Constraints

SCS2209- Database II

Ms. Hiruni Kegalle

Data Integrity

- Ensures that the values entering to the database is accurate and valid.
- Uses integrity constraints
- Integrity constraints maintains accurate databases by eliminating invalid data updates/ insert/ deletes.

eid	Ename	Designation	Salary	did	dname	location
1000	Ajith	Lecturer	60000	1	Academic	CMB
1001	Sunil	Executive	45000	3	Maintenance	SJP
1002	Kamal	Lecturer	75000	1	Academic	CMB
1003	Piyumi	Manager	50000	2	Admin	RHN
1004	Roshan	Lecturer	35000	1	Academic	CMB
1005	Nuwan	Lecturer	80000	1	Academic	CMB
1006	Jayamini	Assistant	25000	2	Admin	RHN
1007	Nishani	Lecturer	42000	1	Academic	CMB
1008	Amal	Assistant	28000	4	NOC	CMB

eid	Ename	Designation	Salary	did	dname	location
1000	Ajith	Lecturer	60000	1	Academic	CMB
1001	Sunil	Executive	45000	3	Maintenance	SJP
1002	Kamal	Lecturer	75000	1	Academic	CMB
1003	Piyumi	Manager	50000	2	Admin	RHN
1004	Roshan	Lecturer	35000	1	Academic	CMB
1005	Nuwan	Lecturer	80000	1	Academic	CMB
1006	Jayamini	Assistant	25000	2	Admin	RHN
1007	Nishani	Lecturer	42000	1	Academic	CMB
1008	Amal	Assistant	28000	4	NOC	CMB
1009	**Saman**	**Lecturer**	**35000**	**16**	**Academic**	**CMB**

eid	Ename	Designation	Salary	did	dname	location
1000	Ajith	Lecturer	60000	1	Academic	CMB
1001	Sunil	Executive	45000	3	Maintenance	SJP
1002	Kamal	Lecturer	75000	1	Academic	CMB
1003	Piyumi	Manager	50000	2	Admin	RHN
1004	Roshan	Lecturer	35000	1	Academic	CMB
1005	Nuwan	Lecturer	80000	1	Academic	CMB
1006	Jayamini	Assistant	25000	2	Admin	RHN
1007	Nishani	Lecturer	42000	1	Academic	CMB
1008	Amal	Assistant	28000	4	NOC	CMB

Database Integrity Constraints

- Constraints are conditions that specify restrictions on the database state.
- Types in relational DB
  - **Entity integrity** does not allow two rows with the same identity in a table
  - **Domain integrity** allows only predefined values
  - **Referential integrity** allows only the consistency of values across related tables
  - **User-defined integrity** define constraints

Database Constraints

- They are the restrictions on the contents of the database and its operations
- Types of Constraints
  - Primary key constraint
  - Foreign key constraint (referential integrity)
  - Unique constraint
  - NOT NULL constraint
  - Check Constraint
  - Default constraint

Primary Key Constraints

- ***Primary key*** uniquely identifies each record in a table.
- It must have unique values and cannot contain nulls.
  - This is because primary key values are used to identify the individual tuples
  - If PK has several attributes, null is not allowed in any of these attributes
- Table can have only one primary key.
- In the below example the **studentId** field is marked as primary key, that means the **studentId** field cannot have duplicate and null values

Primary Key Constraint

```
CREATE TABLE Student (
studentId   CHAR (10),
name        CHAR(20),
address     CHAR(25),
age         INT,
CONSTRAINT pk_stdID PRIMARY KEY (studentId));
```

Unique Constraint

- UNIQUE Constraint enforces a column or set of columns to have unique values.
- If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.

Unique Constraint

```
CREATE TABLE Student (
Stdid       CHAR (10)PRIMARY KEY,
Name        CHAR(20),
Address     CHAR(25),
Age         INT,
NIC         CHAR(10)UNIQUE
);
```

Not Null Constraint

- NOT NULL constraint makes sure that a column does not hold NULL value.
- When we don’t provide value for a particular column while inserting a record into a table, it takes NULL value by default.
- By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

Not Null Constraint

```
CREATE TABLE Student (
Sid     INT Primary Key,
name    CHAR(20) NOT NULL,
address CHAR(25),
age     INT
);
```

Default Constraint

- The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

Default Constraint

```
CREATE TABLE Student (
name        CHAR(20),
Address     CHAR(25),
department CHAR (20) DEFAULT “Computer Science”,
Age         INT
);
```

Check Constraint

- This constraint is used for specifying range of values for a particular column of a table.
- When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

Domain Constraint

- Each table has certain set of columns and each column allows a same type of data, based on its data type.
- The column does not accept values of any other data type.
- Domain constraints are user defined data type and we can define them like this:
- Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)

Check Constraint

```
CREATE TABLE UnderGrad_Student (
sid          CHAR(25) Primary Key,
name         CHAR(20),
address      CHAR(25),
Age          INT,
Reg_Course   CHAR(10)CHECK (Age BETWEEN 19 AND 26)
);
```

Activity

Create a table called “Department” with the following constraints.

- dept_ID is a number used as the primary key
- dept_name cannot be null
- I want default location to be ‘Colombo’
- dept_head specifies a unique number
- number_employees should be an integer between 1-25

Activity - Answer

- dept_ID is a number used as the primary key
- dept_name cannot be null
- I want default location to be 'Colombo'
- dept_head specifies a unique number
- number_employees should be an integer between 1-25

```
CREATE TABLE Department (
dept_ID      INT Primary Key,
dept_name    CHAR(20) NOT NULL,
location     CHAR(25) DEFAULT "Colombo",
dept_head    INT UNIQUE,
number_employees     INT CHECK (number_employees BETWEEN 1 AND 25)
);
```

Foreign Key Constraint (referential integrity)

- A FOREIGN KEY is a key used to link two tables together.
- Foreign keys are the columns of a table that points to the primary key (unique) of another table.
- They act as a cross-reference between tables.
- The table containing the foreign key is called the *child table / referencing table*, and the table containing the candidate key is called the *referenced or parent table.*

Foreign Key Constraint (referential integrity)

- The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column
- It has to be one of the values contained in the table it points to.

PersonID	LastName	FirstName	Age
1	Perera	Saman	35
2	Karuna	Ramesh	19
3	Kate	Rumai	24

OrderID	Location	PersonID
098	Colombo	1
721	Kandy	3
87	Galle	3

Foreign key Constraint

```
CREATE TABLE Orders (

OrderID INT NOT NULL,

location CHAR(25),

personID INT,

PRIMARY KEY (OrderID),

CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)

REFERENCES Persons(PersonID)

);
```

Referential Triggered Action

- Updates may propagate to cause other updates automatically.
- Operations
  - ON DELETE
  - ON UPDATE
- Actions To Take
  - RESTRICT: Reject the row to be deleted
  - SET NULL: Set value of foreign key to NULL
  - SET DEFAULT: Set value of foreign key to default value
  - CASCADE: Delete/ Update referencing row(s) as well
  - NO ACTION

Violations when INSERT / UPDATE

- Domain constraint Violation: if one of the attribute values provided for the new tuple is not of the specified attribute domain
- Key constraint Violation: if the value of a key attribute in the new tuple already exists in another tuple in the relation
- Entity integrity Violation: if the primary key value is null in the new tuple
- Referential integrity Violation: if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation

Example

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Super_Ssn	Dno
Kasun	Perera	234532	1999-12-13	Colombo	M	230000	343534	7
Shiva	Krishan	89892	2000-08-23	Kandy	M	78200	32149	5
Ameena	Safran	43422	2008-07-12	Gampaha	F	82300	89943	1
Stephani	Shaw	21898	2000-09-28	Galle	F	23000	78687	2

1. Insert <'Sama', 'Jayasena',Null,'1989-05-27','Matara','F',67000, Null,4> into EMPLOYEE
2. Insert <'Mary', 'Doe',234532,'2004-05-27','Badulla','F',98000, NULL,4> into EMPLOYEE
3. Insert <'Raj','Kumaran','345454','1989',NULL,'M','100k', Null,4> into EMPLOYEE

Example

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Super_Ssn	Dno
Kasun	Perera	234532	1999-12-13	Colombo	M	230000	343534	7
Shiva	Krishan	89892	2000-08-23	Kandy	M	78200	32149	5
Ameena	Safran	43422	2008-07-12	Gampaha	F	82300	89943	1
Stephani	Shaw	21898	2000-09-28	Galle	F	23000	78687	2

1. Insert <'Sama', 'Jayasena', **Null**,'1989-05-27','Matara','F',67000, Null,4> into EMPLOYEE : violate entity constraint

2. Insert <'Mary', 'Doe', **234532**,'2004-05-27','Badulla','F',98000, NULL,4> into EMPLOYEE: violate ~~entity~~ key constraint

3. Insert <'Raj','Kumaran', **'345454'**,**'1989'**,NULL,'M',**'100k'**, Null,4> into EMPLOYEE: violate domain constraint

Example

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Super_Ssn	Dno
Kasun	Perera	234532	1999-12-13	Colombo	M	230000	343534	7
Shiva	Krishan	898921	2000-08-23	Kandy	M	78200	32149	5
Ameena	Safran	434221	2008-07-12	Gampaha	F	82300	89943	1
Stephani	Shaw	218983	2000-09-28	Galle	F	23000	78687	2

4. Create table WORKS_ON(Essn INT, Hours FLOAT, PRIMARY KEY(Essn), FOREIGN KEY Essn REFERENCE Employee (Ssn));

Insert <999345,25.8> into WORKS_ON

Example

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Super_Ssn	Dno
Kasun	Perera	234532	1999-12-13	Colombo	M	230000	343534	7
Shiva	Krishan	898921	2000-08-23	Kandy	M	78200	32149	5
Ameena	Safran	434221	2008-07-12	Gampaha	F	82300	89943	1
Stephani	Shaw	218983	2000-09-28	Galle	F	23000	78687	2

4. Create table WORKS_ON(Essn INT, Hours FLOAT, PRIMARY KEY(Essn), FOREIGN KEY Essn REFERENCE Employee (Ssn));

Insert <**999345**,25.8> into WORKS_ON violate referential constraint

Violations when DELETE

- DELETE may violate only referential integrity:
- If the primary key value of the tuple being deleted is referenced by other tuples in the database
- Can be remedied by several actions: RESTRICT, CASCADE, SET NULL One of the above options must be specified for each foreign key constraint

Example

```
CREATE TABLE Employee (
NIC VARCHAR(10) PRIMARY KEY,
name VARCHAR(50) ,
works_in INT,
CONSTRAINT fk_EmpDept FOREIGN KEY
(works_in) REFERENCES Department(Dept_Nmbr)
ON DELETE CASCADE
ON UPDATE NO ACTION
)
```

When DROP TABLE

- The actions to take when Dropping tables
- RESTRICT – if there is constraint (FK / View) then do not drop the table
- CASCADE - drop all the other constraints & views that refers the table

```
DROP TABLE Employee [RESTRICT|CASCADE]
```

Add or Remove Constraints

- Drop a table's primary key constraint

```
Alter Table Student Drop Primary Key
```

- Drop a unique, foreign key, or check constraint

```
Alter Table Employee Drop Constraint fk_EmpDept
```

- Add a new constraint

```
Alter Table PassStudents Add Constraint avg_Marks Check( marks >= 50 )
```

Introduction to Triggers

- A trigger is an event that will run automatically when there is a change occur to the database
- It can be invoked either before or after the data is changed by
  - INSERT, UPDATE or DELETE

Reason to use Triggers

- Audit trails (Identify changes)
- Enforce integrity
- To apply business rules
  - E.g. A production organization maintains a warehouse. It maintains a minimum inventory level. When the inventory level goes down how will the reorder manager identify it at once?

Design Triggers

- Model of a trigger also known as event- condition - action model .
- When event occurs, check condition , if true , do action.
- To design a trigger you require to identify;
- When should the trigger be executed?
  - An event – a cause to check the trigger (update/ temporal)
  - A condition – logic to be satisfied to execute the trigger (optional)
  - The action - that should be taken when the trigger is executed

Types of Triggers

- Row Triggers - For each row (affected) of the table trigger is fired. If zero rows affected, no row level trigger will be executed.

Delete one employee from emp table whose department is HR and want the existing employee number to be updated.

- Statement Triggers - Only once the trigger will fire. Does not depends on how many rows are affected.

Update salary of every employee from department HR

Trigger Timing

- Before Trigger – runs before any change is made to the database.
  - If you want to withdraw money from ATM first need to check account balance before doing the transaction
- After Trigger - runs after changes are made to the database
  - After withdrawing money if the available balance is < 5000 reduce 25 from the account balance

Syntax for Specifying a Trigger

<trigger> : CREATE TRIGGER <trigger name>

(AFTER/BEFORE) <triggering events> ON <table name>

[FOR EACH ROW]

BEGIN

[WHEN <condition>] <trigger actions>;

END

<triggering events> : INSERT|UPDATE|DELETE [OF<column name>]

<trigger actions> : <PL/SQL block>

Example 01

Suppose *student_age* table is included in a School DB.

Create the table using following command.

```
Create table Student_age(age INT, Name Varchar(35))
```

Student_age

Age	Name

If we want to prevent inserting negative numbers for the *age* column, we can write a trigger.

Insert 0 automatically, for any number <0.

Example 01

```
DELIMITER //
Create Trigger before_insert_studentage BEFORE INSERT ON student_age
FOR EACH ROW
BEGIN
IF NEW.age < 0 THEN SET NEW.age = 0;
END IF;
END //
```

```
1.  INSERT INTO Student_age(age, Name) values(30, 'Rahul');
2.  INSERT INTO Student_age(age, Name) values(-10, 'Harshit');
```

Example 02

Suppose the following two tables are created in a company DB.

Employee

Name	Ssn	Salary	Dno	Supovisor_ssn

Department

Dname	Dno	Total_sal	Manager_ssn

- Dno can be NULL in Employee table
- Total_sal total salary of all the employees in the Dept.
- Maintaining Total_sal can be done using triggers.

Example 02 Cont.

Events:

1. Inserting (one or more) new employee tuples
2. Changing the salary of (one or more) existing employees
3. Changing the assignment of existing employees from one department to another
4. Deleting (one or more) employee tuples

Example 02 Cont.

Conditions:

1. Inserting (one or more) new employee tuples

    Dno != NULL

2. Changing the salary of (one or more) existing employees

    Dno != NULL

3. Changing the assignment of existing employees from one department to another

    No need to check ; action is always executed.

4. Deleting (one or more) employee tuples

    Dno != NULL

Example 02 Cont.

Actions:

1. Inserting (one or more) new employee tuples

2. Changing the salary of (one or more) existing employees

3. Changing the assignment of existing employees from one department to another

   one to update the totalSal of the employee's old department the other to update the totalSal of the employee's new department.

4. Deleting (one or more) employee tuples

   Automatically update the totalSal of the employee's department

Example 02 Cont.

Insert New

```
CREATE TRIGGER Total_sal1
AFTER INSERT ON Employee
FOR EACH ROW
    UPDATE Department
    SET totalSal = totalSal + NEW.Salary
    WHERE Dno = NEW.Dno;
```

Example 02 Cont.

Update Salary

```
CREATE TRIGGER  Total_sal2
AFTER UPDATE ON Employee
FOR EACH ROW
    UPDATE Department
    SET totalSal = totalSal + NEW.Salary - OLD.Salary
    WHERE Dno = NEW.Dno;
```

Example 02 Cont.

Change Department

```
CREATE TRIGGER  Total_sal3
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
UPDATE Department
SET totalSal = totalSal + NEW.Salary
WHERE Dno = NEW.Dno;
UPDATE Department
SET totalSal = totalSal - OLD.Salary
WHERE Dno = OLD.Dno;
END
```

Example 02 Cont.

Delete Employee

```
CREATE TRIGGER  Total_sal4
AFTER DELETE ON Employee
FOR EACH ROW
    UPDATE Department
    SET totalSal = totalSal - OLD.Salary
    WHERE Dno = OLD.Dno;
```

Activity

Consider the following *employee* table.

employee
Emp_number (PK) Last_name First_name Extension email Job_title

employee_audit
ID(PK) Emp_number Last_name Changed_date action

Suppose the DB administrator wants to log the changes occurring to *employee* table.

1. Create a new table named *employees_audit*
2. create a trigger that is invoked before a change is made to the employees table.

Answer

1.

```
CREATE TABLE employees_audit (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Emp_number INT NOT NULL,
    Last_name VARCHAR(50) NOT NULL,
    Changed_date DATETIME DEFAULT NULL,
    Action VARCHAR(50) DEFAULT NULL
);
```

Answer Cont.

2.

```
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
 INSERT INTO employee_audit
 SET action = 'update',
     Emp_number= OLD.Emp_number,
     Last_name = OLD.Last_name,
     Changed_date= NOW();
```