Birla Institute of Technology and Science, Pilani

CS F212 Database Systems

Supplementary for Lab #1

CHECK

- We can specify a much more general type of constraint using the CHECK constraint. A CHECK constraint specifies a boolean value expression to be evaluated for each row before allowing any data change. Any INSERT, UPDATE, or DELETE that would cause the condition for any row to evaluate to false is rejected by the DBMS.

```
CHECK (<condition>)
```

- A CHECK constraint may be specified as either a column or table constraint. In the following example, we specify CHECK constraints on the students table:

```
CREATE TABLE STUDENT1(
IDNO CHAR(13) PRIMARY KEY,
NAME VARCHAR(20) NOT NULL,
CGPA NUMERIC(4,2) CHECK(CGPA >= 2 AND CGPA <= 10),  -- CGPA CONSTRAINT
ROOMNO NUMERIC(3) CHECK(ROOMNO >99),
HOSTEL_CODE VARCHAR(2) CHECK(HOSTEL_CODE IN ('VK','RP','MB')))
);
```

- Check constraints can also be named.
- Does a roomno with a NULL value violate the CHECK constraint? No. In this case, the CHECK condition evaluates to unknown. The CHECK constraint only rejects a change when the condition evaluates to false. In the SQL standard, a CHECK constraint condition may even include subqueries referencing other tables; however, many DBMSs do not implement this feature.

Rename table:

```
EXEC SP_RENAME 'PHONE','PHONE1';
```

Alter Table

You can modify the columns and constraints of a table using the following:

```
ALTER TABLE <TABLE NAME> ACTION>;
```

There are several types of actions:

1. ADD [COLUMN] <column definition>

Adds a new column to <table name>. The <column definition> may contain any of the elements of a column definition in CREATE TABLE, such as types and constraints.

Rules for adding a Column

1. You may add a column at any time if NOT NULL isn't specified.
2. You may add a NOT NULL column in three steps 2.1. Add the column without NOT NULL specified.
   2.2. Fill every row in that column with data.
   2.3. Modify the column to be NOT NULL.

```
SQL> ALTER TABLE STAFF ADD EMAIL VARCHAR (20);
SQL> ALTER TABLE STUDENTS ADD HOSTEL VARCHAR (20);
```

2. **ALTER COLUMN <column name> <datatype> [<constraints>]**

Rules for modifying a Column

1. You can increase a character column's width at any time.
2. You can increase the number of digits in a NUMBER Column at any time.
3. You can increase or decrease the number of decimal places in a NUMBER Column at any time.

In addition, if a column is NULL for every row of the table, you can make any of these changes.

1. You can change its datatype.
2. You can decrease a character Column's width.
3. You can decrease the number of digits in a NUMBER column.

```
SQL> ALTER TABLE STAFF ALTER COLUMN EMAIL VARCHAR (50);
```

3. DROP <column name> [CASCADE | RESTRICT]

Delete <column name> from <table name>. If there are any external dependencies on this column, the drop will be rejected. If CASCADE is specified, the DBMS will attempt to eliminate all external dependencies before deleting the column. If RESTRICT is specified, the DBMS will not allow the DROP if any external dependencies would be violated. This is the default behaviour.

```
SQL> ALTER TABLE STAFF DROP COLUMN EMAIL;
```

4. ADD <table constraint>

Add a new constraint to <table name>. <table constraint> uses the same syntax as table constraints in CREATE TABLE.

```
SQL> ALTER TABLE STUDENTS ADD CONSTRAINT AGECHECK CHECK (AGE>15);
```

5. DROP CONSTRAINT <constraint name> [CASCADE | RESTRICT]

Delete <constraint name> from <table name>. If there are any external entities that depend on this constraint, the drop could invalidate the entity. If RESTRICT is specified and an entity could be invalidated, the DBMS will not drop the constraint. This is the default behavior. If CASCADE is specified, the DBMS will attempt to eliminate all potentially violated entities before deleting the constraint.

```
SQL> ALTER TABLE COURSE DROP CONSTRAINT UN_COURSE;
```

Case study:

The Employees Database stores information about a business, including employees, departments, and projects. Each employee works for one department and is assigned many projects. Each department has many employees, has many projects, and may be a subdepartment of one department (e.g., Accounting is a subdepartment of Administration). Each project is assigned to one department and is worked on by many employees.

Employees

Employeeid	Numeric(9)	Unique employee identifier (Primary Key)
Firstname	Varchar(10)	Employee first name
Lastname	Varchar(20)	Employee last name
Deptcode	Char(5)	Identifier of department the employee works for. Foreign key referencing departments.code
Salary	Numeric(9,2)	Employee salary

Departments

Code	Char(5)	Unique Department Identifier
Name	Varchar(30)	Department name
Managerid	Numeric(9)	Identifier of employee who manages the department. Foreign key referencing employees.employeeid
Subdeptof	Char(5)	Code of department that includes this department as one of its immediate subdepartments. Foreign key referencing departments.code

Projects

Projectid	Char(8)	Unique project identifier
Deptcode	Char(5)	Identifier of department managing this project. Foreign key referencing departments.code
Description	Varchar(200)	Project description
Startdate	Date	Project start date
Stopdate	Date	Project stop date. NULL value indicates that the project is ongoing
Revenue	Numeric(12,2)	Total project revenue

Workson

Employeeid	Numeric(9)	Identifier of employee working on a project. Foreign key referencing employees.employeeid
Projectid	Char(8)	Identifier of project that employee is working on. Foreign key referencing projects.projectid
Assignedtime	Numeric(3,2)	Percentage of time employee is assigned to project

- Let us create the database for the above schema

```
CREATE TABLE DEPARTMENTS(
```

```
CODE CHAR(5) PRIMARY KEY,
NAME VARCHAR(30),
MANAGERID NUMERIC (9),
SUBDEPTOF CHAR (5)
);
```

- Create employees table and then run the following commands

```
ALTER TABLE DEPARTMENTS ADD CONSTRAINT FK_DEPARTMENTS_EMPLOYEES
FOREIGN KEY (MANAGERID) REFERENCES EMPLOYEES;
ALTER TABLE DEPARTMENTS ADD CONSTRAINT FK_DEPARTMENTS_SUBDEPT
FOREIGN KEY (SUBDEPTOF) REFERENCES DEPARTMENTS;
```

Exercises

1. Create project table with primary key
2. Create workson table
3. Create foreign keys between employees and department
4. Create foreign keys between workson and employees
5. Create foreign keys between projects and departments

---------&----------