DATA INTEGRITY

INTEGRITY

- Ideally the data stored in the database system should always be correct and complete
- This is called data integrity
- However, users can make errors that brakes down the integrity of the data
- Errors are encountered while entering new data or modifying an existing piece of data

INTEGRITY

- Examples:
  - Suppose that the age of an employee must be between 18 and 70 but the user tries to enter 15 as the age
  - The officer tries to assign the ID number of an existing student to a new student as well
  - Registering a new student without entering his/her name

INTEGRITY

- Relational databases allows us to define rules that check the errors
- These rules are called as constraints
- If a statement used for entering or modifying data does not satisfy a constraint
  - execution is aborted
  - the user is warned by an error message

INTEGRITY

- But we cannot prevent all mistakes
- <u>Examples:</u>
  - Age of all employees must be between 18 and 70
  - The age of an employee is 35 but the user enters the age as 45
  - The database system cannot catch this error
- Our goal is to do our best by eliminating errors that can be detected by constraints

COSTRAINTS

- Constraints are defined in a CREATE TABLE statement
- When a statement that enters or modifies a piece data (INSERT INTO, UPDATE) is executed, all constraints defined by the user are checked
- If all constraints are satisfied, the statement is executed. **Otherwise, it is aborted**

COSTRAINTS

- There are different types of constraints
  - Uniqueness
  - NOT NULL
  - Check
  - Primary key
  - Foreign key

UNIQUENESS CONSTRAINT

UNIQUENESS CONSTRAINT

- Values in a column may be required to be different than each other
- <u>Example:</u> Two offices cannot be located at the same city
- We can use uniqueness constraints to impose such restrictions

UNIQUENESS CONSTRAINT

- One way to define uniqueness constraint is to add **UNIQUE** keyword to the end of the column

```
CREATE TABLE OFFICES(
    …
    LOCATION VARCHAR(15) UNIQUE,
    …
)
```

- This is called **column constraint** definition

UNIQUENESS CONSTRAINT

- Suppose that there is already a Boston office in the offices table.
- User tries to run the following statement.

```
INSERT INTO OFFICES  VALUES (53, ’Boston’, Northern, 4,
500000, 615000)
```

- This statement will not be executed. It will be aborted with an error message!!!!

UNIQUENESS CONSTRAINT

- An alternative way to impose a uniqueness constraint is to define it as a table constraint
- **Table constraint:** Define the constraint in a separate line of CREATE TABLE statement (not at the end of the column definition)

UNIQUENESS CONSTRAINT

- Table constraint definition:

```
CREATE TABLE OFFICES(
    …
    LOCATION VARCHAR(15),
    …
    UNIQUE(LOCATION),
    …
)
```

UNIQUENESS CONSTRAINT

- Use of table constraint form becomes mandatory if the constraint applies to more than one column together
- Example:
  - Suppose that we store the **area** code and **phone** numbers of customers in **separate columns**
  - We want to ensure that the whole phone number (area+number) of each customer is different than each other

UNIQUENESS CONSTRAINT

```
CREATE TABLE CUSTOMERS(
    …
    AREA CHAR(3),
    PNUMBER CHAR(7),
    …
    UNIQUE(AREA,PNUMBER),
    …
)
```

NOT NULL CONSTRAINT

NOT NULL CONSTRAINT

- It would unacceptable to leave some column values as empty (NULL) in a table
- <u>Example:</u>
  - Register a new employee without entering his/her name
- We can prevent execution of any statement that makes cells NULL in such columns
- This can be achieved using NOT NULL constraint

NOT NULL CONSTRAINT

- NOT NULL constraint can be defined **only as a column constraint**
- Just add the keyword NOT NULL to the end of the column of interest
- Example:

```
CREATE TABLE EMPLOYEES(
….
FL_NAME VARCHAR(15) NOT NULL,
….
)
```

NOT NULL CONSTRAINT

- If the user executes a statement that makes a cell NULL in that column,
  - the statement will not be executed
  - DBMS will report an error

- Example: will the following statements be executed?

  -
    ```
    INSERT INTO EMPLOYEES(EMP_ID)
    VALUES (111)
    ```

  -
    ```
    UPDATE EMPLOYEES
    SET FL_NAME=NULL
    WHERE EMP_ID=8
    ```

PRIMARY KEY CONSTRAINT

PRIMARY KEY CONSTRAINT

- A primary key value
  - cannot be used more than once in a column
  - cannot be NULL
- These constraints can be imposed using primary key constraint
- It can be defined as a column or table constraint

PRIMARY KEY CONSTRAINT

Example:

- Column constraint definition:

```
CREATE TABLE CUSTOMERS(
    CUST_ID INTEGER PRIMARY KEY,
    ….
)
```

PRIMARY KEY CONSTRAINT

- Table constraint definition:

```
CREATE TABLE CUSTOMERS(
    CUST_ID INTEGER,
    ….
    PRIMARY KEY (CUST_ID),
    ….
)
```

PRIMARY KEY CONSTRAINT

- If a primary key is a combination of more than one column, it must be defined as a table constraint
- Example:

```
CREATE TABLE PRODUCTS(
        MAN_ID CHAR(3),
        PROD_ID CHAR(5),
        ….
        PRIMARY KEY (MAN_ID, PROD_ID)
)
```

CHECK CONSTRAINT

CHECK CONSTRAINT

- SQL allows us to test a value entered into a table by checking if it satisfies a logical expression
- If the result of the logical expression is false, the statement trying to enter/modify the value (INSERT or UPDATE) is rejected
- CHECK constraint can be defined **only as a table constraints**

CHECK CONSTRAINT

<u>Example:</u>

- Age of an employee must be between 18 and 70
- We want to ensure that the values entered to this column will always be in this interval

```
CREATE TABLE EMPLOYEES(
....
AGE INTEGER,
....
CHECK (AGE BETWEEN 18 AND 70),
)
```

CHECK CONSTRAINT

- Any statement (INSERT, UPDATE) that tries to enter an age not between 18 to 70 will be aborted
- Examples:
  - INSERT INTO EMPLOYEES(AGE, …) VALUES (75, …)

  - UPDATE EMPLOYEES SET AGE=14
    WHERE EMP_ID=5

DEFAULT VALUES

DEFAULT VALUES

- Suppose that the value of a column is not specified in an INSERT INTO statement
- The value of the new cell is will be NULL
- We may want to assign a default value to the cells of that column instead of leaving them unspecified (NULL)
- This can be achieved defining a default value to that column.

DEFAULT VALUES

- Example: If the sales representative of a customer is not specified, we want to assign representative 10 by default.

```
CREATE TABLE CUSTOMERS(
….
CST_REP INTEGER DEFAULT 10,
….
)
```

DEFAULT VALUES

- INSERT INTO CUSTOMERS (CUST_ID, COMP_NAME) VALUES (125, ‘Doge Corp.’)

CUST_ID	COMP_NAME	CST_REP	MAX_CREDIT
…	….	….	…
125	Doge Corp.	10	NULL