except Primary key.

→ Alternative Key Constraints -

1) All candidate key constraints are alternative key constraints.

2) NULL values are allowed over alternative key fields.

3) More than one alternative key allowed for relation.

All the key constraints are under

INTEGRITY CONSTRAINTS

↙ ↘

CORRECTNESS | CONDITIONS

(Conditions which are followed to achieve correct data are called integrity constraints.)

92

→ (A) RDBMS relation must consist atleast one candidate key and atleast one candidate key with NOT NULL values.

```
Create Table R
( A   int   Unique  NotNull,
  B   int   Unique
);
```

R is a relation, although it does not contain Primary key but has atleast one unique key with NOT NULL constraint.

93

↳ TABLE SCHEMA FOR EMP :-

```
Create table Emp
( Eid     varchar(10), Primary key,
  Ename   varchar(30),
  PPno    varchar(15), UNIQUE,
  PanID   varchar(10), UNIQUE NOT NULL,
  Bank    Varchar(10),
  AccNo   Integer(10),
  UNIQUE  (Bank, AccNo)
);
```

- **Simple Candidate key:** Candidate key with only one attribute.
- **Compound Candidate key:** Candidate key with atleast two attributes.
- **Prime Attribute:** Attribute belongs to some candidate key of Relation. eg. Eid, PPNo, PanID, Bank, ACCNo etc.
- **Non Prime Attribute:** Attribute not belongs to any candidate key of relation. eg. Ename.

95

Example:- R (A B C D E F)

Candidate keys = { A, BC, CD }

then

Prime Attribute set of R = { A, B, C, D }

Non Prime Attribute set of R = { E, F }

96

↳ ex ① Student (sid, sname, age)

SID : candidate key

So, here no two records in the DB will have same SID.

SID, SNAME : Candidate key ✗

because it is not minimal set of Attributes to uniquely identify records in DB.

97

↳ Super key

Set of attributes used to differentiate records uniquely which may or may not be minimal attribute set.

eg. R(sid sname age)

if we say {sid} : Candidate key

then sid

sid sname

sid age

sid name age

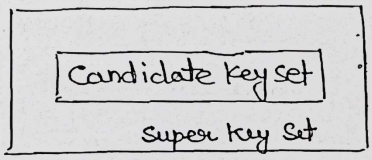
} Super Keys

means no 2 records have same (sid age)

98

④ All candidate keys are super keys but some super keys may not be candidate keys.

Super key set is a superset of candidate key set



X is superkey of R
iff some subset of X must be a candidate key

(OR)

→ Minimal Super key is a candidate key.

99

Problem:-

R(A, B, C, D)

A: Candidate key

How many superkeys in R?

Sol:- A combine with any subset of (B,C,D) will become a superkey.

subsets of (B,C,D) = 8 ($=2^3$)

superkeys of R = 8.

{ A; ABC
AB, ABD
AC, ACD
AD, ABCD }

Q. R(A B C D E)

{AB, BC} : Candidate key

How many superkeys are in R?

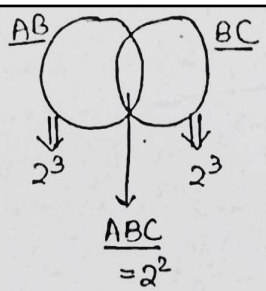
Using set theory,

Superkeys for cand key AB = AB set

Super Keys for cand Key BC = BC set

Super Key Common in AB, BC = having ABC

= # Super Key for ABC (i.e. intersection)



ABC [DE]

$\downarrow$

2^2

superkeys = 8 + 8 − 4 = 12 Superkeys.

method2:-

# of Superkeys = [# of superkeys among prime] * $2^{\text{NO. of None Prime}}$

$\downarrow$

{AB, BC, ABC}

# of Non primes = {D, E}

$= 3 * 2^2 = 12$ superkeys.

3) all attributes form one cand key

$R\{\underline{A_1 A_2 A_3 \ldots A_n}\}$ → cand key

Only 1 superkey.

Minimum possible superkeys for a relation come in that case when all attributes together form a cand. key.

103

→ Foriegn key [Referential key]
(used to relate database tables)

ex

Student (sid, sname, age)

sid	sname	age
S_1	A	20
S_2	B	21
S_3	B	21
S_4	C	18

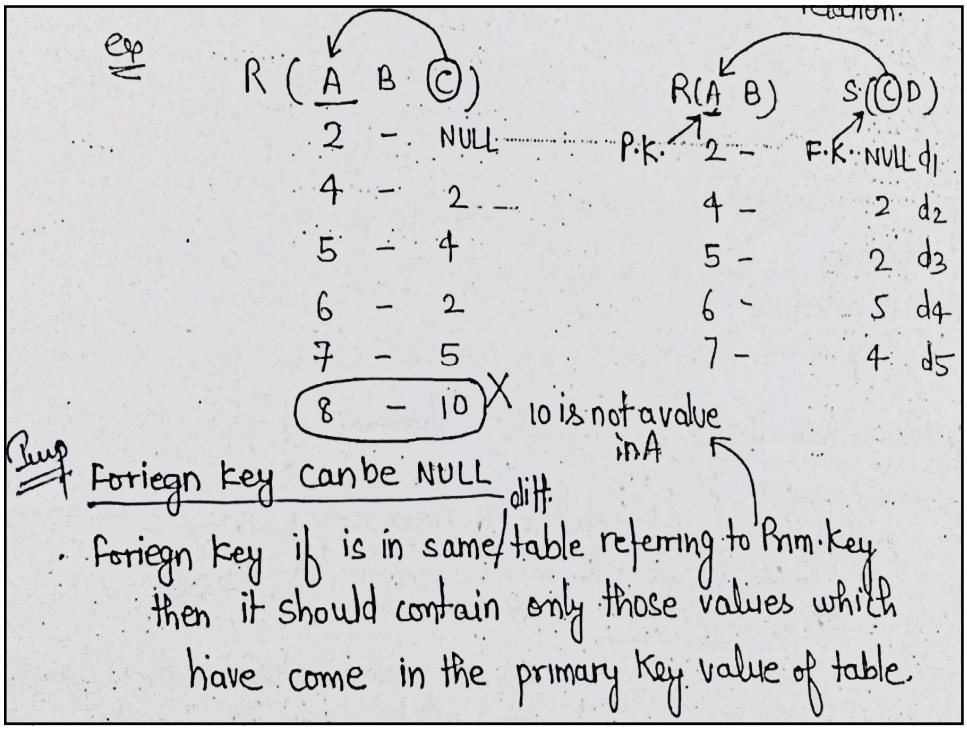
All students

Enroll (sid, cid, fee)

sid	cid	fee	
S_1	C_1	–	
S_1	C_2	–	
S_4	C_2	–	
S_{10}	C_1	–	not allowed

Students enrolled atleast one course

Foriegn key: Set of attributes used to reference primary key/ Alternative key of the same relation/ some other relation.



— The table which contains foriegn key is known as Referencing table.

— The table which is referred by foriegn key is known Referred table.

↳ Referential Integrity Constraint [Foriegn key IC]

a) Referenced Relation (Parent Table) ↳ [Student]

→ i) Insertion :- No Violation

Inserting any value to parent table will not violate foriegn key condⁿ

→ ii) Deletion :- May cause Violation

[1] ON DELETE NO ACTION

If data used in Referencing table then Not allowed to delete data from Referenced Relation.

Eg. To delete S1 from Student (parent table) –

[2] ON DELETE CASCADE :-

Because of deletion of record from Referenced Relation, DBMS is forced to delete related records from Referenang relation also.

Here no requirement of 2 queries.

Both preserves integrity constraint.

[3] ON DELETE SET NULL :-

If FK field allowed to set NULL
then on deletion of referenced record sets
NULL value ∧ is assigned to related foreign key Field

Otherwise deletion from Referenced table
is not allowed

109

→ iii) Updation :- May Cause Violation

1] ON UPDATE NO ACTION

2] ON UPDATE CASCADE

3] ON UPDATE SET NULL

110

b) Referencing Relation (Child Table)
↳ [Enroll]

i) Insertion : May cause Violation

ii) Deletion : No Violation

iii) Updation : May cause Violation

(*) Because of Insertion & Updation, if violation is caused then insertion & updation are restricted.
No other solution is possible in case of violation.

111

→ Create table with Foreign key :-

```
CREATE table Student
   (sid  varchar(10)  Primary key,
    sname  varchar(30),
    age  integer(3)
   );

CREATE  table Enroll
   ( sid  varchar(10),
     cid  varchar(10),
     fee  integer,
     primary key (sid,cid),
     FOREIGN KEY (sid)
       References to student(sid)
```

112

RELATIONAL ALGEBRA

113

RELATIONAL ALGEBRA

Basic Operators		Derived Operators	
π	Projection	$\bowtie$	Join {using $\times, \sigma, \pi$}
σ	selection	$\cap$	Intersection {using $-$}
$\times$	Cross Product	/	Division {using $\times, -, \pi$}
$\cup$	Union		
$-$	set Difference		
ρ	Rename		

⊛ Projection (π)

↳ It is used to project required attributes.

$\pi_{\text{Attributes}}(R)$

eg

R

A	B	C
2	4	6
3	4	2
4	4	6
5	5	3

$\pi_{B,C}(R)$ →

B	C
4	6
4	2
5	3

} distinct records in Result

115

⊛ Selection (σ)

↳ It is used to select records based on specified conditions.

R

A	B	C
✗ 2	4	6
✗ 3	4	2
✓ 4	4	6
✓ 5	5	3

$\sigma_{A \geq 4}(R)$ →

A	B	C
4	4	6
5	5	3

$\sigma_{\text{Condn}}(R)$

116

⑷ Cross Product (X)

(Binary operator) → applied on 2 tables

R X S ⇒ results all attributes of R followed by all attributes of S

(and)

each record of R pairs with every record of S.

117

eg.

R

A	B	C
2	4	6
3	4	4
4	2	4

n distinct tuples

← X →

S

C	D
4	3
7	4

m distinct tuples

← Y →

RXS

A	B	C	C	D
2	4	6	4	3
2	4	6	7	4
3	4	4	4	3
3	4	4	7	4
4	2	4	4	3
4	2	4	7	4

nxm distinct tuples

← X + Y →

⑷ Cross Product is used to compare data of two tables.

⑷ The selection operator ($\sigma_c(R)$) is unary operator & thus can be used to select tuples from only one table. If we want to select tuples from two tables at a time (to compare etc.) we can apply ($\sigma_c(R \times S)$).

Q.

if $|R| = n$ $|S| = 0$.

then $|R \times S| = ?$

Sol: 0 tuples.

(#) R×S result empty record set if either R or S or both are empty record set.

eg: $R = \{2, 4, 6\}$ $S = \{\}$

than $R \times S = \{\}$ // Empty

SET OPERATORS

$\cup$: union

$-$: Minus

$\cap$: Intersection

To Apply $\cup$, $\cap$, $-$ b/w Relations R & S, Relations must be union compatible.

R & S are union compatible iff

1) no. of attributes of R & S must be same (and)

2) domain of i^{th} column of R must be same as domain of j^{th} column of S

domain (A): possible values accepted by Attribute 'A'.

eg

R

A	B
2	4
3	6
3	7

n distinct tuples

S

C	D
4	2
3	6
3	7

m distinct tuples

UNION:-

$$R \cup S \equiv \{ x \mid x \in R \ \vee \ x \in S \}$$

(OR)

$R \cup S$ (Resulted schema is same as R) $\equiv$ R

A	B
2	4
3	6
3	7
4	2

distinct tuples

means table name & name are stated as that of R.

Cardinality of $R \cup S = \{ \max(n,m) \text{ to } n+m \}$

INTERSECTION

$$R \cap S \equiv R - (R - S)$$

$$R \cap S \equiv \{ x \mid x \in R \wedge x \in S \}$$

(AND)

$R \cap S \equiv R$

A	B
3	6
3	7

distinct tuples

R (A, B) S (A, B)	Then $R \cap S \equiv R \bowtie S$

same col. & same Attr. name

Cardinality of $R \cap S = \{ 0 \text{ to } \min(n,m) \}$ tuples

MINUS

$R - S \equiv \{ X \mid X \in R \wedge X \notin S \}$

$R - S \equiv R$

A	B
2	4

cardinality of $R - S = \{0 \text{ to } n \text{ tuples}\}$

123

JOINS

1) Natural Join ($\bowtie$)
2) Conditional Join ($\bowtie_c$)
3) Outer Joins
   a) Left outer Join ($\bowtie$ | $\bowtie_c$)
   b) Right outer Join ($\bowtie$ | $\bowtie_c$)
   c) Full outer Join ($\bowtie$ | $\bowtie_c$)

124

④ Natural Join (⋈)

R ⋈ S Equal to

1) R × S

and 2) $\sigma_p(R \times S)$

P: Equality b/w common attributes of R & S

and 3) $\pi_{\text{distinct attr.}}(\sigma_p(R \times S))$

$$R \bowtie S \equiv \pi_{\underbrace{ABCD}_{\text{distinct attr.}}}(\sigma_{\underbrace{R.C=S.C}_{P}}(R \times S))$$

eg:

R

A	B	C
2	4	6
3	4	4
4	2	4

S

C	D
4	3
7	4

R ⋈ S

A	B	C	D
3	4	4	3
4	2	4	3

Cardinality of R ⋈ S ≡ {0 to n×m}

④ Cardinality of R ⋈ S = 0

i) when there are common attributes but the equality condition is not satisfied.

eg:

A	B	C
2	4	6
3	4	4
4	2	4

×

C	D
14	3
7	4

ii) when R × S = 0

(or) either R or S is Empty.

CONDITIONAL JOIN

$R \bowtie_c S \equiv \sigma_c (R \times S)$

→ instead of equality on we can define any condition.

eg

R

A	B	C
2	6	8
3	4	2
7	2	6

S

C	D
9	3
4	5
7	2

$R \underset{R.C>S.C}{\bowtie} S \equiv \sigma_{R.C>S.C} (R \times S)$

A	B	C	C	D
2	6	8	4	5
2	6	8	7	2
7	2	6	4	5

Cardinality : {0 to n*m tuples}

Outer Joins

a) Left Outer Join ($\bowtie$)

R ⋈ S : results R ⋈ S tuples, (and) tuples of R those have failed join condition. with entries of S as NULL.

eg

R

A	B
2	4
3	5

n

S

B	C	D
4	5	6
1	2	7
6	2	5

m

R ⋈ S

A	B	C	D
2	4	5	6
2	4	2	7
3	5	NULL	NULL

Cardinality : {n to n*m tuples}

b) Right Outer Join (⟖)

which are

R ⟖ S =

A	B	C	D
2	4	5	6
2	4	2	7
NULL	6	2	5

cardinality: {m to m+m tuples}

129

c) Full Outer Join (⟗)

$R \text{ ⟗ } S \equiv (R \text{ ⟕ } S) \cup (R \text{ ⟖ } S)$

R ⟗ S

A	B	C	D
2	4	5	6
2	4	2	7
3	5	NULL	NULL
NULL	6	2	5

130

DIVISION (/)

Enroll (Sid Cid ...)

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_2
S_3	C_1

↑ Students enrolled some course

Course (Cid ...)

Cid
C_1
C_2
C_3

↑ All Courses

$\pi_{Sid\ Cid}(Enroll) \;/\; \pi_{Cid}(course) \Rightarrow$

Sid
S_1

Eg: $S_1\ C_1$, $S_1\ C_2$, $S_1\ C_3$; C_1, C_2, C_3

→ if S_1 pairs with every course (C_1, C_2, C_3) in denominator then S_1 will come in the result.

"A" value must be paired with every "B" in denominator.

$\pi_{AB}(R) \;/\; \pi_{B}(S) \equiv$ Retrieves A values which are pairs with every B value of S.

Relation b/w SQL and Relational Algebra

SQL Query:

```
Select  Distinct  A1, A2, ... An
FROM    R1, R2, ... Rm
WHERE   P
```

SELECT ⇒ Projection (π) operator
FROM ⇒ CrossProduct (X)
WHERE ⇒ selection (σ_P) operator

Equivalent RA Query:

$$\pi_{A_1, A_2 .. A_n}\left(\sigma_P\left(R_1 \times R_2 \times .. \times R_m\right)\right)$$

Q. Sid's enrolled some course taught by Korth.

RA Query:

$$\pi_{Sid}\left(\sigma_{\substack{Enroll.Cid = Course.Cid \wedge \\ Inst = Korth}}\left(Enroll \times Course\right)\right)$$

Aggregate Functions

count()

sum()

Avg()

Min()

Max()

133

NESTED QUERIES

Query Inside Query

SELECT Attributes X

FROM tables ← Subquery

WHERE condition ←

GROUP BY (Attributes) X

HAVING condition ←

Order by (Attributes) X

can be used only in these 2 clauses.

subquery is not allowed with select, group by, and order by clause because these three clauses works on attributes.

134

Special functions used in Nested Queries

1. IN / NOT IN
2. ANY
3. ALL

} Best for Nested Queries with No Correlation

4. EXISTS / NOT EXISTS
5. UNIQUE / NOT UNIQUE

} Best for Nested Queries with Correlation.

(used to compare record value of outer Query with set of values resulted by Inner Query.)


```
Select branch
FROM student
 Group By Branch
 having Avg(marks) > (select avg(marks)
                      From student
                      where gender = female)
```

to retrieve avg marks of all females.