DBMS CT-1 QnA

Discuss the importance of DML pre-compiler. 2

Ans :-

DML Pre-compiler : It translates DML statements in a query language into low level instructions that query evaluation engine understands. It also attempts to transform user's request into an equivalent but more efficient form.

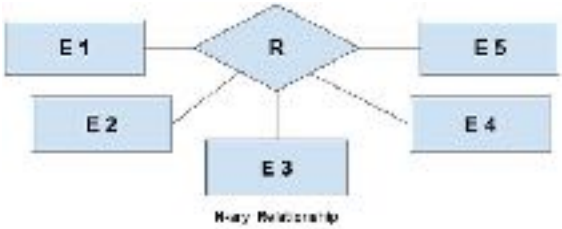
Embedded DML Pre-compiler : It converts DML statements embedded in an application program to normal procedure calls in the host language. The Pre-compiler must interact with the DML compiler to generate the appropriate code.

Define relation state of n-degree relation according to relational model. 2

Ans :-

An N-ary relationship exists when ‘n’ number of entities are participating. So, any number of entities can participate in a relationship. There is no limitation to the maximum number of entities that can participate. But, relations with a higher degree are not common. This is because the conversion of higher degree relations to relational tables gets complex. We are making an E-R model because it can be easily be converted into any other model for implementing the database. But, this benefit is not available if we use higher degree relations. So, binary relations are more popular and widely used. Though we can make a relationship with any number of entity types but we don't do that.

We represent an N-ary relationship as follows:



In the above example, E1 denotes the first entity type, E2 denotes the second entity type and so on. R represents the relationship. So, here we have a total of 5 entity type which participates in the relationship. Therefore, the degree of the above n-ary relationship is 5.

What are the model level constraints in relational model? 2

Ans:-

We have three relational model constraints: **inherent model-based constraints (implicit constraints)**, **schema-based constraints (explicit constraints)** and **application-based constraints (business rules).**

1. Inherent model-based constraints

For example, **a relation can not have duplicate tuples.**

2. Schema-based constraints

We specify constraints directly to the relational schemas using DDL (Data Definition Language). Schema-based constraints include four constraints.

They are **domain constraints, key constraints, constraints on NULL, entity integrity constraints, and referential integrity constraints.**

Referential integrity constraints are specified for two relations while others are for single relations.

I. Domain constraints

Domain constraints specify that each tuple must contain atomic values.

II. Key constraints

uniqueness constraint is specified by superkey: two different tuples in a relation can not have the same value for superkey. Every relation, we'll have at least one superkey: a set of all the attributes.

Because of superkey's uniqueness constraint, it uniquely identifies each tuple in a relation.

A key/ super key should maintain its uniqueness when we insert new tuples in a relation. For example, we can not take name attribute as a key in a student relation. Because over time, there will be a situation of more than a student having the same name.

III. Constraints on NULL

We can specify whether an attribute can have NULL or not using this constraint. For example, if we do not want a NULL value for the student's name then we can specify and constraint it using NOT NULL.

IV. Entity integrity constraint

This constraint specifies that a primary key can not contain NULL values. Primary keys are used to identify each tuple uniquely. If more than a tuple contain NULL as their primary key then it is difficult to identify those tuples uniquely.

V. Referential integrity constraint

The referential integrity constraint is specified between two relations. It helps us to maintain consistency among the tuples in those two relations.

It references a tuple in one relation to an existing tuple in another relation via foreign key.

Child table: table which contains the foreign key.

3. Application-based constraints (business rules)

We specify constraints to the relational schemas using application programs.

For example, in a student table, we can specify our business rules such as students should not have age more than 19. Student can not have a phone number more than 10 digits.

NOTE: Data dependencies are also considered as constraints. It contains functional and multivalued dependencies. These two are used in normalization. Normalization helps us to maintain the database from redundancy, and insertion, update, deletion anomalies.

Specify the impact of referential integrity on DML operation with referenced relation. 1.5

Ans:-

Referential integrity is a database constraint that ensures that relationships between tables are valid and consistent. Specifically, it ensures that foreign key values in a table reference primary key values in another table.

The impact of referential integrity on DML (Data Manipulation Language) operations with referenced relation is that it prevents certain actions that would violate the referential integrity constraint. For example:

Insert: If you try to insert a row into a table that contains a foreign key referencing a primary key in another table, the insert operation will fail if the referenced primary key does not exist in the referenced table.

Update: If you try to update a row in a table that contains a foreign key referencing a primary key in another table, the update operation will fail if the referenced primary key is updated in a way that changes its value or deletes the row.

Delete: If you try to delete a row from a table that contains a primary key referenced by a foreign key in another table, the delete operation will fail if there are any rows in the referencing table that reference the primary key being deleted.

In other words, referential integrity ensures that DML operations with referenced relations do not create inconsistencies or errors in the database. It helps to maintain the accuracy and consistency of the data, and prevents situations where related data becomes out of sync or invalid.

When it is possible that (VX)(NOT P(x)) implies (3x)(Q(x))? Also write the equivalent expression for (Vx)(NOT P(x)) avoiding universal quantifier. 2

Ans:-

The statement (VX)(NOT P(x)) implies (3x)(Q(x)) is known as the contrapositive of the statement (3x)(NOT Q(x)) implies (Vx)(P(x)). These two statements are logically equivalent, so if one is true, then the other must also be true.

To see why the contrapositive is true, suppose that (VX)(NOT P(x)) is true. This means that there exists at least one x such that NOT P(x) is true. Since this negation is true, it follows that P(x) is false for this particular x. Therefore, we have shown that there exists an x such that NOT P(x) is true, which implies that (3x)(NOT P(x)) is true.

Now, we can apply the hypothesis that (3x)(NOT Q(x)) implies (Vx)(P(x)). If (3x)(NOT P(x)) is true, then (Vx)(P(x)) must also be true. But since we have already shown that there exists an x such that NOT P(x) is true, it follows that (Vx)(NOT P(x)) is true. Therefore, we have shown that (Vx)(NOT P(x)) implies (3x)(Q(x)).

The equivalent expression for (Vx)(NOT P(x)) without using the universal quantifier is (NOT (3x)(P(x))). This can be read as "there does not exist an x such that P(x) is true".

Compare full outer join and outer union. 1.5

Ans:-

Full outer join and outer union are two types of set operations used in relational databases. Both operations are used to combine two tables by matching rows based on a specified condition.

The key difference between a full outer join and an outer union is that a full outer join combines all rows from both tables, including those that do not match, while an outer union combines only distinct rows from both tables.

A full outer join returns all rows from both tables, along with NULL values for any columns that do not match. This means that even if there is no match between the two tables, the rows are still included in the result set. The result of a full outer join includes all rows from the left table, all rows from the right table, and any rows where there is a match between the two tables. If there is no match, the result set will contain NULL values for the columns that do not match.

On the other hand, an outer union only combines distinct rows from both tables. This means that duplicate rows are eliminated, and only distinct rows are included in the result set. An outer union does not include rows that do not match between the two tables, and only includes rows that match the specified condition.

In summary, a full outer join returns all rows from both tables, including non-matching rows, while an outer union only returns distinct rows from both tables that match the specified condition.

Consider the relations: STUDENT (ROLL NAME, ADDRESS), SUBJECT (SCODE, SNAME) and RESULT (ROLL, SCODE, SCORE). They contain information of all the students, information of all the subjects and marks details (only for those appeared in the examination of corresponding subject) respectively.
Write down the relational algebra **and relational calculus expressions to find out name of the subjects in which all the students have appeared in the exam. 2+2**

Ans:-
Relational Algebra Expression:

πSNAME((πSCODE(COUNT(DISTINCT ROLL),SCODE)÷COUNT(DISTINCT ROLL))(RESULT⋈SUBJECT))

Explanation:

RESULT⋈SUBJECT : This operation joins the RESULT and SUBJECT relations on their common attribute SCODE, resulting in a new relation with attributes ROLL, SCODE, SCORE and SNAME.

πSCODE(COUNT(DISTINCT ROLL),SCODE) : This operation groups the resulting relation from step 1 by SCODE, and counts the number of distinct ROLL values for each group.

(πSCODE(COUNT(DISTINCT ROLL),SCODE)÷COUNT(DISTINCT ROLL)) : This operation computes the ratio of the count of distinct ROLL values for each SCODE to the total number of distinct ROLL values in the RESULT relation. This will result in a relation with attributes SCODE and a computed attribute that gives the ratio for each SCODE.

πSNAME : This operation projects the SNAME attribute from the SUBJECT relation.

The final result will be the set of SNAME values for those subjects where the ratio computed in step 3 is equal to 1, indicating that all the students have appeared in the exam for that subject.

Relational Calculus Expression:

{t.SNAME | SUBJECT(t) ∧ ∀s STUDENT(s) ∧ ∃r RESULT(r) ∧ s.ROLL=r.ROLL ∧ r.SCODE=t.SCODE}

Explanation:

SUBJECT(t) : This expression selects tuples from the SUBJECT relation and assigns them to variable t.

STUDENT(s) : This expression selects tuples from the STUDENT relation and assigns them to variable s.

∃r RESULT(r) ∧ s.ROLL=r.ROLL ∧ r.SCODE=t.SCODE : This expression checks if there exists a tuple r in the RESULT relation such that r.ROLL is equal to s.ROLL and r.SCODE is equal to t.SCODE.

The final result will be the set of SNAME values for those tuples t in the SUBJECT relation where the expression in step 3 is true for all tuples s in the STUDENT relation.

Consider 1:N binary relation between two entity types A and B. Database is to be designed so that related pair of entities can be retrieved and in this effort null value is to be avoided. None of the entity type participates in relation totally. Explain, how will you design the database? 3

Ans:-

To design a database for a 1:N binary relation between two entity types A and B, we can create two separate tables, one for each entity type, and then establish a foreign key constraint in the table for entity type B that references the table for entity type A. This will ensure that only valid references are allowed and null values are avoided.

The table for entity type A will contain a primary key column (let's call it AID) and any other relevant attributes for entity type A. The table for entity type B will contain a primary key column (let's call it BID), any other relevant attributes for entity type B, and a foreign key column (let's call it AID) that references the AID column in the table for entity type A.

To retrieve related pairs of entities, we can join the two tables on the foreign key constraint. For example, if we want to retrieve all entities of type B that are related to an entity of type A with AID = X, we can execute the following **SQL query:**

SELECT * FROM B WHERE AID = X

This query will return all entities of type B that have a foreign key reference to an entity of type A with AID = X.

By using a foreign key constraint, we can ensure that only valid references are allowed and null values are avoided. If an entity of type B is created without a corresponding entity of type A, the foreign key constraint will prevent it from being inserted into the database. Similarly, if an entity of type

A is deleted, all related entities of type B will also be deleted due to the cascading effect of the foreign key constraint. This ensures data consistency and integrity.

A, B and C are three related entity types. To maintain their association one can have either, set of pair wise binary relations or a ternary relation or both. Discuss the implications of the three cases. 4

Ans:-

There are three ways to maintain the association between three related entity types A, B, and C:

Set of Pairwise Binary Relations:

In this approach, we can create two binary relations: A-B and B-C. This means that there will be two tables, one for the relation between A and B, and another for the relation between B and C. The A-B relation table will have foreign key columns referencing the primary keys of the tables for entity types A and B, respectively. Similarly, the B-C relation table will have foreign key columns referencing the primary keys of the tables for entity types B and C, respectively. To retrieve all related entities, we will need to join these tables using their foreign key constraints.

Implications:

The advantage of this approach is that it is simple and easy to understand. We can use familiar SQL join statements to retrieve related entities. However, if there are many relationships between entities, it may result in a large number of tables, which can make the database more complex and difficult to manage.

Ternary Relation:

In this approach, we can create a single ternary relation that contains the primary keys of all three entity types: AID, BID, and CID. The ternary relation table will have foreign key columns referencing the primary keys of the tables for entity types A, B, and C, respectively. To retrieve all related entities, we can use a single join statement that joins the ternary relation table with the tables for entity types A, B, and C.

Implications:

The advantage of this approach is that it is more efficient than the pairwise binary relation approach, as it requires fewer tables and join operations. However, it can be more complex to understand, as it involves working with a single table that contains multiple foreign key constraints.

Both Pairwise Binary Relations and Ternary Relation:

In this approach, we can create both pairwise binary relations and a ternary relation. The pairwise binary relations will provide a simple way to retrieve related entities between two entity types, while the ternary relation will provide a more efficient way to retrieve related entities between all three entity types.

Implications:

The advantage of this approach is that it combines the simplicity of pairwise binary relations with the efficiency of a ternary relation. However, it can make the database more complex, as it involves working with both pairwise binary relations and a ternary relation. Also, the use of both approaches together can lead to redundancy and inconsistencies, which can impact data integrity.

Define multi-valued attribute. 1

Ans:-

A multi-valued attribute is an attribute in a relational database schema that can have multiple values for a single entity. Unlike a simple attribute that contains a single value, a multi-valued attribute can contain more than one value, which may or may not be related to each other.

For example, consider a database schema for a music store. Each album in the store may have multiple genres associated with it, such as rock, pop, and alternative. In this case, the "genre" attribute is a multi-valued attribute, as it can contain multiple values for a single album.

Multi-valued attributes are typically represented in a separate table, where each row corresponds to a single value of the attribute, along with a foreign key that identifies the entity to which the value belongs. This allows for efficient storage and retrieval of multi-valued attributes, as well as the ability to perform operations such as searching for entities based on their multi-valued attributes.

Multi-valued attributes can also be nested within other attributes, such as arrays or JSON objects, depending on the database system and modeling approach used. However, regardless of the specific implementation, multi-valued attributes allow for more flexible and expressive database schema designs that can better capture complex real-world scenarios.

What is identifying relation in ER model? 2

Ans:-

In an ER (Entity-Relationship) model, an identifying relationship is a relationship between two entities where the primary key of one entity is used as a foreign key in the other entity, thereby establishing a direct relationship between them.

Identifying relationships are also known as strong relationships, because they provide a strong link between the two entities. In an identifying relationship, the primary key of the parent entity is used as a foreign key in the child entity, indicating that the child entity is dependent on the parent entity for its existence.

For example, consider a database for a university, where each student is assigned to a specific department. The relationship between the Student and Department entities is an identifying relationship because each student is associated with exactly one department, and the student's primary key (such as a student ID) is used as a foreign key in the Department entity.

Identifying relationships are often used in situations where a child entity cannot exist without the parent entity, and where the child entity's primary key is dependent on the parent entity's primary key. They help to enforce referential integrity between the two entities, ensuring that data is accurate and consistent.

ATTENDANCE(ROLL, SCODE) will store the attendance of the students appearing for examination
in different subjects. RESULT(ROLL, SCODE, SCORE) will store score of the students in different subjects provided he/she has appeared in the corresponding examination for the subject. Write down the SQL statements to create the tables. Assume, suitable type for the attributes. 5

Ans:-

Here are the SQL statements to create the tables:

```
CREATE TABLE ATTENDANCE (
  ROLL INT NOT NULL,
  SCODE VARCHAR(10) NOT NULL,
  PRIMARY KEY (ROLL, SCODE)
);

CREATE TABLE RESULT (
  ROLL INT NOT NULL,
  SCODE VARCHAR(10) NOT NULL,
  SCORE INT NOT NULL,
  PRIMARY KEY (ROLL, SCODE),
  FOREIGN KEY (ROLL) REFERENCES ATTENDANCE (ROLL),
  FOREIGN KEY (SCODE) REFERENCES ATTENDANCE (SCODE)
);
```

In the above SQL statements, we are creating two tables, ATTENDANCE and RESULT. The ATTENDANCE table has two attributes, ROLL and SCODE, which together form the primary key of the table. The data type for ROLL is INT, while the data type for SCODE is VARCHAR(10).

The RESULT table also has three attributes, ROLL, SCODE, and SCORE, with ROLL and SCODE together forming the primary key. The SCORE attribute stores the score obtained by the student in the corresponding subject. The RESULT table also includes foreign key constraints that reference the primary key of the ATTENDANCE table to ensure that only valid combinations of ROLL and SCODE are present in the RESULT table.