UNIT 2

NULL values:

SQL supports a special value known as NULL which is used to represent the values of attributes that may be unknown or not apply to a tuple. .

• It is important to understand that a NULL value is different from zero value.

• A NULL value is used to represent a missing value, but that it usually has one of three different interpretations:

• Value unknown (value exists but is not known)

• Value not available (exists but is purposely withheld)

• Attribute not applicable (undefined for this tuple)

It is often not possible to determine which of the meanings is intended. Hence, SQL does not distinguish between the different meanings of NULL

Structure of a Relational Databases

A relational Database consists of a collection of tables,each of which is assigned a unique name.

The relational model represents the database as a collection of relations of data .

Formally, in Relational Model,the term relation is used to refer to a table.

Table consists of rows and columns.

Term t**uple** is used to refer to a row in relation.

Term **attribute** is used to refer to a column in relation.

A **primary key** having two or more attributes is called composite key. It is a combination of two or more columns.

A **foreign-key** constraint from attribute(s) A of relation r1 to the primary-key B of relation r2 states that on any database instance, the value of A for each tuple in r1 must also be the value of B for some tuple in r2.

Many users consider Primary Key as Unique Key, since both uniquely identify a table, but Unique Key is different from Primary Key. Unique Key accepts null values and Primary Key cannot have null.

A primary key cannot accept **NULL** values; this makes Primary Key different from Unique Key, since Unique Key allows one value as NULL value.

Database Architectures:

The architecture of a DB system is greatly influenced by the underlying computer system on which the DB systems runs.

- DB systems can be centralized or client-server.
- DB systems can also be designed to exploit parallel computer architectures.
- Distributed DB's span multiple geographically separated machines.

Database Users & Administrators

- A primary goal of a database system is to retrieve information from and store new information in the database.
- People who work with a database can be categorized as database users or DB administrators.

\

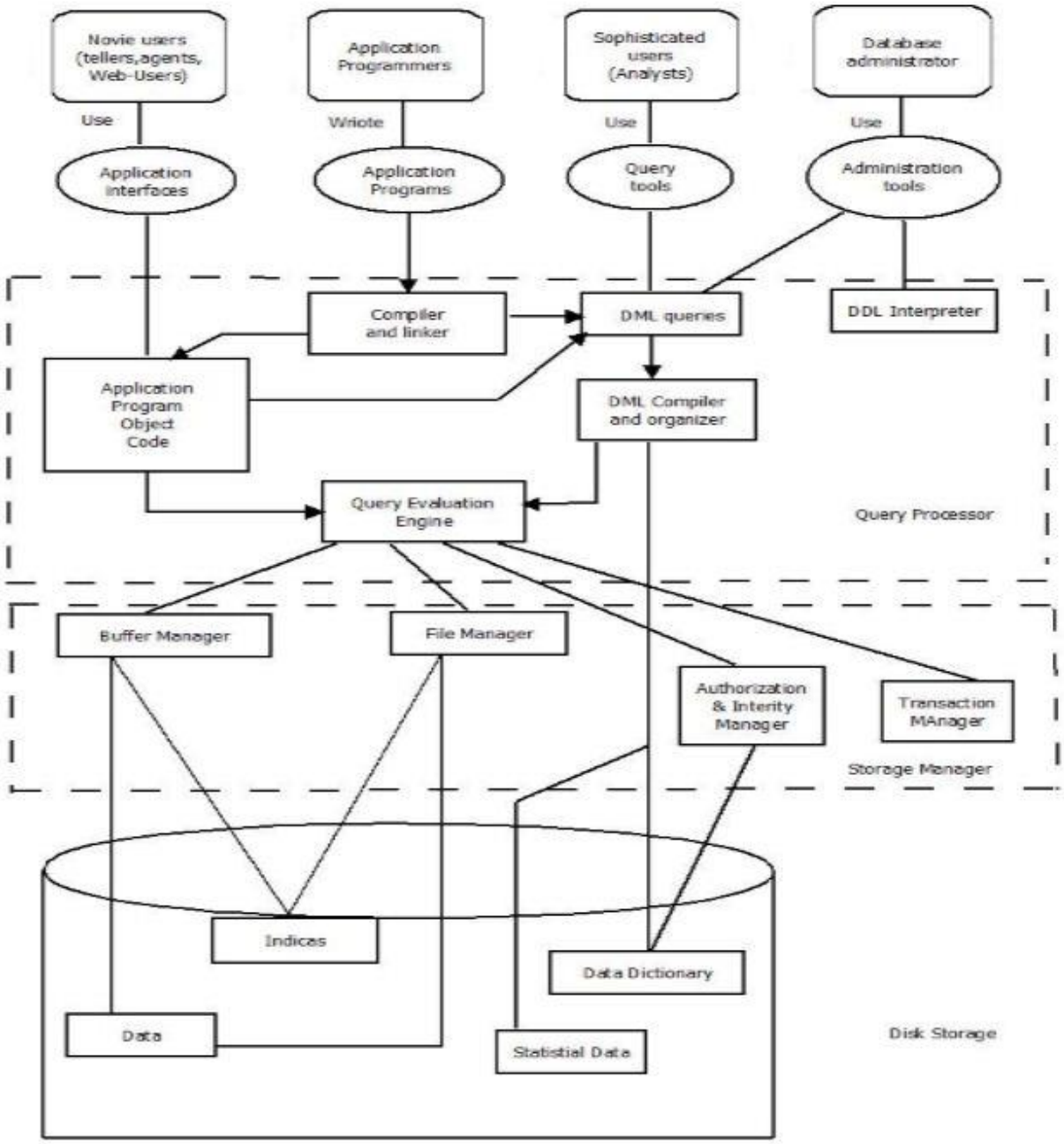


Fig1. Database System Architecture.

1. Naive users

- Unsophisticated users.
- Interact with the system by using predefined user interfaces, such as web or mobile applications.
- The typical user interface for naive users is a forms interface

2. Application programmers

- computer professionals who write application programs.
- Application programmers can choose from many tools to develop user interfaces.

3. Sophisticated users

- Interact with the system without writing programs.
- Form their requests either using a database query language or by using tools such as data analysis software.
- Analysts who submit queries to explore data in the database fall in this category.

4. Specialized Users

- Sophisticated users who write specialized DB applications that do not fit into the traditional data-processing framework.

Database Administrator

- One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data.
- A person who has such central control over the system is called a database administrator (DBA).

UNION: The UNION operator is used to combine the result-set of two or more SELECT statements.

Each SELECT statement within UNION must have the same number of columns

The columns must also have similar data types

The columns in each SELECT statement must also be in the same order

Union compatibility: Two relations R and S , are said to be UNION COMPATIBLE, only if they satisfy these two conditions:

- Both the relations , i.e. R and S should be of same arity, i.e. they should have same number of attributes.
- Domain of the similar attributes, i.e. dom

A) Attribute:it refers to the properties that characterize an entity set. Attribute also known as data field or data element.
B) Entity: Entity in DBMS can be a real-world object with an existence, For example, in a College database, the entities can be Professor, Students, Courses, etc.
C)Entityset:Entity is an object of Entity Type and set of all entities is called as entity set. e.g.;E1 is an entity having Entity Type Student and set of all students is called Entity Set.
D) Relationship set:set of relationships of same type is known as relationship set

Cardinality ratio specify

Number of relationship instances in which the entity can participate is defined as

Cardinality Ratio . There are three possible cardinality ratios , they are:

1. One to One Relationship (1:1)

2. One to Many Relationship (1:N)

3. Many to One Relationship (N:N)

STEPS IN DATABASE DESIGN.

User requirements: First we need them, we need a user capable of using a Computer, understanding our language, etc.

Conceptual design: This is used to describe information that the DB will contain. This is

Where we typically use the ER (Entity - Relationship) model, although some people refer to

This design as drawing a simple schema about relationships without using the full version of

ER model.

Input: Requirement specifications to build the DB

Output: Conceptual schema

Logical design: Used to describe the structure of the DB that can be processed by the DBMS

(Database Management System), it can depend on the type of data the DBMS can used but

Not on the DBMS itself. Here we'll use the ER, including Attributes and Primary, Foreign an

Alternative keys.

Input: Conceptual schema

Output: Logical schema

Physical design. At this point we create the file index and the design of the filesystem.

Implementation. We test the DB.

Components of ERD:

Rectangles: represents the entity sets.

Ellipses: represents the attributes.

Diamonds: represents the relationship sets.

Lines: link the attributes to entity sets and entity sets to relationship sets.

Double ellipses: represents the multivalued attributes.

Dashed ellipses: represents derived attributes.

Doubles lines: indicate the total participation of an entity in a relationship set.

Double rectangle: represents the weak entity sets

Example:

Consider an example - a person (name, street, city) be an entity set which is further classified among “customer” and “employee”.

Each of these person type is describe by a set of attributes that include all the attribute of entity set “person” plus possible additional attribute like “customer id” of “customer” entity. “employee id”, “salary” of “employee” entity.

Extended / Additional E-R Features:

1.Specialization – The process of designating to sub grouping within an entity set is called specialization.

2. Generalization – generalization is relationship that exist between higher level entity set and one or more lower level entity sets. Generalization synthesizes these entity sets into single entity set.

3. Higher level and lower level entity sets – This property is created by specialization and generalization. The attributes of higher level entity sets are inherited by lower level entity sets.

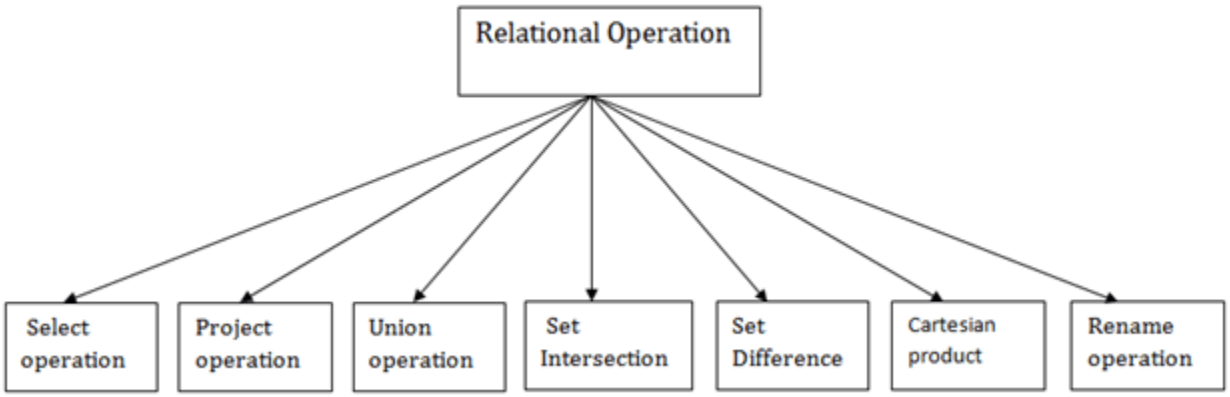
4. Attribute inheritance: When given entity set is involved as a lower entity set in only one "ISA" (is a) relationship, it is referred as a single attribute inheritance. If lower entity set is involved in more than one ISA (is a) relationship, it is referred as a multi attribute inheritance.

Aggregation: there is a one limitation with E-R model that it cannot express relationships among relationships. So aggregation is an abstraction through which relationship is treated as higher level entities.

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. **Notation: σ p(r)**

Where: **σ** is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT

Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

- It is denoted by ∏.

Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93

Smith	L-11
Williams	L-17

Input::

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a **cross product**.
- It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing

3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

Example:

EMPLOYEE

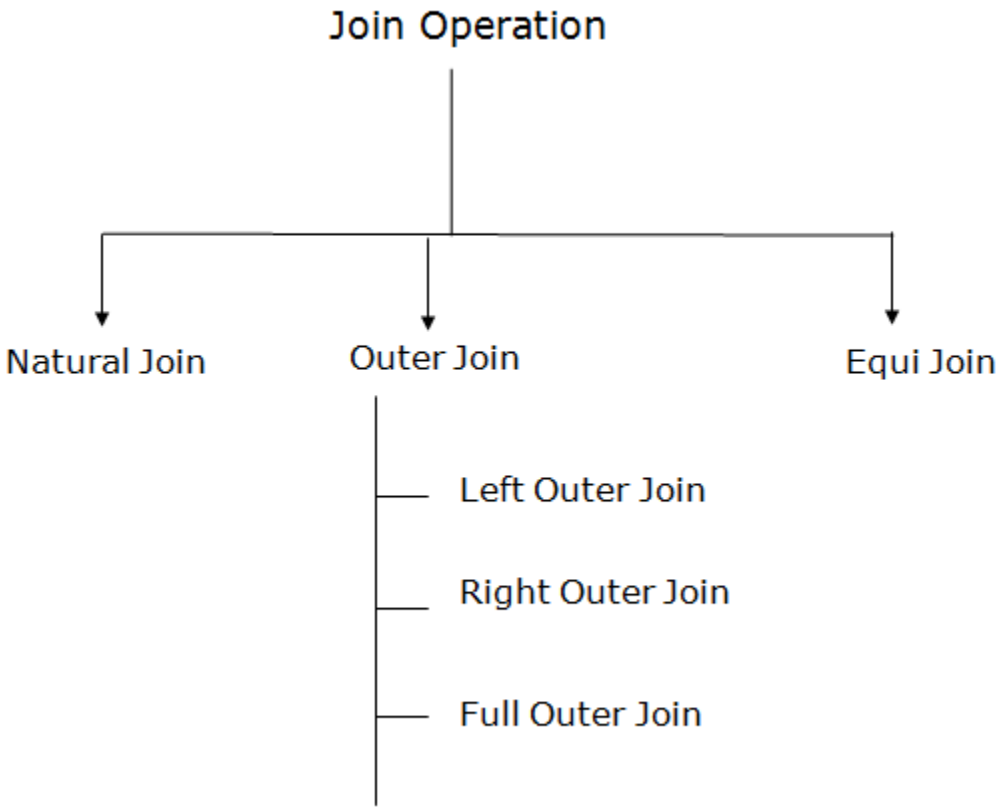
EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

Operation: (EMPLOYEE ⋈ SALARY)

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000


Join Operation
Natural Join
Outer Join
Equi Join
Left Outer Join
Right Outer Join
Full Outer Join

1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.

Example: Let's use the above EMPLOYEE table and SALARY table:

Input:

1. ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

The outer join operation is an extension of the join operation. It is used to deal with missing information.

Example:

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai

Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input:

1. (EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

An outer join is basically of three types:

a. Left outer join
b. Right outer join
c. Full outer join

- LEFT OUTER JOIN - keep data from the left-hand table
- RIGHT OUTER JOIN - keep data from the right-hand table
- FULL OUTER JOIN - keep data from both tables

Left outer join:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

Right outer join:

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

Full outer join:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000