Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

CREATING CREATORS - SLTIET

Relational Database Management Systems

RDBMS - Subject Code: 4330702

Prof. Rinkal Umaraniya

CSE Department

SLTIET, Rajkot

CREATING CREATORS - SLTIET

Unit - 3

Database Integrity Constraints & Objects

CREATING CREATORS - SLTIET

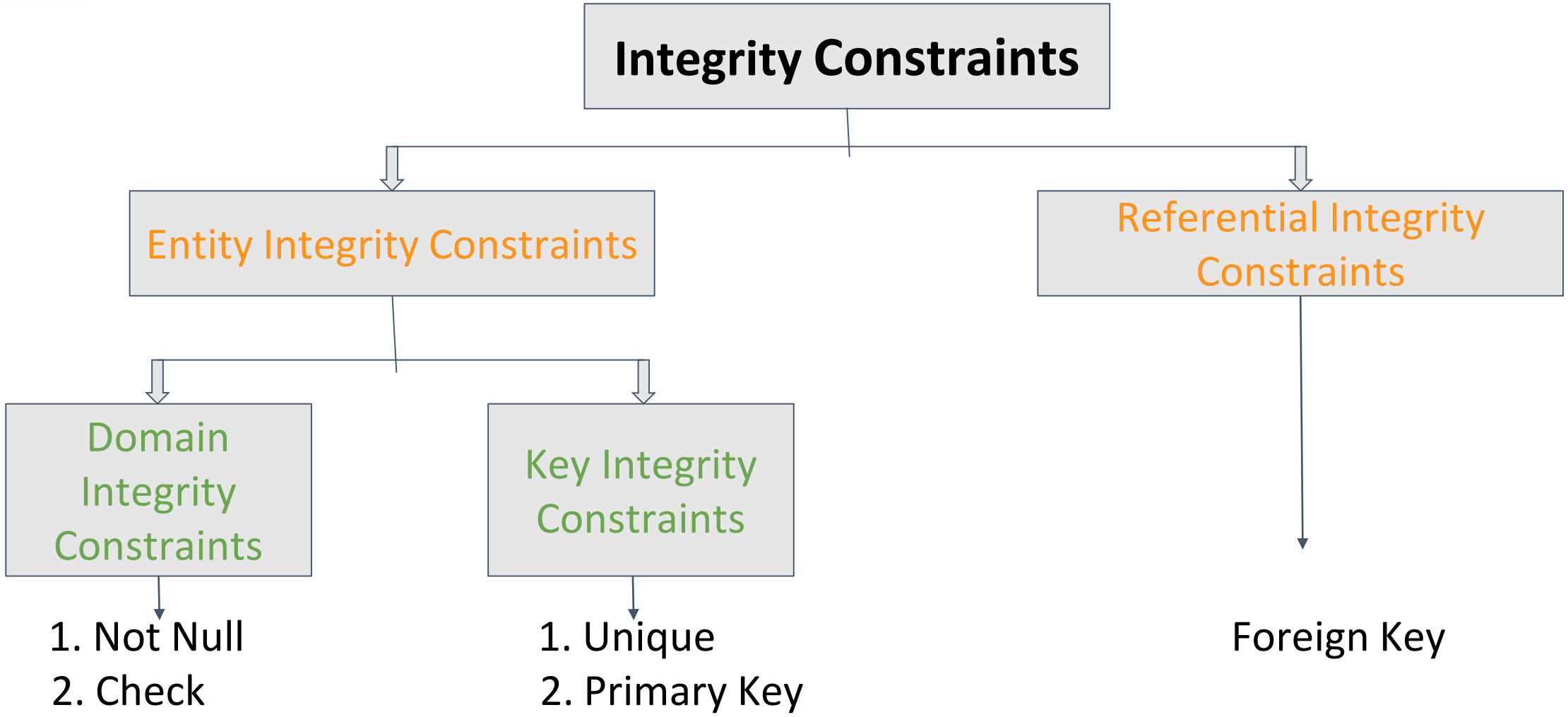
Sub Topics:

3.1 Entity Integrity Constraints
3.2 Referential Integrity Constraints
3.3 Views
3.4 Synonym
3.5 Sequences
3.6 Index

What is Integrity Constraints?

- Integrity constraints in DBMS are used to ensure that data is consistent and accurate.
- Data integrity in the database is the correctness, consistency and completeness of data.
- In Database Management Systems, integrity constraints are pre-defined set of rules that are applied on the table fields(columns) or relations to ensure that the overall validity, integrity, and consistency of the data present in the database.

Integrity Constraints



Example : Table Name - Banking

We take one example to understand all the Integrity Constraints:

```
-> CREATE TABLE Banking( acc_no      int             NOT NULL    PRIMARY KEY,
                        name         varchar[30]     NOT NULL,
                        balance      int             NOT NULL    CHECK(balance>0),
                        contact_no   int             NOT NULL    UNIQUE,
                        city         varchar[25] );

-> INSERT INTO Banking ( acc_no, name, balance, contact_no, city)
                VALUES ( A111, 'Suhana', 55000, 6357895421, 'Rajkot'),
                       ( A112, 'Rockey', 150000, 8563214698, 'Pune'),
                       ( A113, 'Mira', 60000, 9658741235, 'Jamnagar');
```

Example : Table Name - Banking

We take one example to understand all the Integrity Constraints:

acc_no	name	balance	contact_no	city
A111	Suhana	55000	6357895421	Rajkot
A112	Rocky	150000	8563214698	Pune
A113	Mira	60000	9658741235	Jamnagar

3.1 Entity Integrity Constraints

"Entity integrity constraints restrict the values in a row of an individual table."

1) Domain Integrity Constraints

- Specifies that the value of each column must belong to the domain of that column.

- For example, a balance for any account in the banking system should not be negative value.

- Domain Integrity further divided into two parts:

  A) Not Null

  B) Check

Continue…

A) Not Null:

There may be Records in the table that do not contain any value for some fields. In other words, a NULL value represents an empty field.

- A NULL value indicates ‘not applicable’, ‘missing’, or ‘not known’.
- A NULL value distinct from zero or other numeric value for numerical data.
- A NULL value is also distinct from blank space for character data.
- A NULL value will evaluate to null in any expression.
- The result of any condition including null value is unknown, and treated as a FALSE.

Continue…

Syntax : ColumnName datatype (size) NOT NULL

Example: INSERT INTO Banking (acc_no, name, balance, contact_no, city)

VALUES (A114, ‘Vamika’, NULL, 5698756321, ‘Mumbai’);

Output : An error occurred because in “balance column NULL value is not allowed”.

Continue…

B) Check:

- ➔ The CHECK constraints is used to implement business rules.
- ➔ Business rules may vary from system to system.
- ➔ Example of business rules are -

  A balance in any account should not be a negative value in any situation.

  An account number must start with ‘A’.

- ➔ This rules are specific to a particular bank system.

Syntax: ColumnName datatype (size) CHECK (CONDITION)

Continue…

Example: INSERT INTO Banking (acc_no, name, balance, contact_no, city)

VALUES (A114, ‘Ruhi’, 0, 8654231786, ‘Rajkot’);

Output : Here, error occurred when we insert data in table. Because when create table, put a condition to CHECK balance should be greater than 0.

Continue…

2) Key Integrity Constraints

Ensure that there must be some way to uniquely distinguish two different rows of the same table.

Key integrity constraints divided into two parts :

A) Primary Key

- ➔ Primary key is a set of one or more columns used to identify each record uniquely in a column.
- ➔ It can not accept null, duplicate values.
- ➔ A single column primary key is called a simple key, while a multi-column primary key is called a composite key.

Continue…

Example: INSERT INTO Banking (acc_no, name, balance, contact_no, city)

VALUES (A113, ‘Mansi’, 85000, 7563241586, ‘Rajkot’);

Output : Here, error occurred because we insert account number A113 that is already exists in table (duplicate value is not allowed in acc_no column because it is a Primary Key).

Example: INSERT INTO Banking (acc_no, name, balance, contact_no, city)

VALUES (‘Mansi’, 85000, 7563241586, ‘Rajkot’);

Output : Here, error occurred because we insert nothing in account number.(null value is not allowed in acc_no column because it is a Primary Key).

Continue…

B) Unique Key

A column must have unique values. This is required to identify all records stored in the table uniquely. It is like a Primary key but it can accept only one null value and it can not have duplicate values.

Syntax: columnName datatype (size) UNIQUE

Example: INSERT INTO Banking (acc_no, name, balance, contact_no, city)

VALUES (A114, 'Mansi', 85000, 'Rajkot');

Continue…

Output :

acc_no	name	balance	contact_no	city
A111	Suhana	55000	6357895421	Rajkot
A112	Rocky	150000	8563214698	Pune
A113	Mira	60000	9658741235	Jamnagar
A114	Mansi	85000	NULL	Rajkot

3.2 Referential Integrity Constraints

Referential Integrity Constraint ensures that there must always exist a valid relationship between two relational database tables. This valid relationship between the two tables confirms that a foreign key exists in a table. It should always reference a corresponding value or attribute in the other table.

Foreign Key

"A Foreign key is a set of one or more columns whose values are derived from the primary key or unique key of another table."

- The table, in which a foreign key is defined, is called a foreign table, detail table or child table.
- The table in which primary key or unique key is referred, is called a primary table, master table or parent table.

Continue…

Here, we take example to use of FOREIGN KEY.

Account

a_no	balance	b_name
A01	5000	vvn
A02	6000	ksad
A03	7000	anand
A04	8000	ksad
A05	6000	vvn

Branch

b_name	b_address
vvn	Mota bazar, VV Nagar
ksad	Chhota bazar, Karmsad
anand	Nana bazar, Anand

Here, b_name is common between both tables and provide a link between Account and Branch. It represent which account is related to which branch.

A FOREIGN KEY constraints is used to maintain data consistency between two tables.

Continue…

Example :

Create Account table containing column b_name as a FOREIGN KEY referring to the Branch table.

Input :

```
CREATE TABLE Account( a_no       int              PRIMARY KEY,
                      balance    int ,
                      b_name     varchar[30],
                      FOREIGN KEY (b_name)  REFERENCES  Branch (b_name)
                    );
```

3.3 Views

- A view is a virtual or logical table that allows viewing or manipulating the parts of the tables.
- A view is derived from one or more tables known as base tables.
- A view looks like and works similarly to normal tables. But a view does not have storage space to store data.
- A view is created by a query, i.e. a SELECT statement which uses base tables.
- A view is dynamic and always reflects the current data of the base tables.
- Only the definition of view is stored in the database.

Continue…

- When a view is referenced in a SQL statement following steps will be followed:
  - o Its definition is retrieved from the database.
  - o The base tables are opened.
  - o A query, specified in definition is executed.
  - o A view is created on top of the base tables.
- When any operation is performed on view, it is actually performed on the base table.
- For example, any SELECT operation on view displays data from the base table. In a similar way, INSERT, UPDATE, DELETE operations modify the contents of the base table.

Continue…

Types of Views:

1) Read-only View:

- Allows only SELECT operation, this means user can only view data.
- No INSERT, UPDATE or DELETE operations are allowed. This means the contents of the base table cannot be modified.

2) Updateable View:

- Allows SELECT as well as INSERT, UPDATE and DELETE operations. This means contents of the base tables can be displayed as well as modified.

Example

Table Name : Employee

Emp_ID	Name	Salary	Department	Town
101	Herry	20000	Admin	Rajkot
102	Tom	40000	Account	Ahmedabad
103	Mr. Baxi	35000	H & R	Jamnagar
104	Jack	50000	Technical	Rajkot
105	Gracy	25000	Packaging	Jamnagar

i) Create View

A view is created with the CREATE VIEW statement.

Syntax: CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;

Example: => CREATE VIEW [Rajkot Employee] AS
SELECT Name, Department
FROM Employees
WHERE Town= 'Rajkot';

=> Select * from [Rajkot Employee];

Output :

Name	Department
Herry	Admin
Jack	Technical

ii) Alter View

A view can be updated with the ALTER VIEW statement.

Syntax: ALTER VIEW view_name AS
SELECT column1, column2, …
FROM table_name
WHERE condition;

Example: => ALTER VIEW [Rajkot Employee] AS
SELECT Name, Department, Salary
FROM Employees
WHERE Town= 'Rajkot';

=> Select * from [Rajkot Employee];

Output :

Name	Department	Salary
Herry	Admin	20000
Jack	Technical	50000

iii) Drop View

A view can be deleted with the DROP VIEW statement.

Syntax: DROP VIEW view_name;

Example: **=>** DROP VIEW [Rajkot Employee];

=> Select * from [Rajkot Employee];

Output : No any table saw because we deleted the view.

3.4 Synonym

- A synonym is an alternative name for database object such as tables, indexes, sequences.
- A synonym can be used to hide the actual identity of the object being referenced.
- For example, if there is a need to hide name of some particular table, then a synonym can be created to refer that table, hiding the original name.
- Another use of the synonym is to abbreviate (Shorten) the table names, particularly tables from other users.
- For example, user1 can create synonym for Customer table owned by user2.Appropriate privileges must be granted to a user before the user can use the synonym.

i) Create Synonym

Syntax: **CREATE SYNONYM** [name_of_schema.] name_of_synonym

FOR name_of_base_object;

Example :

1) First, we create a database. The name of the database is **'salesdb'.**

=> **Create database** saledb;

2) Now, we will create a schema and a table inside that schema. The syntax for the creation of a schema and a table is:

CREATE SCHEMA grocery;

Continue…

```
CREATE TABLE grocery.fruits( fruit_name VARCHAR(100) NOT NULL,
                             quantity  INT PRIMARY KEY );
```

3) Now let us add some data to the fruits table. The syntax for the same is:

```
INSERT INTO grocery.fruits(fruit_name,quantity) VALUES('Apple',12);
INSERT INTO grocery.fruits(fruit_name,quantity) VALUES('Coconut',2);
INSERT INTO grocery.fruits(fruit_name,quantity) VALUES('Blueberry',30);
```

Continue…

=> **Select * from saledb.grocery.fruits;**

Output :

fruit_name	quantity
Apple	12
Coconut	2
Blueberry	30

4) Whenever we want to refer to the fruits table, first we need to write the database name followed by the schema name and then the name of the table. To reduce this overhead, we can create a synonym.

Continue…

CREATE SYNONYM fruits **FOR** salesdb.grocery.fruits;

5) Select * from fruits;

fruit_name	quantity
Apple	12
Coconut	2
Blueberry	30

ii) Drop Synonym

The DROP SYNONYM statement deletes one or more synonyms from a database. A synonym is an alias (alternate name) for a table, view, or index.

Syntax: **DROP SYNONYM** synonym_name;

Example : **DROP SYNONYM** fruits;

3.5 Sequences

- To distinguish different records of a table from each other, it is required that each record must have distinct values.
- The primary key constraint ensures this by not allowing duplicate or NULL values in columns defined as a primary key.
- Such columns generally contain sequential numbers such as 1, 2, 3,... or combination of sequential values with some strings, such as ‘A01’, ‘A02’,....
- While entering data manually in insert or update operations, it is difficult to track such a type of sequence.
- “A Sequence helps to ease the process of creating unique identifiers for a record in a database.”
- A Sequence is simply an automatic counter, which generates sequential numbers whenever required.

Continue…

- A value generated can have a maximum 38 digits.
- A sequence can be defined for following purpose:
  -> To generate numbers in ascending or descending order.
  -> Provide intervals between numbers.
  -> Caching of sequence numbers in memory to speed up their availability.

Note :

➔ A default sequence, i.e. created without any option, always starts with 1.
➔ It is in ascending order.
➔ And value are incremented by 1.

i) Create Sequence

Syntax:

CREATE SEQUENCE sequence_name

START WITH initial_value

INCREMENT BY increment_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

Continue…

Where,

-> **sequence_name:** Name of the sequence.

-> **initial_value:** Starting value from where the sequence starts.
Initial_value should be greater than or equal to minimum value and less then equal to maximum value.

-> **increment_value:** Value by which sequence will increment itself.
Increment_value can be positive or negative.

-> **minimum_value:** Minimum value of the sequence.

-> **maximum_value:** Maximum value of the sequence.

-> **cycle:** When a sequence reaches its set_limit it starts from the beginning.

-> **nocycle:** An exception will be thrown if the sequence exceeds its max_value.

Continue…

Example:

For Ascending Order:

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

For Descending Order:

```
CREATE SEQUENCE sequence_2
start with 100
increment by -1
minvalue 1
maxvalue 100
cycle;
```

Continue…

Example to use **Sequence**

1) Create a table named students with columns as id and name.

Input : CREATE TABLE students (ID number(10),
NAME char(20));

2) Now insert values into a table

Input : INSERT into students VALUES
(sequence_1.nextval,'Gaurav')
(sequence_1.nextval,'Aman');

Output :

ID	NAME
1	Gaurav
2	Aman

ii) Alter Sequence

Syntax:

ALTER SEQUENCE sequence_name

RESTART WITH initial_value

INCREMENT BY increment_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

Example :

ALTER SEQUENCE sequence_1

RESTART WITH 5;

Output :

ID	NAME
5	Gaurav
6	Aman

iii) Drop Sequence

Syntax:

DROP SEQUENCE sequence_name;

Example :

DROP SEQUENCE sequence_1;

3.6 Index

- SQL INDEX is used to quickly find data without searching every rows in a database table.
- SQL INDEX improves the speed of search operation on a database table. But additional you need more storage space to maintain duplicate copy of the database.
- End users does not know for indexes is created on table, only they are searching data more quickly and efficiently.
- Search is always efficient when data to be searched is sorted in some specific order such as in ascending order.
- An Index is an ordered list of contents of the column (or a group of columns) of a table.
- An index is similar to a table. It contains at-least two columns:
  1) A column having sorted data on which an index is created.
  2) A column representing RowID for each row in a table.
- A RowID is a unique identifier for each record inserted in a table.

i) Create Index

Type of SQL INDEX:

- **Simple INDEX :**

  Create INDEX on one column.

- **Composite INDEX :**

  Create INDEX on multiple columns.

- **Unique INDEX :**

  Create INDEX on column to restrict duplicate values on INDEX column.

- **Duplicate INDEX :**

  Create INDEX on column to allow duplicate values on INDEX column.

1) Simple Index

Simple INDEX create only one selected column of the database table.

Syntax :

```
CREATE INDEX index_name
ON table_name (column_name)
```

Example :

```
CREATE INDEX index_user_name
ON userinfo (name);
```

We are creating a simple index on the name column of the userinfo table. In this column allow duplicate values of the column.

2) Composite Index

Composite INDEX create on multiple selected column of the database table.

Syntax :

```
CREATE INDEX index_name
ON table_name (column_name, column_name)
```

Example :

```
CREATE INDEX index_user_name
ON userinfo (no, name);
```

We are creating a composite index on the no and name column of the userinfo table. In this column allow duplicate values of the column.

3) Unique Index

Unique INDEX create on selected column of the database table and does not allow duplicate values of that indexes column.

Syntax :

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

Example :

```
CREATE UNIQUE INDEX index_user_name
ON userinfo (name);
```

We are creating a unique index on the name column of the userinfo table. Duplicate name value are does not allow again for creating indexes.

4) Duplicate Index

Duplicate INDEX create by default on column of the database table when we not specify UNIQUE keyword and allow duplicate values of that indexes column.

Syntax :

```
CREATE INDEX index_name
ON table_name (column_name)
```

Example :

```
CREATE INDEX index_user_name
ON userinfo (name);
```

We are creating a duplicate index on the name column of the userinfo table. Duplicate name value are allow for creating indexes.

ii) Alter Index

Syntax : ALTER INDEX index_name

RENAME TO new_index_name;

Example : ALTER INDEX index_user_name

RENAME TO u_name;

iii) Drop Index

Syntax : DROP INDEX index_name ;

Example : DROP INDEX index_user_name;

RowID

- A RowID is a unique identifier for each record inserted in a table.
- A RowID is a hexadecimal string and contains the logical address of the location in a database where a particular record is stored.
- RowID column can be used like any other column in the SELECT statement and with WHERE clause.

Questions :

Q-1) What is Integrity Constraints?
Q-2) Explain Entity and Referential Integrity Constraints with example.
Q-3) Explain View with Create, Alter and Drop command.
Q-4) How Create, Alter and Drop Sequence with example.
Q-5) Write about the Synonym.
Q-6) Explain in detail about Index.

SLTIET
आत्मदीपोभव

Thank You

CREATING CREATORS - SLTIET