**DBMS 2nd Internal**

Answer all five questions each carry two marks

1. Define entity and attributes?

Ans. An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

1. Define relationship type?

A Relationship Type is **a type of association that can exist between two different (or same) entity types**.

For example, a person can manage another person, or an organization can legally own another organization.

There are 3 relationship type

* One-to-One Relationship
* One-to-Many or Many-to-One Relationship
* Many-to-Many Relationship

1. Define the term Domain, Attribute, Tuple & Relation.

A domain is **a unique set of values that can be assigned to an attribute in a database**. For example, a domain of strings can accept only string value

Attributes **stores values that are used to describe the entity**. The attributes which are able to store only one value are known as Single Valued Attributes. These attributes cannot store more than one value. For example, The attributes of an Employee entity are Employee id, DOB, Gender

Relationa An relationship exists between two tables when one of them has a foreign key that references the primary key of the other table

1. What is Domain Constraint?

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

**Example:**

Create DOMAIN CustomerName

CHECK (value not NULL)

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

1. List Relational Algebra Operations From Set Theory

* Unary Relational Operations

SELECT (symbol: **s** (sigma))

PROJECT (symbol: **p** (pi))

RENAME (symbol: **ρ** (rho))

* Relational Algebra Operations From Set Theory

UNION ( **È** ), INTERSECTION ( **Ç** ), DIFFERENCE (or MINUS, **–** )

CARTESIAN PRODUCT ( **x** )

* Binary Relational Operations

JOIN (several variations of JOIN exist)

* Additional Relational Operations

OUTER JOINS

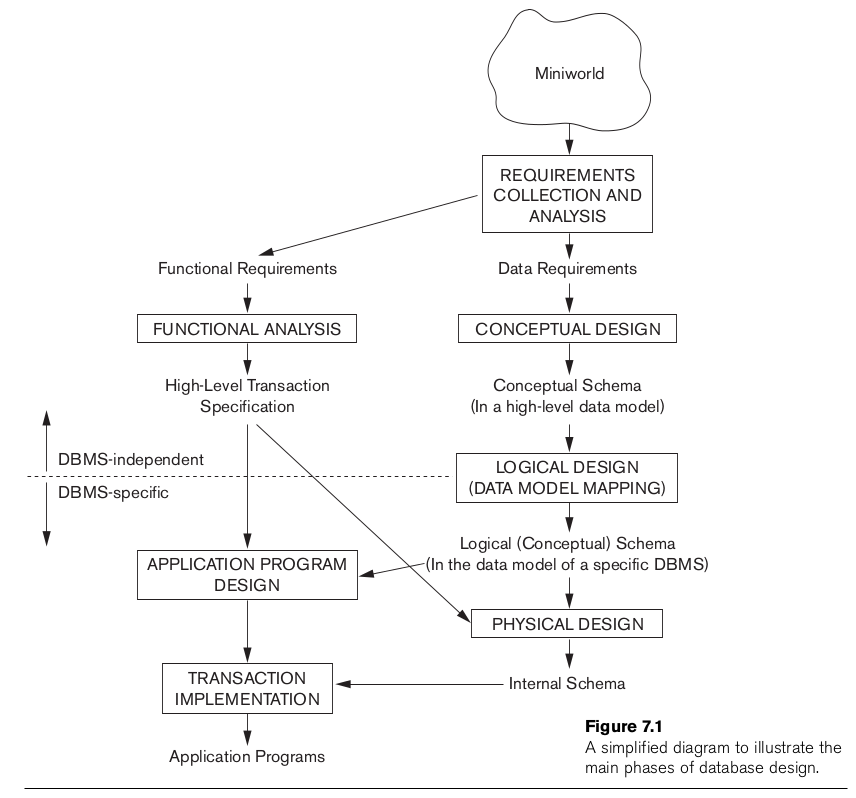
INNER JOINS

1. Define group of clause?

 The GROUP BY clause **groups the selected rows based on identical values in a column or expression**. This clause is typically used with aggregate functions to generate a single result row for each set of unique values in a set of columns or expressions

Answer any four questions each carry five marks

1. Draw a neat labeled diagram and explain the main phases of database design?



This phase consists of three parts: **the conceptual design, the logical design and the physical design**. Some methodologies merge the logical design phase into the other two phases.

**The design process consists of the following steps:**

* Determine the purpose of your database. ...
* Find and organize the information required. ...
* Divide the information into tables. ...
* Turn information items into columns. ...
* Specify primary keys. ...
* Set up the table relationships. ...
* Refine your design. ...
* Apply the normalization rules.

1. Explain the concept of joins?

Ans)Joins can be simply defined as the combining or merging the related tuples from the two different relations into a single type.

* The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is
* 

Types of SQL JOIN

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

## Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

Sample Table

**EMPLOYEE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angelina | Chicago | 200000 | 30 |
| 2 | Robert | Austin | 300000 | 26 |
| 3 | Christian | Denver | 100000 | 42 |
| 4 | Kristen | Washington | 500000 | 29 |
| 5 | Russell | Los angels | 200000 | 36 |
| 6 | Marry | Canada | 600000 | 48 |

**PROJECT**

52.4M

1.1KFeatures of Java

|  |  |  |
| --- | --- | --- |
| **PROJECT\_NO** | **EMP\_ID** | **DEPARTMENT** |
| 101 | 1 | Testing |
| 102 | 2 | Development |
| 103 | 3 | Designing |
| 104 | 4 | Development |

1. INNER JOIN

**In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.**



**Syntax**

1. **SELECT table1.column1, table1.column2, table2.column1,....**
2. **FROM table1**
3. **INNER JOIN table2**
4. **ON table1.matching\_column = table2.matching\_column;**

**Query**

1. **SELECT EMPLOYEE.EMP\_NAME, PROJECT.DEPARTMENT**
2. **FROM EMPLOYEE**
3. **INNER JOIN PROJECT**
4. **ON PROJECT.EMP\_ID = EMPLOYEE.EMP\_ID;**

**Output**

|  |  |
| --- | --- |
| **EMP\_NAME** | **DEPARTMENT** |
| Angelina | Testing |
| Robert | Development |
| Christian | Designing |
| Kristen | Development |

2. LEFT JOIN

**The SQL left join returns all the values from left table and the matching values from the right table. If there is no matching join value, it will return NULL. In some databases LEFT JOIN is called LEFT OUTER JOIN.**



**Syntax**

1. **SELECT table1.column1, table1.column2, table2.column1,....**
2. **FROM table1**
3. **LEFT JOIN table2**
4. **ON table1.matching\_column = table2.matching\_column;**

**Query**

1. **SELECT EMPLOYEE.EMP\_NAME, PROJECT.DEPARTMENT**
2. **FROM EMPLOYEE**
3. **LEFT JOIN PROJECT**
4. **ON PROJECT.EMP\_ID = EMPLOYEE.EMP\_ID;**

**Output**

|  |  |
| --- | --- |
| **EMP\_NAME** | **DEPARTMENT** |
| Angelina | Testing |
| Robert | Development |
| Christian | Designing |
| Kristen | Development |
| Russell | NULL |
| Marry | NULL |

3. RIGHT JOIN

**In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.**



**Syntax**

1. **SELECT table1.column1, table1.column2, table2.column1,....**
2. **FROM table1**
3. **RIGHT JOIN table2**
4. **ON table1.matching\_column = table2.matching\_column;**

**Query**

1. **SELECT EMPLOYEE.EMP\_NAME, PROJECT.DEPARTMENT**
2. **FROM EMPLOYEE**
3. **RIGHT JOIN PROJECT**
4. **ON PROJECT.EMP\_ID = EMPLOYEE.EMP\_ID;**

**Output**

|  |  |
| --- | --- |
| **EMP\_NAME** | **DEPARTMENT** |
| Angelina | Testing |
| Robert | Development |
| Christian | Designing |
| Kristen | Development |

4. FULL JOIN

**In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found. The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.**

**Tip:** **FULL OUTER JOIN and FULL JOIN are the same.**



**Syntax**

1. **SELECT table1.column1, table1.column2, table2.column1,....**
2. **FROM table1**
3. **FULL JOIN table2**
4. **ON table1.matching\_column = table2.matching\_column;**

**Query**

1. **SELECT EMPLOYEE.EMP\_NAME, PROJECT.DEPARTMENT**
2. **FROM EMPLOYEE**
3. **FULL JOIN PROJECT**
4. **ON PROJECT.EMP\_ID = EMPLOYEE.EMP\_ID;**

**Output**

|  |  |
| --- | --- |
| **EMP\_NAME** | **DEPARTMENT** |
| Angelina | Testing |
| Robert | Development |
| Christian | Designing |
| Kristen | Development |
| Russell | NULL |
| Marry | NULL |

1. Explain the characteristics of relations?

* Ordering of tuples in a relation
* Ordering of values within a tuple
* Values and NULLs in the tuples
* Interpretation of a relation
  1. Ordering of tuples in a relation
* A relation is defined as set of tuples.
* Mathematically, elements of a set have no order among them; hence tuples in a relation do not have any particular order.
* A relation is not sensitive to the ordering of tuples.

2. Ordering of values within a tuple

* According to the previous definition of a relation, an n-tuple is an ordered list of n values, the ordering of values in a tuple and attributes in a relation schema is important.
* However, at more abstract level, the order of attributes and their values is not that important as long as the correspondence between attributes and values is maintained

3.Values and NULL’s in the Tuple

* Each value in a tuple is atomic value. [ composite and multivalued attributes are not allowed].
* This model is sometimes called as **flat relational model.**
* The NULL values are used to represent the values of attributes that may be unknown or may not apply to a tuple.
* Several meaning for NULL values: Value unknown, value exists but not available, attribute doesn’t apply.

4.Interpretation of a Relation

* The relation schema can be interpreted as declaration or a type of assertion.
* For example, the schema of the STUDENT relation asserts that, a student entity has a Name, Roll#, Mobile.. Etc attributes.
* Each tuple in the relation can then be interpreted as a **fact** or a particular instance of the assertion.
* For example, the first tuple in the STUDENT relation asserts the fact that there is a student whose name is Prakash, Roll# 10 and etc.

4.Explain the concept of set theory in relational algebra?

Ans)

* Type Compatibility of operands is required for the binary set operation UNION È, INTERSECTION Ç, and SET DIFFERENCE –
* R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  + they have the same number of attributes, and
  + the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
* The resulting relation for R1ÈR2 also for R1ÇR2, or R1–R2 has the same attribute names as the *first* operand relation R1
  + UNION Operation
  + Binary operation, denoted by È
  + The result of R È S, is a relation that includes all tuples that are either in R or in S or in both R and S
  + Duplicate tuples are eliminated
  + The two operand relations R and S must be “type compatible” (or UNION compatible)
* INTERSECTION is denoted by Ç

The result of the operation R Ç S, is a relation that includes all tuples that are in both R and S

The attribute names in the result will be the same as the attribute names in R

* SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –

The result of R – S, is a relation that includes all tuples that are in R but not in S

The attribute names in the result will be the same as the attribute names in R

5.Draw a ER diagram of a COMPANY database [consider all The necessaryConstraints]

Ans)

