**LAB03: Integrity Constraints**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission**:

* Submit a lab file named “int191\_lab03\_xxxxxxxxxxx.docx/.pdf” into the LEB2 system. xxxxxxxxxxx = your student id

**Due Date & Time**:

* Lecturer will inform the LAB03 due date and time in lab class.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Integrity Constraints**

**- Key Constraint**

This defines that certain attributes (keys) can be used to uniquely identify rows in a table. Examples include primary keys, candidate keys, and unique keys. These constraints prevent duplication of data.

**- Entity Integrity Constraint**

This guarantees that every row in a table is uniquely identifiable. It requires that the primary key must have unique values and cannot contain NULL. This ensures that no two rows are identical.

**- Referential Integrity Constraint**

This maintains consistency between related tables. A foreign key in one table must correspond to an existing primary key in another table. It prevents invalid references, ensuring that relationships between records are always valid.

**- Domain Constraint**

This ensures that the values in each column must come from a predefined domain. The domain specifies *the data type, format, or range of valid values*. For example, an age field must only contain positive numbers.

**- NOT NULL Constraint**

This specifies whether a column cannot be NULL values. It ensures that certain critical attributes (e.g., IDs, names) must always have a value.

**The SQL CREATE TABLE Statement**

## SQL provides the CREATE TABLE statement to create a new table in a given database. An SQL query to create a table must define the structure of a table. The structure consists of the name of a table and names of columns in the table with each column's data type. Note that each table must be uniquely named in a database.

**Syntax for creating a table**

**CREATE TABLE** **table\_name**(

column1 **datatype** [**PRIMARY KEY| UNIQUE| NOT NULL**],

column2 **datatype,**

column3 **datatype,**

.....

columnN **datatype,**

[**PRIMARY KEY**( one or more columns ),

**UNIQUE** (column),

**CHECK** (condition),

**FOREIGN KEY**(column) **REFERENCES** parent\_table\_name (pk\_column)

[**ON DELETE** *reference\_option*]

[**ON UPDATE** *reference\_option*]]

**);**

reference\_option:

**RESTRICT | CASCADE | SET NULL**

**Creating a Table from an Existing Table**

## Instead of creating a new table every time, one can also copy an existing table and its contents including its structure, into a new table. This can be done using a combination of the CREATE TABLE statement and the SELECT statement. Since its structure is copied, the new table will have the same column definitions as the original table. Furthermore, the new table would be populated using the existing values from the old table.

**Syntax**

## The basic syntax for creating a table from another table is as follows –

**CREATE TABLE** **NEW\_TABLE\_NAME**

**AS**

**SELECT [column1, column2...columnN]**

**FROM EXISTING\_TABLE\_NAME**

**WHERE Condition;**

## Here, column1, column2... are the fields of the existing table and the same would be used to create fields of the new table.

**SQL − ALTER TABLE Statement**

## The SQL ALTER TABLE command is a part of Data Definition Language (DDL) and modifies the structure of a table. The ALTER TABLE command can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself.

**Syntax:**

**ADD COLUMN:**

ALTER TABLE table\_name ADD column\_name datatype;

**DROP COLUMN:**

ALTER TABLE table\_name DROP COLUMN column\_name;

**ADD PRIMARY KEY**

ALTER TABLE table\_name

ADD [CONSTRAINT constraint\_name] PRIMARY KEY (column1, column2...);

**DROP PRIMARY KEY**

ALTER TABLE table\_name DROP PRIMARY KEY;

**ADD CONSTRAINT**

ALTER TABLE table\_name

ADD [CONSTRAINT constraint\_name] CONSTRAINT\_TYPE(column1, column2...);

Hint: CONSRAINT\_TYPE: UNIQUE, CHECK, FOREIGN KEY

**DROP CONSTRAINT**

ALTER TABLE table\_name DROP CONSTRAINT constraint\_name;

Reference: <https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html#foreign-key-examples>

**-- DML Syntax --**

**INSERT INTO table\_name|view\_name [(column\_list)]**

**VALUES (value1, value2,…, value3);**

**INSERT INTO table\_name|view\_name [(column\_list)]**

**SELECT column(s)**

**FROM table\_name| view\_name**

**[WHERE condition(s)];**

**UPDATE table\_name|view\_name**

**SET column = value [,column2 = value2,...]**

**[WHERE condition(s)];**

**DELETE table\_name|view\_name**

**[WHERE condition(s)];**

**Task 1: Using the** ''**stdXXX**'' **database and write SQL statements to answer the following questions.**

1. Write SQL statements to create tables based on the table structure below, stored in the “stdXXX” database.

STUDENTS

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Referenced Table |
| StudentID | INT | Primary Key |  |
| Fname | VARCHAR(50) | NOT NULL |  |
| Lname | VARCHAR(100) | NOT NULL |  |
| DOB | DATE | Check (DOB < '2014-01-01') |  |
| EMAIL | VARCHAR(100) | Check (EMAIL like '%@%') |  |
| AdmissionDate | DATE | NOT NULL |  |
| ProgramID | INT | Foreign Key | Program |

PROGRAMS

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Referenced Table |
| ProgramID | INT | Primary Key |  |
| ProgramCode | VARCHAR(10) | Unique, NOT NULL |  |
| ProgramName | VARCHAR(100) | NOT NULL |  |
| Faculty | VARCHAR(100) | NOT NULL |  |

1. Insert in the following data into the tables and save the data:

STUDENTS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StudentID | Fname | Lname | DOB | EMAIL | AdmissionDate | ProgramID |
| 6691 | Emi | McKinsey | 2006-09-01 | <NULL> | 2024-07-31 | 1 |
| 6692 | Bob | Loy | 2006-12-10 | <NULL> | 2024-07-31 | 2 |
| 6693 | Scott | Tiger | 2006-04-25 | scott@sit.com | 2024-07-31 | 1 |

PROGRAMS

|  |  |  |  |
| --- | --- | --- | --- |
| ProgramID | ProgramCode | ProgramName | Faculty |
| 1 | B.Sc.IT | Information Technology | School of Information Technology |
| 2 | B.Sc.CS | Computer Science | School of Information Technology |
| 3 | BA.DSI | Digital Service Innovation | School of Information Technology |

1. Try to insert the given data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StudentID | Fname | Lname | DOB | EMAIL | AdmissionDate | ProgramID |
| 6694 | Lisa | Mac | 2014-03-11 | <NULL> | 2024-07-31 | 4 |

Can the row be inserted?

If yes, please show all rows of the STUDENTS table.

If no, please show error code and error message and explain.

1. Try to insert the given data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StudentID | Fname | Lname | DOB | EMAIL | AdmissionDate | ProgramID |
| 6695 | Mary | MacKinsey | 2009-06-07 | mary.sit.kmutt | 2024-07-31 | <NULL> |

Can the row be inserted?

If yes, please show all rows of the STUDENTS table.

If no, please show error code and error message and explain.

1. Try to insert the given data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StudentID | Fname | Lname | DOB | EMAIL | AdmissionDate | ProgramID |
| 6696 | Mac | Donald | 2010-02-11 | mac@sit.kmutt | 2024-07-31 | <NULL> |

Can the row be inserted?

If yes, please show all rows of the STUDENTS table.

If no, please show error code and error message and explain.

1. Try to change the program ID of student ID 6693 from 1 to 2.

Can the row be modified?

If no, please show error code and error message and explain.

If yes, please show all data of student ID 6693.

1. Try to remove the row of program ID 1 from the PROGRAMS table.

Can the row be removed?

If no, please show error code and error message and explain.

If yes, what happens about the students of the program ID 1 in the STUDENTS table?

1. Try to remove the row of program ID 3 from the PROGRAMS table.

Can the row be removed?

If no, please show error code and error message and explain.

If yes, what happens about the students of the program ID 3 in the STUDENTS table?

1. Remove the existing foreign key of the STUDENTS table and add the foreign key constraint with ON DELETE CASCADE option into the STUDENTS table. If you not specify the constraint name, what is the constraint name of FK generated by the system?
2. Try to remove the row of program ID 2 from the PROGRAM table.

Can the row be removed?

If no, please show error code and error message and explain.

If yes, what happens about the students of the program ID 2 in the STUDENT table?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_