MODULE 2

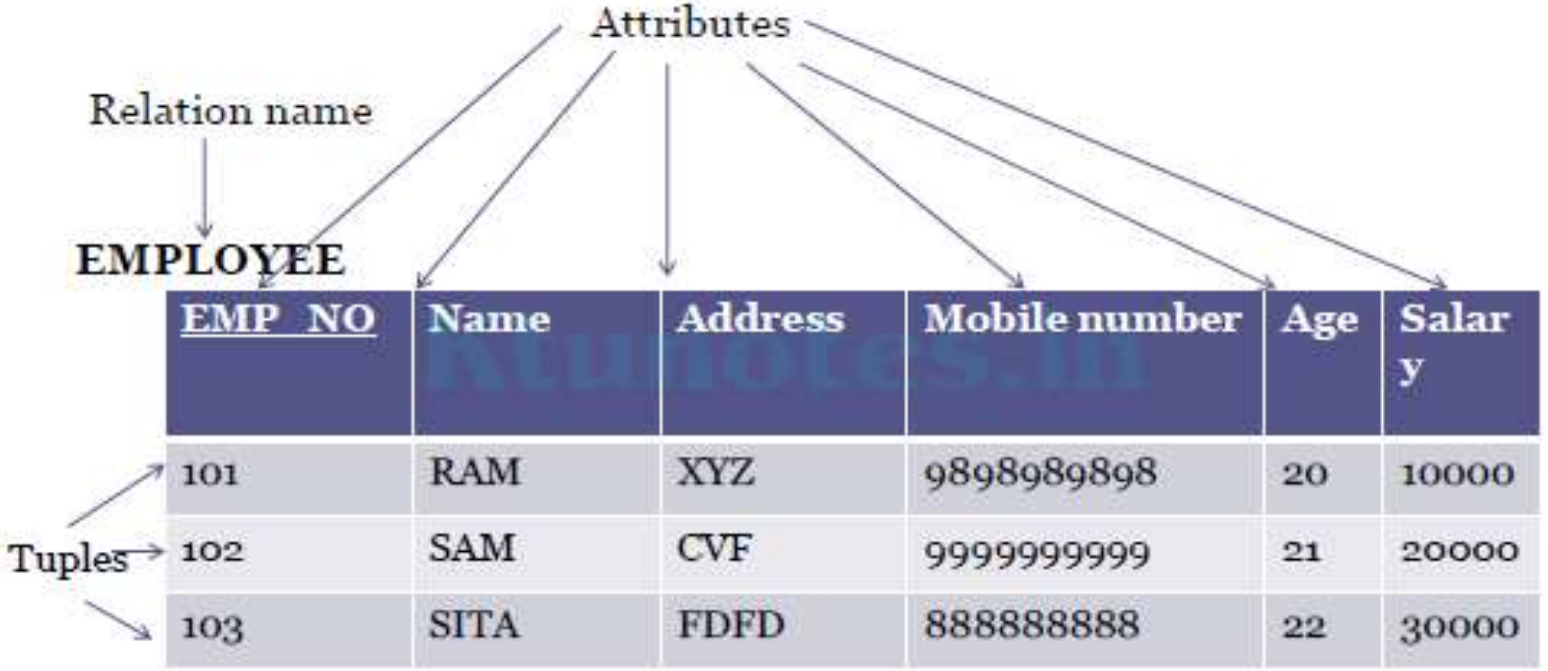
RELATIONAL MODEL

RELATIONAL DATABASE

• The Relational Model represents data and their relationships through a collection of tables, and can establish relationships between data points

- Each table also known as a **relation** consists of rows and columns**.**
- In other words, data and their relations are organized in 2 dimensional tables which consist of a set of rows and set of columns.
- Relation is a table which has values and rows in table is a collection of related data values
- Each row in table is a fact
- Row in relational table is called a **tuple**
- Every column has a unique name or column header and it corresponds to a specific attribute
- Column header is **attribute**
- The relational model has provided the basis for:
    - Research on the theory of data/relationship/constraint
    - Numerous database design methodologies
    - The standard database access language called s*tructured query language (SQL)*
    - Almost all modern commercial database management systems
- Some popular relational databases are used nowadays like Oracle, Sybase, DB2, MySQL Server etc.

Structure of relational databases



EMP NO	Name	Address	Mobile number	Age	Salary
101	RAM	XYZ	9898989898	20	10000
102	SAM	CVF	9999999999	21	20000
103	SITA	FDFD	888888888	22	30000

- In the relational model, all data is logically structured within **relations (also called table)**
- Informally a relation may be viewed as a named two-dimensional table representing an entity set.
- A relation has a fixed number of named columns and variable number of rows.

Components of relational database

- The main components of relational database structure are as follows:

1. Domains

2. Tuples (rows)

3. Columns

4. Keys

5. Relations (Tables)

1. Relation

- A table of values
- A relation may be thought of as a **set of rows.**
- A relation may alternately be thought of as a **set of columns.**
- That is a table is perceived as a two-dimensional structure composed of rows and columns.
- Each row represents a fact that corresponds to a real-world **entity or relationship.**
- Each row has a value of an item or set of items that uniquely identifies that row in the table
- Sometimes row – ids or sequential numbers are assigned to identify the rows in the table
- Each column typically is called by its column name or column header or attribute name

2. Domain

- A Domain is a set of atomic values.
- Atomic means each value in the domain is indivisible to the relational model.
- For each attribute there is a set of permitted values called domain of that attribute.
- It has three parts

  Name , Data type , Format
- A domain has a logical definition :

  eg : “ Phone numbers “ are the set of 10 digit phone numbers valid

3. Tuples (Rows)

- A tuple is a set of values
- Each row in the relation is known as a tuple.
- Tuple is a portion of a table containing data that described only entity, relationship, or object
- Also known as **record**
- Each value is derived from an appropriate domain
- For eg : The above relation contains 3 tuples one of which is shown as

102	SAM	CVF	9999999999	21	20000

Ie, < 102, SAM, CVF, 9999999999, 21, 20000 > is a tuple belonging to the EMPLOYEE relation

4 . Attributes (Columns)

- Columns in a table are also called **attributes or fields** of the relation
- A single cell in a table called field value, attribute value or data element
- For eg : for the entity EMPLOYEE , attributes could include Emp_No, Name, Addres, MobileNo, Age, Salary

EMP_NO	Name	Address	Mobile number	Age	Salary
101	RAM	XYZ	9898989898	20	10000

5. Keys

- These are basically the keys that are used to identify the rows uniquely or also help in identifying tables.

Other Terminologies:

- **Relation Instance :** The set of tuples of a relation at a particular instance of time is called a relation instance.
  It can change whenever there is an insertion, deletion or update in the database.

- **Degree :** The number of attributes in the relation is known as the degree of the relation.
  Eg : The EMPLOYEE relation defined above has degree 6.

- **Cardinality :** The number of tuples in a relation is known as cardinality.
  Eg: The EMPLOYEE relation defined above has cardinality 3.

- **NULL Values :** The value which is not known or unavailable is called a NULL value.
  It is represented by NULL

- **Relation State :** It is a subset of the cartesian product of the domains of its attributes
  Each domain consist of the set of all possible values of the attributes can take

Schema of a relation :

- It is the overall description of database.
- Ie, Schema of a relation is basically an outline of how the data is organized
- It is denoted by R(A1,A2,…An)
    - Here, R is relation name and it has some attributes A1 to An
- Each attribute have some domain and it is represented by dom(Ai)
- Relation state r (R) is a finite set of mappings r = { t1, t2, … tn}, where **each tuple *t*i is a mapping from R to D**, and
- D is the union of the attribute domains

  Ie, D = dom(A1) U dom(A2) U …U dom(An)

Definition Summary

Informal Terms

- Table
- Column Header
- All possible column values
- Row
- Table Definition
- Populated Table

Formal Terms

- Relation
- Attribute
- Domain
- Tuple
- Schema of a relation
- State of the relation

Example :

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI	NULL	18	IT

Characteristics of Relation

- Data is organized in tables with rows representing the records and columns representing the data fields
- Atomic Values :
    - Each attribute in table contains atomic values (ie, no multivalued or nested data is allowed in a single cell)
- Unique Key :
    - Every table has a primary key to uniquely identify each record ensuring no duplicate rows
- Each attribute has a defined domain, specifying the valid data types and constraints for values it can hold
- Ordering of tuples in a relation r(R)
    - Tuples are not considered to be ordered, even though they appear to be in the tabular form
    - In simple, the tuples are order independent
- Ordering of attributes in a relation schema R (and of values within each tuple)
    - Consider the attributes in R(A1,A2,…An) and the values in t = < v1,v2,…vn > to be ordered
- A special NULL value is used to represent values that are unknown or inapplicable to certain tuples
- It ensures data consistency through constraints
- It supports operations like selection, join, union, and intersection (enable data retrieval and manipulation)

Advantages of the Relational Model

- **Simple model:** Relational Model is simple and easy to use in comparison to other languages.
- **Flexible:** Relational Model is more flexible than any other relational model present.
- **Secure:** Relational Model is more secure than any other relational model.
- **Data Accuracy:** Data is more accurate in the relational data model.
- **Data Integrity:** The integrity of the data is maintained in the relational model.
- **Operations can be Applied Easily:** It is better to perform operations in the relational model.

Disadvantages of the Relational Model

- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

KEYS

- Keys are one of the basic requirements of a relational database model.
- It is widely used to identify the tuples(rows) uniquely in the table.
- We require keys in a DBMS to ensure that data is organized, accurate, and easily accessible.
- **Keys help to uniquely identify records or tuples in a table**, which prevents duplication and ensures data integrity.
- Keys also establish relationships between different tables, allowing for efficient querying and management of data.
- Without keys, it would be difficult to retrieve or update specific records, and the database could become inconsistent or unreliable.

- **Different Types of Database Keys :**
    - Primary Key
    - Candidate Key
    - Foreign Key
    - Composite Key
    - Alternate Key
    - Super Key

Super Key

- Attribute or a set of attributes that can uniquely identify a tuple (record) is known as Super Key
- ie, it is a set of attributes such that no two tuples in a valid relation will have same value
- A super key is a group of single or multiple keys that uniquely identifies rows in a table.
- It supports NULL values in rows.
- A super key can contain extra attributes that aren't necessary for uniqueness.

Example:

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Consider the table shown above.

STUD_NO+PHONE is a super key.

Primary Key

- A primary key is a unique identifier assigned to each record within a database table.
- The **primary key is an attribute that help to uniquely identify the tuples (records or rows) in the relational table**.
- ie, **It provides the tuple identity**
- A primary key index is automatically created for each table with a primary key column.
- The primary key is used to identify the tuples of a relation **none of its attribute values can be null**.
- Primary Keys are **Non-Null** values (ie, There is a mandatory value in the primary key attribute)
- **No duplicate values** are allowed (ie, coloumn assigned as primary key should have unique values only)
- Primary Keys are **Immutable** (ie, the values of primary key should not change frequently)
- **The primary key attributes are underlined**.
- Since a relation may have more than one candidate key (such as EMP_ Id, PAN no in EMPLOYEE table). So one of these candidate keys should be chosen as the primary key.
- **Example:** Consider the CAR relation schema

CAR(StateReg#, SerialNo, Make, Model, Year)

We chose SerialNo as the primary key

Composite Key

- In some cases, combination of attributes are set as primary key in relational model.
- A composite key **is a primary key that consists of two or more columns in a database table**.
- It is **used when a single column cannot uniquely identify a record**, but the combination of multiple columns can.
- **For example** consider the Student_Enroll relation schema

STUDENT_ENROLL (Stdid , Courseid ,Enroll_date)

Here , Stdid or Courseid cannot uniquely identify a row. Because a student can enroll in multiple courses . And also a course can have multiple students . So in this relation we use combination of multiple columns as Primary key

Candidate Key

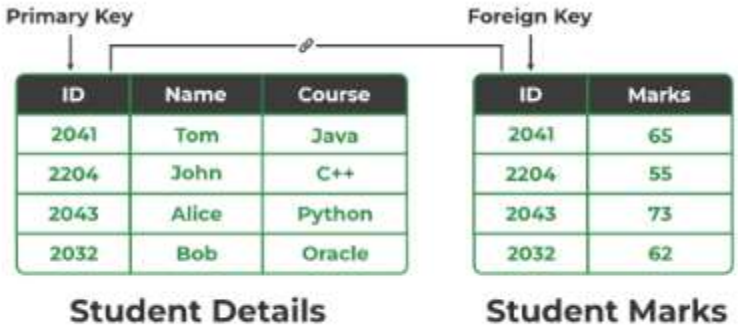
- A candidate key is **any attribute or combination of attributes that uniquely identifies rows in the table** and the attribute that forms the key can not be further reduced.
- Candidate key can be defined as a **minimal superkey**
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- A candidate key must contain unique values, ensuring that no two rows have the same value in the candidate key's columns.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key.
- **For Example** : Imagine a Student table as

STUDENT (StudentID, Name, Email)

Here, Both StudentID and Email can uniquely identify a student record, so they are both considered candidate keys. The database administrator would choose one of these as the primary key depending on the application needs.

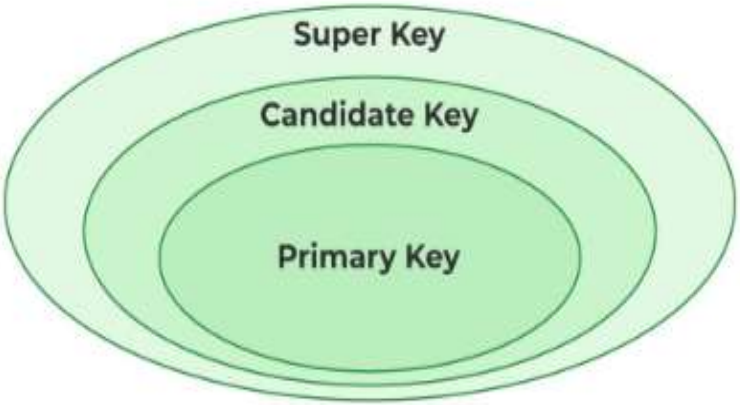
Foreign Key

- Also known as **Referential Key**
- A foreign key in a relational schema is **a column or columns in a table that reference a primary key in another table**.
- Foreign keys are **used to link tables together** and define relationships between data.
- A foreign key column in a table is linked to a column in another table.This is essential for maintaining data integrity and preventing data redundancy.
- They act as a cross-reference between the tables.
- The foreign key column's value must match a value in the referenced column of the other table.
- Table containing FK is called as **Child Table** and table which it refers are called as **Parent Table**
- Foreign keys in a relational schema is **represented by a directed arrow that starts at the foreign key field name and points to the column name of the related table**



Relation between Primary Key and Foreign Key

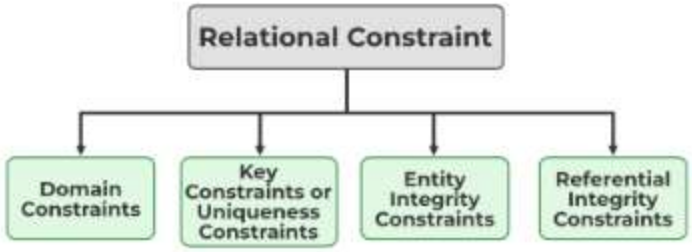
Relation between Primary Key, Candidate Key ,Super Key



- Primary Key (PK) is a candidate key, but candidate key is not a primary key
- Candidate key is a subset of Super Key (SK)
- ie, Super Key is the superset

RELATIONAL MODEL CONSTRAINTS

- In modeling the design of the relational database we can put some restrictions like what values are allowed to be inserted in the relation, and what kind of modifications and deletions are allowed in the relation.
- **Relational Constraints** are the restrictions or sets of rules imposed on the database contents.
- It validates the quality of the database.
- It validates the various operations like data insertion, updation, and other processes that have to be performed without affecting the integrity of the data.
- It protects us against threats/damages to the database.
- Mainly Constraints on the relational database are of 4 types
    - Domain constraints
    - Key constraints or Uniqueness Constraints
    - Entity Integrity constraints
    - Referential integrity constraints



DOMAIN CONSTRAINTS

- Domain constraints specify that within each tuple, the **value of each attribute 'A' must be an atomic value from the domain dom(A)**
- Every domain must contain atomic values (smallest indivisible units) which means composite and multi-valued attributes are not allowed.
- ie, Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

- Eg 1 . If we assign the data type of attribute age as int, we can't give it values other than int datatype.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

- Eg 2.

In the above relation, Name is a composite attribute and Phone is a multi-values attribute, so it is violating domain constraint

EID	Name	Phone
01	Bikash Dutta	123456789 234456678

KEY CONSTRAINTS (and Constraints on NULL values)

- These are **called uniqueness constraints**
- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- Also it **ensures that every tuple in the relation should be unique.**
- A relation can have multiple keys or candidate keys out of which we choose one of the keys as the primary key, we don't have any restriction on choosing the primary key out of candidate keys.
- Null values are not allowed in the primary key, hence **Not Null constraint is also part of the key constraint.**
- **Eg :**

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

- In the above table, ID is the primary key, and the third and the last tuple have the same value in ID ie, 1002. So it is violating the key constraint.

ENTITY INTEGRITY CONSTRAINTS

- Entity integrity constraints say that no primary key can take a NULL value
- This is because the primary key value is used to identify individual tuples (rows) in a relation
- Having NULL values for the primary key implies that we cannot identify some tuples
- A table can contain a null value other than the primary key field.
- For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

- In the above relation, EMP_ID is made the primary key, and the primary key can't take NULL values but in the fourth tuple, the primary key is null, so it is violating Entity Integrity constraints.

REFERENTIAL INTEGRITY CONSTRAINTS

- Key constraints and entity integrity constraints are specified on individual relations.
- The **referential integrity constraint is specified between two relations** and is used to maintain the consistency among tuples in the two relations.
- Informally, the referential integrity constraint **states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.**
- To define referential integrity more formally, first we define the concept of a foreign key.
- The conditions for a foreign key ,given below ,specify a referential integrity constraint between the two relation schemas R1 and R2.
- A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following rules :

    - When an **attribute in the foreign key of relation R1 has the same domain as the primary key of relation R2**, then the foreign key of R1 is said to reference or refer to the primary key of relation R2.

    - The **values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values**, but can't be empty.

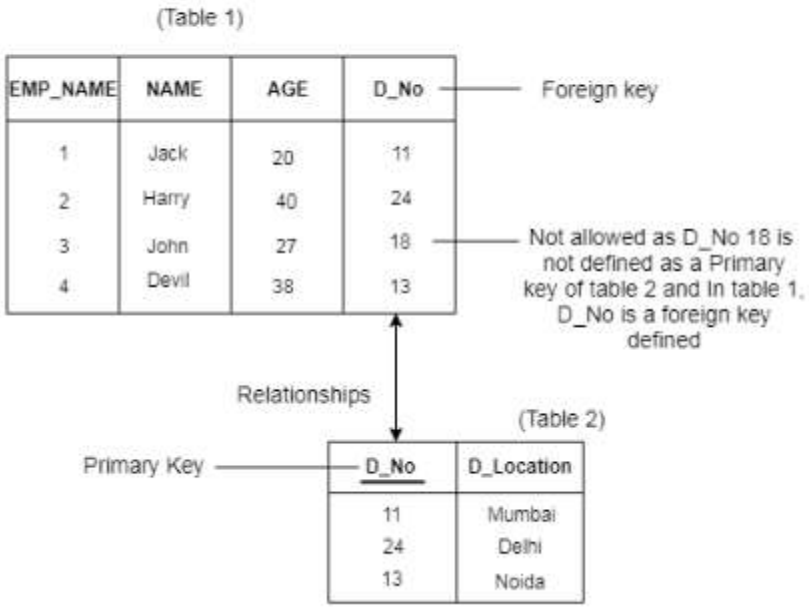
- **t1[FK] = t2[PK]**

We say that the tuple t1 references or refers to the tuple t2.

In this definition, **R1 is called the referencing relation**

R2 is the referenced relation

Example:



UPDATE OPERATIONS & VIOLATION OF INTEGRITY CONSTRAINTS

- The operations of the relational model can be categorized into **retrievals and updates**.
- There are three basic operations that can change the states of relations in the database: **Insert, Delete, and Update**
- ie, They insert new data, delete old data, or modify existing data records.
- Insert is used to insert one or more new tuples in a relation
- Delete is used to delete tuples
- Update (or Modify) is used to change the values of some attributes in existing tuples.
- Whenever **these operations are applied**, ,the **integrity constraints** specified on the relational database schema that **may be violated** by each of these operations

1. INSERT OPERATION

- The Insert operation provides a list of attribute values for a new tuple ***t*** that is to be inserted into a relation R.
- **Insert can violate any of the four types of constraints**
- **Domain constraints** can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type.
- **Key constraints** can be violated if a key value in the new tuple ***t*** already exists in another tuple in the relation r(R).
- **Entity integrity** can be violated if any part of the primary key of the new tuple ***t*** is NULL.
- **Referential integrity** can be violated if the value of any foreign key in *t* refers to a tuple that does not exist in the referenced relation.

- **Example** to illustrate these constraints :

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Fig : 2.1 COMPANY relational database

1. Insert <‘Cecilia’,‘F’,‘Kolonsky’, NULL,‘1960-04-05’,‘6357 Windy Lane, Katy, TX’,F,28000,NULL,4> into EMPLOYEE.

   Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn),so it is rejected

2. Insert <‘Alice’, ‘101’,Zelaya’,‘999887777’,‘1960-04-05’,‘6357 Windy Lane, Katy, TX’,F,28000,‘987654321’,4> into EMPLOYEE.

   Result : This insertion **violates the key constraint** because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

3. Insert <‘Cecilia’,‘F’, ‘Kolonsky’, ‘677678989’,‘1960-04-05’, ‘6357 Windswept, Katy,TX’,F,28000,‘987654321’,7 > into EMPLOYEE.

   Result : This insertion **violates the referential integrity constraint** specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

4. Insert <‘Cecilia’,‘F’,‘Kolonsky’,‘677678989’,‘1960-04-05’,‘6357 Windy Lane, Katy,TX’,F,28000,NULL,4> into EMPLOYEE.

   Result : This insertion satisfies all constraints, so it is acceptable.

- If an insertion violates one or more constraints, the default option is to reject the insertion.

2. DELETE OPERATION

- **The Delete operation can violate only referential integrity**.
- This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database.
- To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted.
- Here are some **example**.

1. Delete the WORKS_ON tuple with Essn = ‘999887777’and Pno = 10.

   Result: This deletion is acceptable and deletes exactly one tuple.

2. Delete the EMPLOYEE tuple with Ssn = ‘999887777’.

   Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

3. Delete the EMPLOYEE tuple with Ssn = ‘333445555’.

   Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

- Three options are available if a deletion operation causes a violation.
- The **first option, called restrict**, is to reject the deletion.

- The **second option, called cascade**, is propagate the new primary key value into the foreign keys of the referencing tuples
- **Cascade** option in a deletion operation ensures that when a record in a parent table (referenced record) is deleted, all related records in the child table are automatically deleted. This is a referential action defined as part of the **foreign key constraint** in the child table.

- A **third option, called set null or set default**, is to modify the referencing attribute values that cause the violation; each such value is either set to NULL or changed to reference another default valid tuple.
- If a referencing attribute that causes a violation is part of the primary key, it cannot be set to NULL; otherwise, it would violate entity integrity.
- **Set Null option** in deletion operation ensures that when a record in the parent table is deleted, **all corresponding foreign key values in the child table are set to NULL** instead of deleting the child records.
- **Set default option** in a deletion operation ensures that when record in the parent table is deleted, **all corresponding foreign key values in the child table are set to a predefined default value** instead of being deleted or set to NULL. This helps to maintain referential integrity while ensuring that child records still have valid references

3. UPDATE OPERATION

- The Update (or Modify) operation is used to change the values of one or more attributes in a tuple (or tuples) of some relation R.
- It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified.
- **UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified**

- **Updating an attribute that is neither part of a primary key nor of a foreign key** usually causes no problems, the DBMS need only check to confirm that the new value is of the correct data type and domain. (ie, **only violate domain constraint**)

- **Modifying a primary key value is similar to deleting one tuple and inserting another in its place** because we use the primary key to identify tuples**.**

- **If a foreign key attribute is modified**, the DBMS must make sure that the new value refers to an existing tuple in the referenced relation (or is set to NULL). Ie, **it may violate referential integrity**

- Here are some examples :

1. Update the salary of the EMPLOYEEtuple with Ssn = ‘999887777’to 28000.

   Result: Acceptable.

2. Update the Dno of the EMPLOYEEtuple with Ssn = ‘999887777’to 1.

   Result: Acceptable

3. Update the Dno of the EMPLOYEEtuple with Ssn = ‘999887777’to 7.

   Result: Unacceptable , because it violates referential integrity.

4. Update the Ssn of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘987654321’.

   Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.

SYNTHESIZING ER DIAGRAM TO RELATIONAL SCHEMA

Mapping of ER Diagram to Relational Schema

- Relation schema defines the design and structure of the relation or table in the database.
- It is the way of representation of relation states in such a way that every relation database state fulfills the integrity constraints set (Like Primary key, Foreign Key, Not null, Unique constraints) on a relational schema.
- Conceptual ER-models allow to more accurately represent the subject
- But, there are no DBMSs that support ER models.
- So, ER diagram is converted into the tables in Relational Data Model (RDM or RM).
- Relational models can be easily implemented by RDBMS like mySQL, MS SQL, PostgreSQL, Oracle etc.

- **How to display relational database schema ?**
    - An entity type within ER diagram is turned into a table
    - Name of the relation is written above the row table
    - Each attribute turns into a column in the table.
    - ie, Each relation schema can be displayed as a row of attribute names
    - The key attribute of the entity is the primary key of the table which is usually underlined.
    - It is highly recommended that every table should start with its primary key attribute.
    - Foreign Key is displayed as a directed arrow from Foreign Key attributes to referenced table.

- To convert ER model to Relational model, a **7 step algorithm** is used

- **Algorithm**

  Step 1 : Mapping of Regular (Strong) Entity Types

  Step 2 : Mapping of Weak Entity Types

  Step 3 : Mapping of Binary 1:1 Relation Types

  Step 4 : Mapping of Binary 1:N Relationship Types.

  Step 5 : Mapping of Binary M:N Relationship Types.

  Step 6 : Mapping of Multi-valued attributes.

  Step 7 : Mapping of N-ary Relationship Types.

Step 1 : Mapping of Regular (Strong) Entity Types

(a) <u>Strong Entity Set With Only Simple Attributes</u>

- A strong entity set with only simple attributes will require only 1 table in relational model.
- Attributes of the table will be the attributes of the entity set.
- Derived attributes removed from the table.
- The primary key of the table will be the key attribute of the entity set.

(b) <u>Strong Entity Set With Composite Attributes</u>

- A strong entity set with any number of composite attributes will require only 1 table in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

Step 2 : Mapping of Weak Entity types

• Weak entity set always appears in association with identifying relationship with total participation constraint.

• For each weak entity type W in the ER with owner entity type E, create a table R and include all simple attributes of W as attributes of R

• Include as foreign key attributes of R, the primary key attributes of the table that correspond to the owner entity type.

• The primary key of R is the combination of the primary key of the owner and the partial key of the weak entity type W

Step 3 : Mapping of Binary 1 : 1 Relationship Types

• For each binary 1:1 relationship type R in the ER, identify the tables S and T that correspond to the entity types participating in R. There are three possible approaches :

a) The foreign key approach (more used)

b) The merged relationship approach

c) The cross reference or relationship relation approach

a) Foreign Key Approach

- Choose one of the tables between S and T and include in S as a foreign key, the primary key of T
- It is better to choose an entity type with total participation in R in the role of S
- Include all the simple attributes (or simple components of composite attribute) of the 1 : 1 relationship type R as attribute of S

b) Merged Relation Approach

- We can merge the two entity and the relationship into a single relation
- This is **possible when both participations are total ,** because the two tables will have the exact same number of tuples

c) Cross- reference or relationship relation approach

- Setting up a third table R for the purpose of cross- referencing the primary keys of the two tables S and T representing the entity
- The table R will include the primary key attributes of S and T as foreign key to S and T. The primary key of R will be one of the two foreign keys
- **The drawback is having an extra table**, and requiring an extra join operation when combining related tuples from the tables

Step 4 : Mapping of Binary 1 : N Relationship Types

• For each binary 1: N relationship R, **identify the table S at the N side of the relationship**

• Include as foreign key in S , the primary key of the table T that represents the other entity participating in R

(We do this because each instance on the N -side can be related to more than one instance on the 1- side of the relationship)

• Include any simple attributes of the relationship as attribute of S, if any.

Step 5 : Mapping of Binary M : N Relationship Types

• For each binary M : N relationship R, create a new table S to represent R

• Include as foreign key attributes in S, the primary keys of the tables that represent the participating entity; their combination will form the primary key of S

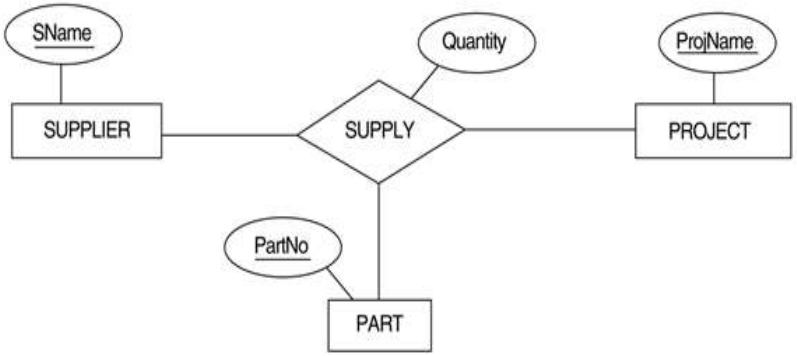
• Include any simple attributes of the M : N relationship as attributes of S , if any

Step 6 : Mapping of Multi-valued Attributes

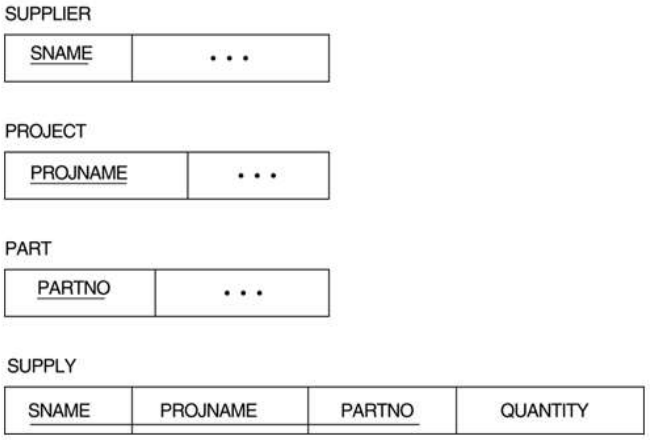
- For each multi -valued attribute A create a new table R
- Include in R an attribute corresponding to A
- Include in R the primary key attribute as a foreign key in R
- Primary Key of the table that represent the entity that has A as a multi-valued attribute
- The primary key of R is the combination of A and K
- If the multi-valued attribute is a composite attribute, we include in R its simple attributes

Step 7 : Mapping N-ary Relationships

- For each n-ary relationship type R, where n>2, create a new relation S to represent R.
- Primary keys of participating relations in R become foreign keys in S.

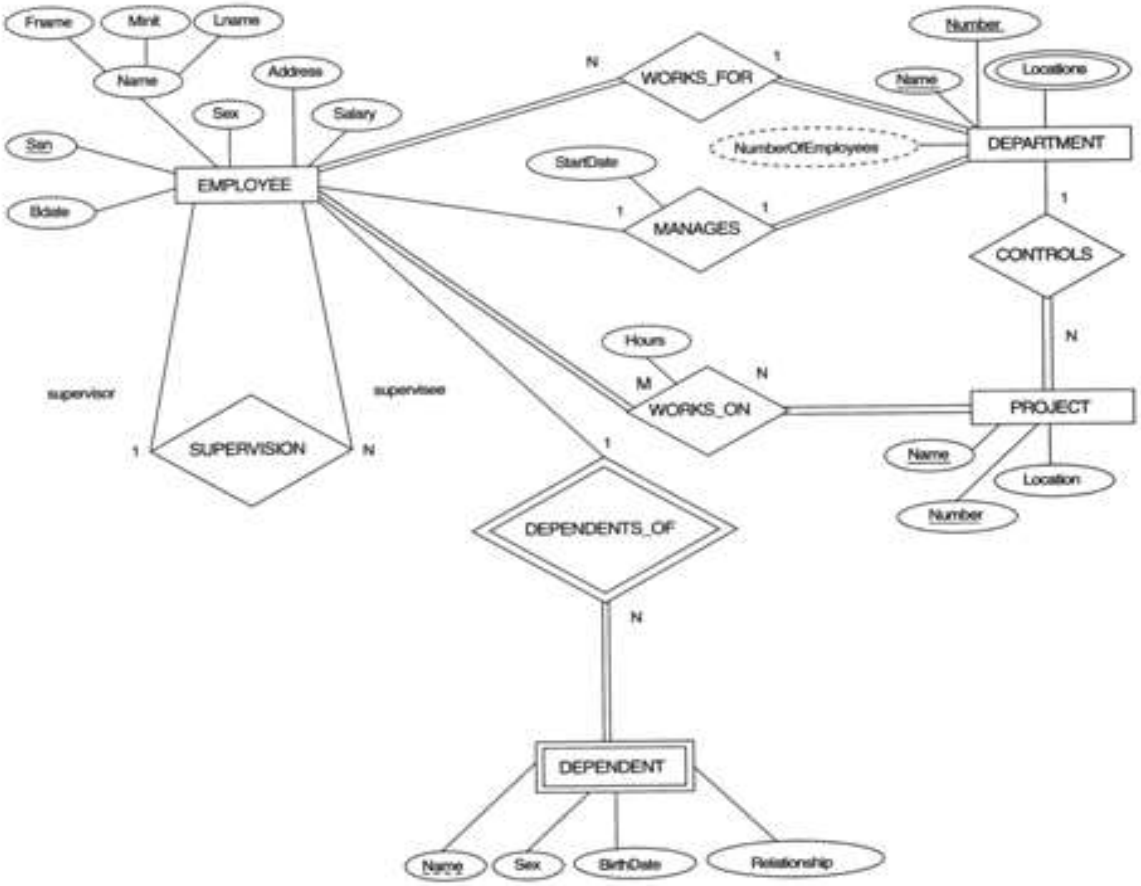


ER Diagram with Ternary relation



Mapping the *n*-ary relationship type SUPPLY

Eg : Convert the ER Conceptual Schema for Company database into relational schema



Detailed Explanation of how to convert ER Schema for a COMPANY database to Relational Schema :

Step 1: **Mapping of Regular Entity Types**.

• We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Step 2 : **Mapping of Weak Entity Types**

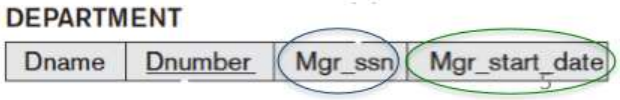
- Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  - Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
  - The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

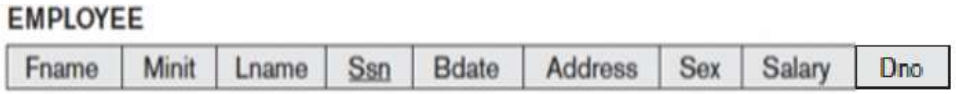
Step 3: **Mapping of Binary 1:1 Relation Types (Foreign Key Approach)**

- 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.



Step 4 : **Mapping of Binary 1:N Relationship Types**

- 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
- For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO. (DEPENDENT relation is already satisfied in Step 2)



(WORKS _FOR)

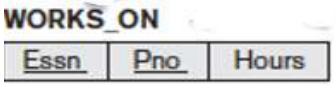
(CONTROLS)

(SUPERVISION)

Step 5 **: Mapping of Binary M:N Relationship Types.**

• The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.

- The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
- Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

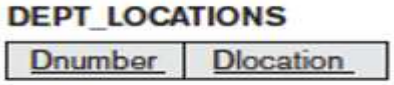


Step 6 **: Mapping of Multi-valued attributes**

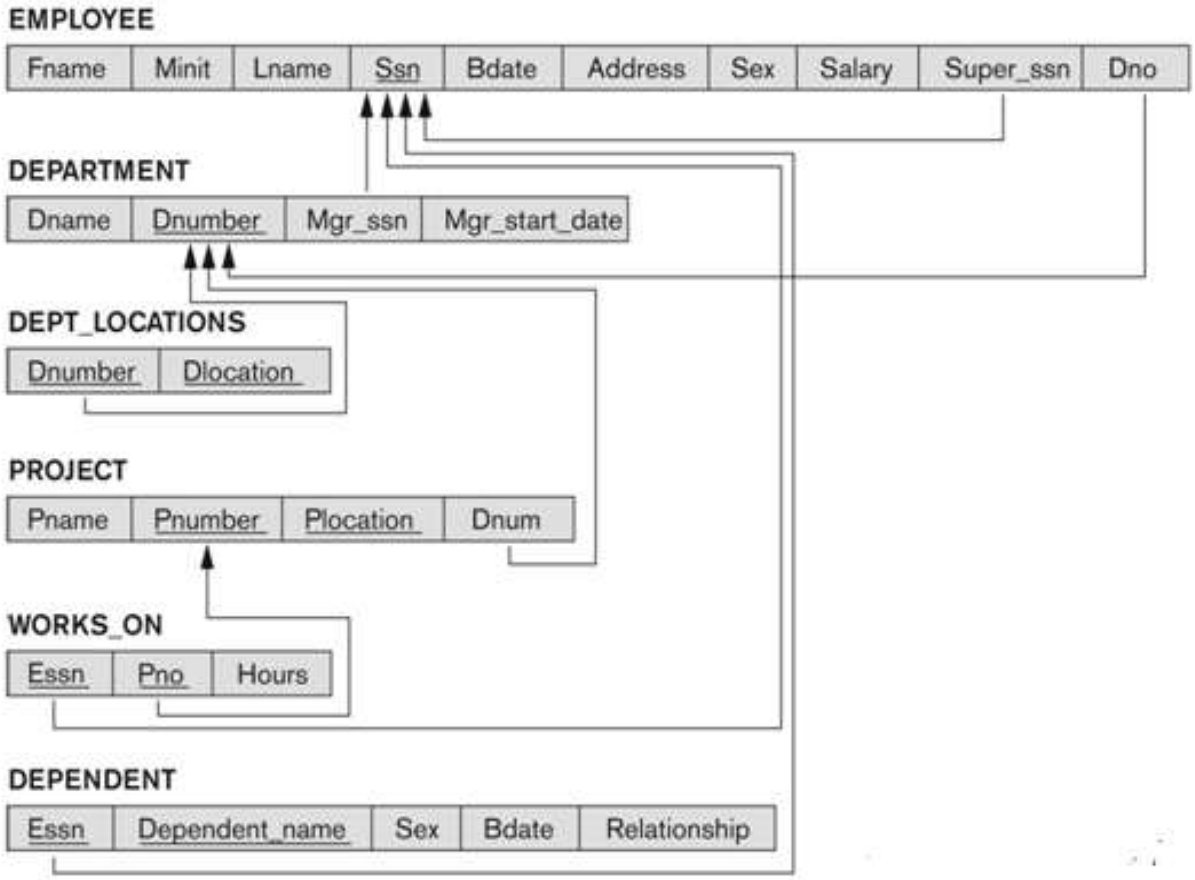
- The relation DEPT_LOCATIONS is created.

• The attribute DLOCATION represents the multi-valued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.

- The primary key of R is the combination of {DNUMBER, DLOCATION}.

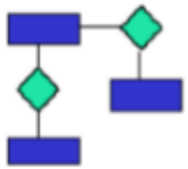


Final Result of mapping the COMPANY ER schema into a relational schema.



Note : Foreign keys have been linked with a directional line (directed arrow) with their own primary key

Correspondence between ER and Relational Schema



ER MODEL	RELATIONAL MODEL
Entity	Relation or Table
1:1 relationship	Joining relation or Foreign Key (or relationship relation)
1:N relationship	Foreign Key (or relationship relation)
M:N relationship	Relationship relation and two foreign keys
Simple attribute	Attribute
Composited Attribute	Set of the simple component attributes
Multivalued attribute	Table and foreign key
Value set	Domain
Key attribute and/or Candidate key	Primary key

MODULE 2 (Part II)

INTRODUCTION TO RELATIONAL ALGEBRA

&

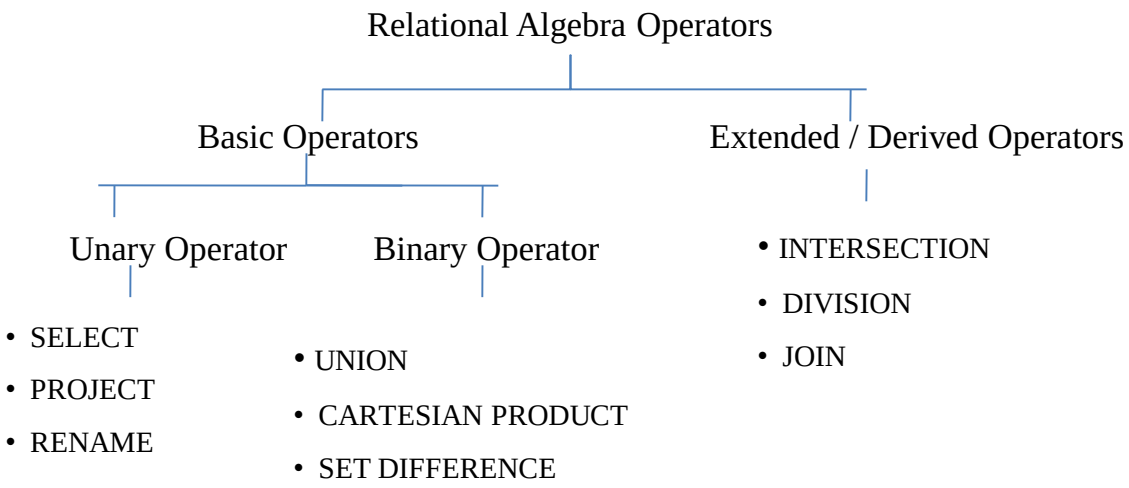
SQL QUERIES (DDL,DML)

Prepared By,
Josmi Jose
Assistant Professor
Department of CSE
College of Engineering,Chengannur

Relational Algebra

- Relational Algebra is **a procedural query language**.
- It mainly provides a theoretical foundation for relational databases and SQL.
- It consists of a certain **set of operations that are widely used to manipulate and query data from a relational database**.
- ie, It gives a step by step process to obtain the result of the query & uses operators to perform queries.
- The main purpose of using Relational Algebra is to **define operators that transform one or more input relations into an output relation.**
- The basic set of operations for the relational model is the relational algebra.

TYPES OF RELATIONAL OPERATORS



Unary Relational Operations:

1. SELECT operation

- It is used **to choose a subset of the tuples from a relation that satisfies a selection condition**
- ie, the SELECT operation to be a filter that keeps only those tuples that satisfy a qualifying condition
- It can also be **visualized as a horizontal partition** of the relation into two sets of tuples
- Those tuples that satisfy the condition and are selected, and those tuples that do not satisfy the condition and are discarded
- **It is denoted by sigma (σ)**
- $\sigma_{<\text{selection condition}>}(R)$
    - R is generally a relational algebra expression whose result is a relation ,the simplest such expression is just the name of a database relation.
    - The relation resulting from the SELECT operation has the same attributes as R.

Eg :

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

• For example ,to select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000, we can individually specify each of these two conditions with a SELECT operation as follows:

σ Dno = 4 (EMPLOYEE)

σ Salary > 30000 (EMPLOYEE)

The Boolean expression specified in <selection condition> is made up of a **number of clauses** of the form

<attribute name> <comparison op> <constant value>

or

<attribute name> <comparison op> <attribute name>

where <attribute name> is the name of an attribute of R,<comparison op> is normally one of the operators {=,<,≤,>,≥,≠},and <constant value> is a constant value from the attribute domain.

- Clauses can be connected by the standard Boolean operators AND, OR, NOT to form a general selection condition.

- For example, to select the tuples for all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000, we can specify the following SELECT operation:

σ(Dno=4 AND Salary>25000) OR (Dno=5 AND Salary>30000)(EMPLOYEE)

The result is shown below

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

- The SELECT operator is unary ; that is, it is applied to a single relation.
- Moreover, the selection operation is applied to each tuple individually; hence, selection conditions cannot involve more than one tuple.
- **The degree of the relation resulting from a SELECT operation** (its number of attributes**) is the same as the degree of R**.
- The number of tuples in the resulting relation is always less than or equal to the number of tuples in R.
- The **SELECT operation is commutative**

  - Ie, σ <cond1> (σ <cond2> (R)) = σ <cond2> (σ <cond1> (R))

  Hence, a sequence of SELECTs can be applied in any order

- We can always combine a cascade (or sequence) of SELECT operations into a single SELECT operation with a conjunctive (AND) condition

  - ie, $\sigma_{<cond1>}(\sigma_{<cond2>}(...(\sigma_{<condn>}(R))\ ...)) = \sigma_{<cond1>\ \mathbf{AND}\ <cond2>\ \mathbf{AND}...\mathbf{AND}\ <condn>}(R)$

2. PROJECT Operation

- The PROJECT operation, **selects certain columns from the table and discards the other columns**.
- If we are interested in only certain attributes of a relation, we use the PROJECT operation to project the relation over these attributes only.
- Therefore, the result of the PROJECT operation can be **visualized as a vertical partition** of the relation into two relations
- **It is denoted by ∏.**
- ∏ A1, A2, An (R)

  where, A1, A2, A3 is used as an attribute name of relation R.

- For example, to list each employee's first and last name and salary,we can use the PROJECToperation as follows:

  ∏ Lname, Fname, Salary (EMPLOYEE)

- The resulting relation is shown below

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

• The result of the PROJECT operation has only the attributes specified in <attribute list> in the same order as they appear in the list.

• Hence, **its degree is equal to the number of attributes in <attribute list>.**

• The **PROJECT operation removes any duplicate tuples,** so the result of the PROJECT operation is a set of distinct tuples, and hence a valid relation. This is known as duplicate elimination.

• For example, consider the following PROJECT operation

$$\prod_{Sex, Salary}(EMPLOYEE)$$

The result is shown below

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

• The number of tuples in a relation resulting from a PROJECT operation is always less than or equal to the number of tuples in R.

• **Commutativity does not hold on PROJECT**.

3. RENAME operation

- Rename operation is a unary operator and it is used **to rename the output relation name**
- For most queries, we need to apply several relational algebra operations one after the other
- Either we can write the operations as a single relational algebra expression by nesting the operations, or we can apply one operation at a time and create intermediate result relations we must give names to the relations that hold the intermediate results.
- The results of the operations that we are performing are saved without any name.
- If we want to give some name to the result of any operation then we can rename the result of the operations using the rename operation.
- It can also be used to rename an old relation.
- **It is denoted by symbol rho (ρ).**
- If no renaming is applied, the names of the attributes in the resulting relation of a SELECT operation are the same as those in the original relation and in the same order.
- For a PROJECT operation with no renaming, the resulting relation has the same attribute names as those in the projection list and in the same order in which they appear in the list.

RENAME Operation General form

• RENAME operation when applied to a relation R of degree n is denoted by any of the following three forms

(i) $\rho_{S(B1, B2, ..., Bn)}(R)$

where , S is the new relation name

B1, B2, ..., Bn are the new attribute names.

- This expression renames both the relation and its attributes

(ii) $\rho_S(R)$

- This will renames the relation only

(iii) $\rho_{(B1, B2, ..., Bn)}(R)$

- This will renames the attributes only.
- If the attributes of R are (A1, A2, ..., An) in that order, then each Ai is renamed as Bi.

Binary Relational Operations:

1. Union Operation

- This is a binary operation
- The Union operation is applied to two relations
- It is denoted by U
- The union operation results in a relation that **includes all the tuples that are in either (or both) of the two relations**
- The relations on which union operation is applied should be **union compatible**
- Two relations are union compatible ,
  - if they have the same degree or same number of attributes (ie, type compatible)
  - And the domains (or datatypes)of the corresponding attributes are identical (ie, domain compatible)

- The UNION operation also **removes the duplicate tuples** from the resulting relation
- The general synatx of the UNION operation is :

  R = P U Q has tuples drawn from P & Q

- The **UNION operation is commutative** . Ie, P U Q = Q U P

Example: FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

Consider the following table of Students having different optional subjects in their course.

π (Student_Name) FRENCH U π (Student_Name) GERMAN

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

Eg : 2

To retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5,we can use the UNION operation as follows

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$$

$$RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$$

$$RESULT \leftarrow RESULT1 \cup RESULT2$$

- The relation RESULT1 has the Ssn of all employees who work in department 5, whereas RESULT2 has the Ssn of all employees who directly supervise an employee who works in department 5.
- The UNION operation produces the tuples that are in either RESULT1 or RESULT2 or both while eliminating any duplicates. Thus,the Ssn value '333445555'appears only once in the result.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

2. Set-Difference Operation (Minus Operation)

- This is a binary operation
- **It is denoted by** –
- The general syntax of the SET DIFFERENCE operation is: R=P-Q
- The SET DIFFERENCE operation results in a relation that includes all tuples that are in the first relation, but not in the second relation.
- **The relations on which set difference operation is applied should be union compatible**.
- The SET DIFFERENCE operation is **not commutative**.

  ie, $P - Q \neq Q - P$

Eg :

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

INSTRUCTOR − STUDENT.

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

3. CARTESIAN PRODUCT (CROSS PRODUCT)

- It is a binary operation
- This operation is used to **obtain all possible combinations of tuples from two relations**
- **It is denoted by X**
- In general, the result of P (A1,A2,…An) X Q (B1,B2,…Bm) is a relation R with n + m attributes

  R (A1,A2,…An,B1,B2,…Bm) in that order
- **The degree of the resulting relation is n + m**
- **No. of tuples in resulting relation will be the number of tuples in P multiplied by the number of tuples in Q**
- **ie, Cardinality of resulting relation is n * m**
- The general syntax of cartesian product is R = P X Q
- Same attribute name can be appear in both relations
- Example :

Student

Name	Age	Sex
Ram	14	M
Sona	15	F
Kim	20	M

Course

ID	Course
1	DS
2	DBMS

Student X Course

Name	Age	Sex	ID	Course
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

DERIVED OPERATIONS

1. INTERSECTION OPERATION

- This is a binary operation
- **The intersection operation results in a relation that includes all tuples that are in both of the two relations**
- Ie, it selects common tuples in two relations
- The relation on which intersection operation is applied should be **union compatible**
- **It is denoted by ∩ . Ie, P ∩ Q**
- Example :

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

π(Student_Name)FRENCH ∩ π(Student_Name)GERMAN

Ie, FRENCH ∩ GERMAN

Student_Name
Vivek
Geeta

2. DIVISION OPERATION

- It is a binary operation and **denoted by** ÷
- **Ie, written as P ÷ Q ,** where P and Q are relations such that **attributes of P are a superset of attributes of Q**
- **The result of division operation will be a relation with attributes unique to P**
  - **Ie, attributes in P which are not in Q**

The UNION, INTERSECTION, and MINUS Properties

- UNION and INTERSECTION are commutative operations

$$R \cup S = S \cup R \text{ and } R \cap S = S \cap R$$

- Both UNION and INTERSECTION can be treated as N-ary operations applicable to any number of relations because both are also associative operations

$$R \cup (S \cup T) = (R \cup S) \cup T \text{ and } (R \cap S) \cap T = R \cap (S \cap T)$$

- The MINUS operation is not commutative; that is, in general,

$$R - S \neq S - R$$

- INTERSECTION can be expressed in terms of union and set difference as follows

$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$

3. JOIN Operation

- The JOIN operation, denoted by ⋈
- It is **used to combine related tuples from two relations into single** "longer" tuples.
- This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations
- The general form of a JOIN operation on two relations R (A1,A2,…An) and S (B1,B2,…Bm) is

$$R \bowtie_{(join\ condition)} S$$

- In general, the result of JOIN operation of two relations P (A1, A2,…An) and Q (B1,B2,…Bm) in that order
- The join operation retreives all tuples from the cartesian product of the two relations with a matching condition
- Example :

EMP . Dept_Name = DEPT . Dept_Name

Employee

Name	ID	Dept_Name
A	120	IT
B	125	HR
C	110	Sales
D	111	IT

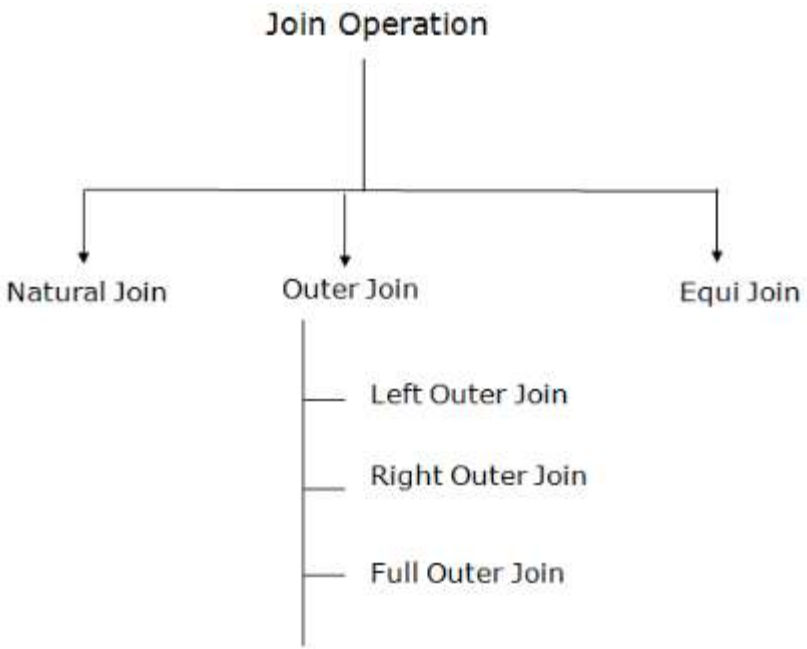
Department

Dept_Name	Manager
Sales	Y
Production	Z
IT	A

Ie, Employee ⋈ Department

Name	ID	Dept_Name	Manager
A	120	IT	A
C	110	Sales	Y
D	111	IT	A

Types of Join operations:



Example

Retrieve the name of the manager of each department.

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$
$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

• To get the manager's name, we need to combine each department tuple with the employee tuple whose Ssn value matches the Mgr_ssn value in the department tuple.

• We do this by using the JOIN operation and then projecting the result over the necessary attributes, as follows

The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation.

Difference between JOIN and CARTESIAN PRODUCT

- In JOIN, only combinations of tuples satisfying the join condition appear in the result,
- Whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.
- The join condition is specified on attributes from the two relations R and S and is evaluated for each combination of tuples.
- Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation Q as a single combined tuple.

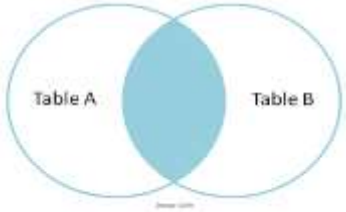
INNER JOIN

• It is a join operation in DBMS that combines two or more tables based on related columns and returns only rows that have matching values among tables.

• In simple, **Returns records that have matching values in both tables**

• It can be classified into 3

1. Theta join
2. EQUI join
3. Natural join



Theta join (or Conditional Join)

• It is a type of inner join in which tables are combined based on the specified condition, where each <condition> is of the form **Ai θ Bj**, Ai is an attribute of R, Bj is an attribute of S, Ai and Bj have the same domain, and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.

• ie, In conditional join, the join condition can include <, >, <=, >=, ≠ operators in addition to the = operator.

• Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result

• The JOIN operation does not necessarily preserve all of the information in the participating relations, because tuples that do not get combined with matching ones in the other relation do not appear in the result.

• Eg : $A \bowtie_{S<T} B$

Equi Join

- It is a type of Inner join in which we use equivalence ('=') condition in the join condition
- In the result of an EQUIJOIN we always have one or more pairs of attributes that have identical values in every tuple
- Eg : $A \bowtie_{A.Column\ B\ =\ B.Column\ B} (B)$

Natural Join

- It is a type of inner join in which we do not need any comparison operators.
- In natural join, columns should have the same name and domain.
- There should be at least one common attribute between the two tables.
- ie, Natural join requires that the two join attributes (or each pair of join attributes) have the same name in both relations.
- If this is not the case, a renaming operation is applied first.
- Eg : $A \bowtie B$
- The join condition for NATURAL JOIN is constructed by equating each pair of join attributes that have the same name in the two relations and combining these conditions with AND.
- There can be a list of join attributes from each relation, and each corresponding pair must have the same name.

More general, but nonstandard definition for NATURAL JOIN is

$$Q \leftarrow R *_{(<\text{list1}>),(<\text{list2}>)} S$$

- <list1> specifies a list of i attributes from R, and
- <list2> specifies a list of i attributes from S.
- The lists are used to form equality comparison conditions between pairs of corresponding attributes,

OUTER JOIN

- It is a type of join that retrieves matching as well as non-matching records from related tables.
- These three types of outer join
    1. Left outer join
    2. Right outer join
    3. Full outer join

Left Outer Join

- It is also called left join.
- In the left outer join, operation allows keeping all tuple in the left relation.
- ie, this type of outer join retrieves all records from the left table and retrieves matching records from the right table
- If there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values

Eg : Suppose there are two tables Table A and Table B

Table A

Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

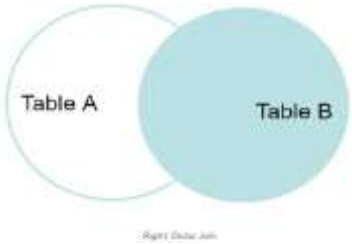
$A \bowtie B$ is

Output

Number	Square	Cube
2	4	8
3	9	27
4	16	NULL

Right Outer Join

- It is also called a right join.
- This type of outer join retrieves all records from the right table and retrieves matching records from the left table
- However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values



Eg : Suppose there are two tables Table A and Table

Table A

Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

$A \bowtie B$ is

Output:

Number	Square	Cube
2	4	8
3	9	27
5	NULL	125

Full Outer Join

- It creates the result set by combining the results of both LEFT JOIN and RIGHT JOIN.
- The result set will contain all the rows from both tables.
- For the rows for which there is no matching, the result set will contain *NULL* values.
- Eg : Suppose there are two tables Table A and Table B

Table A

Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

$A \mathbin{⟗} B$ *is*

Output:

Number	Square	Cube
2	4	8
3	9	27
4	16	NULL
5	NULL	125

Q) Consider the following relational database of a college

Student (Rollno, Sname, Address)

Teacher (Tid , Tname , Tsubject)

Colege (Rollno , Tid)

Write relational algebra for following statements :

a) Find the name of students who live in Delhi

$$\Pi_{Sname}(\sigma_{Address="Delhi"}(\text{Student}))$$

b) Find the name of teacher who teaches “DBMS”

$$\Pi_{Tname}(\sigma_{Tsubject="DBMS"}(\text{Teacher}))$$

c) Find the name of teacher who teaches “OS” to a student “ John”

$$A \leftarrow \sigma_{Sname=" John"} (\text{Student})$$

$$B \leftarrow A \bowtie \text{College}$$

$$C \leftarrow \sigma_{Tsubject=" OS"} (\text{Teacher})$$

$$D \leftarrow B \bowtie C$$

$$\text{Result} \leftarrow \prod_{Tname} (D)$$

d) Insert a new tuple into relation Teacher

$$\text{Teacher} \leftarrow \text{Teacher } U \{ \text{“110”, “Alice “, “COA”} \}$$

e) Delete records of Students whose address is “ Mumbai “

$$\text{Student} \leftarrow \text{Student} - (\sigma_{Address=" Mumbai "} (\text{Student}))$$

Q) Consider the following relational database of a Company

Employee (Fname, Minit, Lname, SSN, Bdate, Address, Sex , Salary, Super_SSN, Dno)

Department (Dname , Dnumber, Mgr_SSN, Mgr_Start_date)

Dept_Location (Dnumber, Dlocation)

Project (Pname, Pnumber, Plocation, Dnum)

Works_On (ESSN, Pno, Hours)

Dependent (ESSN, Dependent_name, Sex, Bdate, Relationship)

Write relational algebra for following statements :

1. Retrieve the name and address of all employees who work for the ‘Research’ department.
2. For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date
3. Make a list of project numbers for projects that involve an employee whose last name is “Smith” , either as a worker or as a manager of the department that controls the project

Answer :

1.

$$RESEARCH\_DEPT \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$$
$$RESEARCH\_EMPS \leftarrow (RESEARCH\_DEPT \bowtie_{Dnumber=Dno} EMPLOYEE)$$
$$RESULT \leftarrow \pi_{Fname,\ Lname,\ Address}(RESEARCH\_EMPS)$$

2.

$$STAFFORD\_PROJS \leftarrow \sigma_{Plocation='Stafford'}(PROJECT)$$
$$CONTR\_DEPTS \leftarrow (STAFFORD\_PROJS \bowtie_{Dnum=Dnumber} DEPARTMENT)$$
$$PROJ\_DEPT\_MGRS \leftarrow (CONTR\_DEPTS \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE)$$
$$RESULT \leftarrow \pi_{Pnumber,\ Dnum,\ Lname,\ Address,\ Bdate}(PROJ\_DEPT\_MGRS)$$

- Here, first we select the projects located in Stafford, then join them with their controlling departments, and then join the result with the department managers.
- Finally, we apply a project operation on the desired attributes

3.

$$SMITHS(Essn) \leftarrow \pi_{Ssn}(\sigma_{Lname='Smith'}(EMPLOYEE))$$
$$SMITH\_WORKER\_PROJS \leftarrow \pi_{Pno}(WORKS\_ON * SMITHS)$$
$$MGRS \leftarrow \pi_{Lname,\ Dnumber}(EMPLOYEE \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)$$
$$SMITH\_MANAGED\_DEPTS(Dnum) \leftarrow \pi_{Dnumber}(\sigma_{Lname='Smith'}(MGRS))$$
$$SMITH\_MGR\_PROJS(Pno) \leftarrow \pi_{Pnumber}(SMITH\_MANAGED\_DEPTS * PROJECT)$$
$$RESULT \leftarrow (SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS)$$

Structured Query Languages
(SQL)

- SQL stands for Structured Query Language
- It is a programming language designed for managing data in relational database
- SQL has a variety of functions that allow its users to read, manipulate and change data
- Though SQL is commonly used by engineers in software development and data analysts because of following reasons :
    - It is semantically easy to understand and learn
    - It can be used to access large amounts of data directly where it's stored , analysts don't have to copy data into other applications
    - Data analysis done in SQL is easy to audit and replicate
    - It is designed to handle complex queries and large datasets with optimal performance
    - It provides a universal method to interact with relational databases across platforms
    - Flexible and Scalable : It can be extended with procedural programing to build complex business logic functions

- SQL languages can be classified into :

    - Data Definition Langauges : It is used to define database structure
    - Data Manipulation Languages : It is used for accessing and manipulating the data in a database
    - Data Control Languages : It is used to retrieve the stored data

Attribute , Datatype, and Domains in SQL

- Basic datatypes available for atributes include Numeric , String, Boolean , Date & Time

1. Numeric data types

- It include integer numbers of various sizes

  INTEGER or INT

  SMALLINT

  FLOAT or REAL

  DOUBLE PRECISION

- Formatted numbers can be declared by using

  DECIMAL(i,j) or DEC(i,j) or NUMERIC(i,j)

  where i, the precision, is the total number of decimal digits and

  j, the scale, is the number of digits after the decimal point.

- The default for scale is zero,and the default for precision is implementation-defined.

2. Character-string data types

• These are either fixed length or varying length

CHAR(n) or CHARACTER(n), where n is the number of characters

VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n),where n is the maximum number of characters.

• When specifying a literal string value,it is placed between single quotation marks (apostrophes), and it is case sensitive (a distinction is made between uppercase and lowercase).

• For fixed length strings, a shorter string is padded with blank characters to the right.

• For example, if the value 'Smith' is for an attribute of type CHAR(10), it is padded with five blank characters to become 'Smith '

3. Boolean data type

• It has the traditional values of TRUE or FALSE.

• In SQL, because of the presence of NULL values, a three-valued logic is used, so a third possible value for a Boolean data type is UNKNOWN.

4. DATE data type

• It has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD.

TIME data type

• It has at least eight positions, with the components HOUR,MINUTE,and SECOND in the form HH:MM:SS.

• Only valid dates and times should be allowed by the SQL implementation.

• This implies that months should be between 1 and 12 and dates must be between 1 and 31

CREATING DOMAIN

• It is possible **to specify the data type of each attribute directly**

• Alternatively, a domain can be declared, and the domain name used with the attribute specification.

• This makes it easier to change the data type for a domain that is used by numerous attributes in a schema, and improves schema readability.

• For example, we can create a domain SSN_TYPE by the following statement:

CREATE DOMAIN < domain name > AS < datatype > ;

Specifying Constraints in SQL.

• These include key and referential integrity constraints, restrictions on attribute domains and NULLs,and constraints on individual tuples within a relation.

1. Specifying Attribute Constraints and Attribute Defaults

- SQL allows NULLs as attribute values,a constraint **NOT NULL** may be specified if NULL is not permitted for a particular attribute.
- This is always implicitly specified for the attributes that are part of the primary key of each relation, but it can be specified for any other attributes whose values are required not to be NULL
- It is also possible to define a default value for an attribute by appending the clause **DEFAULT <value>** to an attribute definition.
- The default value is included in any new tuple if an explicit value is not provided for that attribute
- If no default clause is specified, the default default value is NULL for attributes that do not have the NOT NULL constraint.
- Another type of constraint can restrict attribute or domain values using the **CHECK** clause following an attribute or domain definition

2. Specifying Key and Referential Integrity Constraints

- There are special clauses within the CREATE TABLE statement to specify keys and referential integrity constraints.
- The **PRIMARY KEY** clause specifies one or more attributes that make up the primary key of a relation.
- If a primary key has a single attribute, the clause can follow the attribute directly.
- For example, the primary key of DEPARTMENT can be specified as follows

```
Dnumber  INT  PRIMARY KEY;
```

- The **UNIQUE** clause specifies alternate (secondary) keys
- The UNIQUE clause can also be specified directly for a secondary key if the secondary key is a single attribute,as in the following example:

```
Dname VARCHAR(15) UNIQUE;
```

- Referential integrity is specified via the **FOREIGN KEY** clause
- A referential integrity constraint can be violated when tuples are inserted or deleted, or when a foreign key or primary key attribute value is modified.
- The default action that SQL takes for an integrity violation is to reject the update operation that will cause a violation, which is known as the **RESTRICT** option.
- The schema designer can specify an alternative action to be taken by attaching a referential triggered action clause to any foreign key constraint. The options include **SET NULL, CASCADE, and SET DEFAULT**.

3. Specifying Constraints on Tuples Using CHECK

- Table constraints can be specified through additional **CHECK** clauses at the end of a CREATE TABLE statement.
- These can be called tuple-based constraints because they apply to each tuple individually and are checked whenever a tuple is inserted or modified.

4. Giving Names to Constraints

- A constraint may be given a constraint name, following the keyword **CONSTRAINT**.
- The names of all constraints within a particular schema must be unique.
- A constraint name is used to identify a particular constraint in case the constraint must be dropped later and replaced with another constraint

Data Definition Language (DDL)

- DDL is an SQL command which is **used to define the database schema**
- It allows the specification of information about relations include :
    - Schema for each relation
    - The domain of values associated with each attributes
    - Integrity constraints
- In other words, commands of DDL deals with how the data should exist in the database
- **DDL Commands are :**
    - **CREATE**
    - **ALTER**
    - **DROP**
    - **TRUNCATE**

CREATE : is used to create the database

ALTER : is used to alter the structure of the database

DROP : is used to delete the objects from database

TRUNCATE : Remove all records from table

CREATE TABLE

- A CREATE TABLE statement is **used to create a new table in a database**
- Syntax :

CREATE TABLE < tablename > (A1 D1, A2 D2,….An Dn, Constraint 1, …. Constraint n);

where, Ai is the attribute names of the relation

Di is the datatypes of the attributes

Eg : CREATE TABLE Person (Pid INT, Fname VARCHAR(30), Lname VARCHAR (30),
Salary DECIMAL (8,2));

Integrity Constraints in CREATE TABLE 's are:

- PRIMARY KEY (A1….An) : The attributes required to be non-null and unique

- FOREIGN KEY (A1…An) REFERENCES T : The values of the attributes (A1…An) for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation T

- NOT NULL : Specifies that the null value is not allowed for that attribute

DROP TABLE

• It is **used to delete a whole database (** ie, all things associated with the table are completely deleted)

• Syntax :

DROP TABLE < tablename>;

• Eg : DROP TABLE Person ;

ALTER TABLE

• It is **used to modify the structure of a table**

• **To add new attributes** to existing database :

ALTER TABLE < tablename> ADD (<attributes>);

• **To delete an attribute from table** :

ALTER TABLE <tablename> DROP COLUMN (<columname>);

• **To modify the existing columns** in a table :

ALTER TABLE <tablename> MODIFY (<column name > < column type >)

Example :

```
CREATE DATABASE BandBooking;
CREATE TABLE Band (
BandName varchar2(25),
NumberOfMembers number(1));
ALTER TABLE Band ADD PRIMARY KEY(BandName) ;
ALTER TABLE Band-Booking ADD FOREIGN KEY (BandName
REFERENCES Band (BandName);
```

The above eg shows that , once the database has been created, the tables can be created and the attributes are defined. It is possible to define a Primary key and a foreign key with in the CREATE TABLE command

DATA MANIPULATION LANGUAGE (DML)

- It is used when a database is first created, to populate the tables with data
- It can be then used for the ongoing maintenance
- In simple, **DML is used to manipulate and retrieve the data in database**
- ie, it cannot change the structure of the database , changes made only on the data stored
- DML commands are
    - INSERT
    - DELETE
    - UPDATE
    - SELECT

Apart from these commands, there are also other manipulative operators/functions such as:

- Operators
- Aggregate Functions
- NULL functions
- Aliases & Case Statement

INSERT INTO

• This statement is used to insert new records into the table.

• Here , we must specify the relation name and list of values for the tuple

• The values should be listed in the same order in which corresponding attributes were specified in CREATE TABLE command

• Syntax :

```
INSERT INTO TableName (Column1, Column2, Column3, ...,ColumnN)
VALUES (value1, value2, value3, ...);
```

• If we don't want to mention the column names then use the below syntax

```
INSERT INTO TableName VALUES (Value1, Value2, Value3, ...);
```

Eg :

```
INSERT INTO Employee_Info(EmployeeID, EmployeeName, Emergency ContactName, PhoneNumber, Address, City, Country) VALUES ('06', 'Sanjana','Jagannath', '9921321141', 'Camel Street House No 12', 'Chennai', 'India');

INSERT INTO Employee_Info VALUES ('07', 'Sayantini','Praveen', '9934567654', 'Nice Road 21', 'Pune','India');
```

UPDATE

• This statement is used to modify the records already present in the table.

• **Syntax**

UPDATE TableName SET Column1 = Value1, Column2 = Value2, ... WHERE Condition;

• Eg :

UPDATE Employee_Info SET EmployeeName = 'Aahana', City= 'Ahmedabad'

WHERE EmployeeID = 1;

DELETE

• This statement is used to delete the existing records in a table.

• **Syntax**

DELETE FROM TableName WHERE Condition;

• Eg :

DELETE FROM Employee_Info WHERE EmployeeName='Preeti';