International Islamic University Chittagong (IIUC)

Department of Computer and Communication Engineering

Program: **B.sc (Eng.)**
Course Code: **CCE–3505**
Type: **Theory**
Segment: **Final Term-Segment 4**

Baizid MD Ashadzzaman
Designation: **Adjunct Lecturer**
Email: **baizid.md.ashadzzaman@gmail.com**
Phone: **8801862420119**

Course Contents:

Contents	Note
Database Design and the E-R Model: Overview of the Design Process, Entity-Relationship Model, Constraints, Removing Redundant Attributes in Entity Sets, Entity-Relationship Diagrams, Reduction to Relational Schemas, Entity-Relationship Design Issues, Extended E-R Features, Alternative Notations for Modeling, Data, Other Aspects of Database Design.	

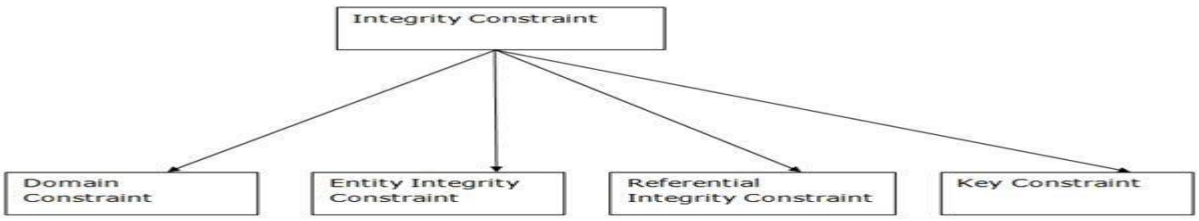
Integrity, Security and Relational Database Design

Integrity constraints:

Integrity constraints are a set of rules that are applied on the attributes to ensure that the overall validity and consistency of the data present in a database. It is used to maintain the quality of information.

Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.

Types of Integrity Constraint



Domain constraints

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.

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

Entity integrity constraints

The entity integrity constraint states that primary key value can't be null. This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows. A table can contain a null value other than the primary key field.

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

Referential Integrity Constraints

A referential integrity constraint is specified between two tables. In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

Key constraints

Keys are the entity set that is used to identify an entity within its entity set uniquely.

An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique value in the relational table.

Example:

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

Operation of Integrity constraints

Not Null Constraint

The not null constraint tells a column that it can't have any null values in it. This is also a type of integrity constraint.

Unique Key

This is also a type of integrity constraint. It can only accept one null value and cannot have duplicate values.

The check Clause

check clause is a predicate or condition that is used to ensure that attribute values satisfy specified conditions Example:

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE ,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
   CHECK (Age>=18) );
```

Assertion:

In SQL, an assertion is a statement that ensures that the certain conditions are always enforced in the database .An 'ASSERTION' in SQL is a predicate expressing a condition that we demand a database to always satisfy. Assertions are used to ensure the integrity of a database.

Example 1: Create an assertion called **"salary_assertion"** that checks that no employee in the "employees" table has a salary greater than $100,000, you could use the following statement:

```
CREATE ASSERTION salary_assertion
CHECK (salary <= 100000);
```

Example 2: Suppose you want to ensure that no employee has a salary higher than the maximum salary defined for their department.

```
CREATE ASSERTION MaxSalaryConstraint
        CHECK ( NOT EXISTS
```

(SELECT 1 FROM Employees e WHERE e.Salary > (SELECT MAX(Salary) FROM Employees

WHERE DepartmentID = e.DepartmentID)));

Privileges:

In a database management system (DBMS), privileges are permissions granted to users or roles to perform certain actions on database objects such as tables, views, procedures, and so on. These privileges control what users can do within the database.

Here are some common privileges in a typical database system:

SELECT: Allows users to retrieve data from a table or view.

INSERT: Permits users to add new rows of data into a table.

UPDATE: Enables users to modify existing rows of data in a table.

DELETE: Grants users the ability to remove rows of data from a table.

CREATE: Allows users to create new database objects such as tables, views, indexes, or procedures.

ALTER: Permits users to modify the structure of existing database objects.

DROP: Grants users the ability to delete database objects.

GRANT: Allows users to grant privileges to other users or roles.

REVOKE: Enables users to revoke privileges previously granted to other users or roles.

Encryption and Decryption

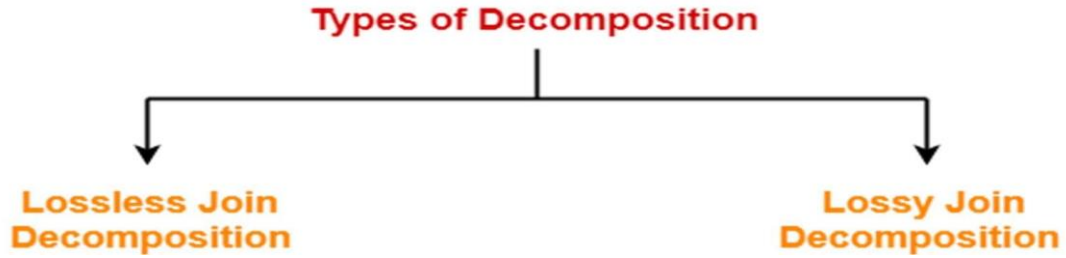
Encryption is the process of converting simple data into a coded one. In technical terms, encryption is a method of converting plaintext into ciphertext so that only authorized users can understand it. The process of encryption requires a key and both sender and receiver have this key, thus the data traveling from one end to another can be secure

Decryption is a process where the coded data is converted back to the readable form.
i.e., the ciphertext is converted into plain text. The process of decryption also requires a key so that when the data is received at the receiver end, it can be converted into readable form.

Decomposition

Decomposition in DBMS is a process of dividing a table into multiple tables, aiming to remove redundancy, inconsistencies, and anomalies.

Types of Decomposition



-

Lossless Join Decomposition-

Consider there is a relation R which is decomposed into sub relations R1 , R2 , …. , R_n. This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.

For lossless join decomposition, we always have-

$$R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n = R$$

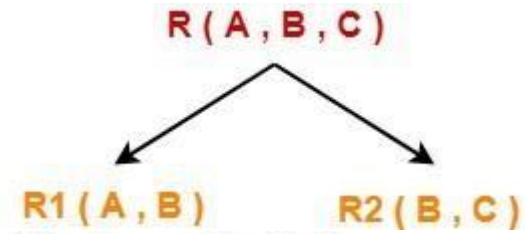
where ⋈ is a natural join operator

Example-

Consider the following relation R (A , B , C)-

A	B	C
1	2	1
2	5	3
3	3	3

Consider this relation is decomposed into two sub relations R1(A , B) and R2(B , C)-



The two sub relations are-

A	B
1	2
2	5
3	3

R_1(A , B)

B	C
2	1
5	3
3	3

R_2(B , C)

Now, let us check whether this decomposition is lossless or not. For lossless decomposition, we must have- **R1 ⋈ R2 = R**

Now, if we perform the natural join (⋈) of the sub relations R1 and R2 , we get

A	B	C
1	2	1
2	5	3
3	3	3

This relation is same as the original relation R. Thus, we conclude that the above decomposition is lossless join decomposition.

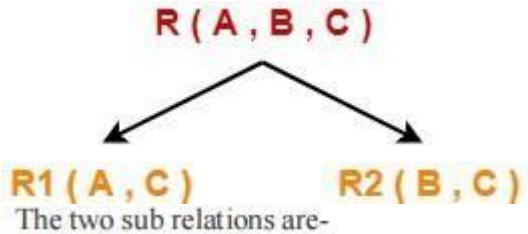
Lossy Join Decomposition-

Consider there is a relation R which is decomposed into sub relations R1 , R2 ,… , Rn. This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed.

Example-

Consider the above relation R(A , B , C)-

Consider this relation is decomposed into two sub relations as R1(A , C) and R2(B , C)-



The two sub relations are-

A	C
1	1
2	3
3	3

R_1(A , B)

B	C
2	1
5	3
3	3

R_2(B , C)

Now, let us check whether this decomposition is lossy or not.

For lossy decomposition, we must have **R1 ⋈ R2 ⊃ R**

Now, if we perform the natural join (⋈) of the sub relations R1 and R2 we get

A	B	C
1	2	1
2	5	3
2	3	3
3	5	3
3	3	3

This relation is not same as the original relation R and contains some extraneous tuples. Thus, we conclude that the above decomposition is lossy join decomposition.

Previous Question Exercise:

Q1. Consider the following database schema and answer the questions.

Employee(E_id,C_id,E_Name,Email,Phone_no,Salary,Address,F_id)

Customer(C_ID, C_Name, Email, Phone_no,F_id)

Food Item(F_ID, C_ID,F_Name, Price, Quantity)

1) Write the sql statement required to create these relations, you have to include allintegrity constraints.
2) Define constraints on food item table as such quantity should be at least one item. 3) Define constraints on Employee such that a minimum 15000 BDT pay scale is followed by the organization.

Answer:

1)

```
CREATE TABLE Employee (
    E_id INT PRIMARY KEY,
    C_id INT,
    E_Name VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone_no VARCHAR(20) UNIQUE,
    Salary DECIMAL(10, 2) CHECK (Salary >= 15000),
    Address VARCHAR(255),
    F_id INT,
    FOREIGN KEY (C_id) REFERENCES Customer(C_ID),
   FOREIGN KEY (F_id) REFERENCES FoodItem(F_ID) );

CREATE TABLE Customer (
    C_ID INT PRIMARY KEY,
    C_Name VARCHAR(50) NOT NULL,
```

```
    Email VARCHAR(100) UNIQUE,
    Phone_no VARCHAR(20) UNIQUE,
    F_id INT,
    FOREIGN KEY (F_id) REFERENCES FoodItem(F_ID) );

CREATE TABLE FoodItem (
    F_ID INT PRIMARY KEY,
    C_ID INT,
    F_Name VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    Quantity INT CHECK (Quantity >= 1),
    FOREIGN KEY (C_ID) REFERENCES Customer(C_ID) );
```

2)

```
ALTER TABLE FoodItem
ADD CONSTRAINT CHK_Quantity CHECK (Quantity >= 1);
```

3)

```
ALTER TABLE Employee
ADD CONSTRAINT CHK_Salary CHECK (Salary >= 15000);
```

Q2. The test that must be made to preserve referential integrity for delete/update operation

To preserve referential integrity for delete and update operations in a relational database, you typically perform tests or checks to ensure that the operations do not violate any foreign key constraints. Referential integrity ensures that relationships between tables remain consistent and valid.

Here are the tests commonly performed:

Delete Operation Test:

Cascade Delete: If a parent row is deleted, the database should automatically delete any related child rows. To test this:

- ✓ Delete a parent row.
- ✓ Verify that all related child rows are also deleted automatically.

Restrict Delete: Prevents the deletion of a parent row if there are related child rows.

To test this:

- ✓ Attempt to delete a parent row that has related child rows. Verify that the deletion is blocked or an error is raised.

Set Null/Delete: Sets the foreign key column in related child rows to NULL or a default value when the parent row is deleted. To test this:

- ✓ Delete a parent row.
- ✓ Verify that foreign key columns in related child rows are set to NULL or a default value.

Update Operation Test:

Cascade Update: If a primary key in the parent table is updated, the corresponding foreign key values in related child rows should also be updated. To test this:

- ✓ Update a primary key in the parent table.
- ✓ Verify that corresponding foreign key values in related child rows are also updated.

Restrict Update: Prevents the update of a primary key in the parent table if there are related child rows. To test this:

- ✓ Attempt to update a primary key in the parent table that has related child rows. □ Verify that the update is blocked or an error is raised.

Set Null/Update: Sets the foreign key column in related child rows to NULL or a default value when the primary key in the parent table is updated. To test this:

- ✓ Update a primary key in the parent table.
- ✓ Verify that foreign key columns in related child rows are set to NULL or a default value.

These tests ensure that referential integrity constraints are enforced properly and that the database maintains consistency when performing delete and update operations across related tables.