LAB NO.: 2 **Date:**

INTEGRITY CONSTRAINTS IN SQL

Objectives:

In this lab, student will be able to:

- Understand the use of integrity constraints.

Integrity Constraints:

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL.

- NOT NULL Constraint: Ensures that a column cannot have NULL value.
- DEFAULT Constraint: Provides a default value for a column when none is specified.
- UNIQUE Constraint: Ensures that all values in a column are different.
- PRIMARY Key: Uniquely identifies each row/record in a database table.
- FOREIGN Key: Uniquely identifies row/record in another database table.
- CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use ALTER TABLE statement to create constraints even after the table is created.

Dropping Constraints:

Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.

For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command:

```
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;
```

Integrity Constraints Syntax:

- Ensure the uniqueness of the primary key(PRIMARY KEY)
  →column_name data_type primary key
  → Primary key(column_name(s))
- Ensure the uniqueness of the candidate key which is not the primary key
  →column_name data_type unique
- Ensure that child records in related tables have a parent record.

→foreign key(column_name) references table_name(column_name)

- Delete child records when the parent record is deleted.
  →foreign key(column_name) references table_name(column_name) on delete cascade
- Ensure that columns always contain a value.
  →column_name data_type not null
- Ensure that a column contains a value within a set/specific range.
  →check (column_name in (value1, value2,..))
  →check (predicate)
- Ensure that a default value is placed in a column.
  →column_name data_type default (value)
- Naming Constraints: Constraints can have unique user defined name as given below
  →CONSTRAINT <constraint_name> <constraint_definition>
  Ex.: constraint account_pk primary key(account_number)
- To list constraints defined on the table:
  Select * from user_cons_columns where table_name='employee';
- Modifying Constraints:
  ALTER TABLE <table_name>
  ADD / MODIFY/DROP/DISABLE/ENABLE/VALIDATE/NOVALIDATE
  CONSTRAINT <constraint_name>

Built-in Functions:

- **LENGTH (string):** The Oracle/PLSQL LENGTH function returns the length of the specified string.
  Syntax: LENGTH(string1)

- **LOWER (string):** The Oracle/PLSQL LOWER function converts all letters in the specified string to lowercase. If there are characters in the string that are not letters, they are unaffected by this function.
  Syntax: LOWER(string)

- **SUBSTR(string, start, count):**The Oracle/PLSQL SUBSTR functions allows you to extract a substring from a string.
  Syntax: SUBSTR(string, start_position [, length])

- **UPPER (string):**The Oracle/PLSQL UPPER function converts all letters in the specified string to uppercase. If there are characters in the string that are not letters, they are unaffected by this function.

  Syntax: UPPER (string)

- **NVL (column name, substitute value):** The Oracle/PLSQL NVL function lets you substitute a value when a null value is encountered.
  Syntax: NVL (string, replace_with)
- **ROUND (value, precision):** The Oracle/PLSQL ROUND function returns a number rounded to a certain number of decimal places.
  Syntax: ROUND (number [, decimal places])
- **TO_CHAR (date1, format):** The Oracle/PLSQL TO_CHAR function converts a number or date to a string.
  Syntax: TO_CHAR(value [, format mask] [, nls_language])
- **LAST_DAY(date):**The Oracle/PLSQL LAST_DAY function returns the last day of the month based on a *date* value.
  Syntax: LAST_DAY(date)
- **MONTHS_BETWEEN (date1, date2):**The Oracle/PLSQL MONTHS_BETWEEN function returns the number of months between *date1* and *date2*.
  Syntax: MONTHS_BETWEEN (date1, date2)
- **NEXT_DAY(date1, 'day'):**The Oracle/PLSQL NEXT_DAY function returns the first weekday that is greater than a *date*.
  Syntax: NEXT_DAY (date, weekday)
- **TO_DATE (string, 'format'):**The Oracle/PLSQL TO_DATE function converts a string to a date.
  Syntax:TO_DATE(string1 [, format_mask] [, nls_language])

  Ex: to_date ('12021998', 'DDMMYYYY')

LAB EXERCISES:

Consider the following schema:
Employee (EmpNo, EmpName, Gender, Salary, Address, DNo)
Department (DeptNo, DeptName, Location)

1. Create Employee table with following constraints:
   - Make EmpNo as Primary key.
   - Do not allow EmpName, Gender, Salary and Address to have null values.
   - Allow Gender to have one of the two values: 'M', 'F'.
2. Create Department table with following:
   - Make DeptNo as Primary key
   - Make DeptName as candidate key
3. Make DNo of Employee as foreign key which refers to DeptNo of Department.
4. Insert few tuples into Employee and Department which satisfies the above constraints.

5. Try to insert few tuples into Employee and Department which violates some of the above constraints.
6. Try to modify/delete a tuple which violates a constraint.

   (**Ex:** Drop a department tuple which has one or more employees)
7. Modify the foreign key constraint of Employee table such that whenever a department tuple is deleted, the employees belonging to that department will also be deleted.

Naming Constraints:

8. Create a named constraint to set the default salary to 10000 and test the constraint by inserting a new record.

University Database Schema:

Student (ID, name,dept-name, tot-cred)
Instructor(ID, name, dept-name, salary)
Course (Course-id, title,dept-name, credits)
Takes (ID, course-id, sec-id, semester, year, grade)
Classroom (building, room-number, capacity)
Department (dept-name, building, budget)
Section (course-id, section-id, semester, year, building, room-number, time-slot-id)
Teaches (id, course-id, section-id, semester, year)
Advisor(s-id, i-id)
Time-slot (time-slot-id, day, start-time, end-time)
Prereq (course-id, prereq-id)

(Use University database for the exercise problems given below)

Retrieving records from a table:

9. List all Students with names and their department names.
10. List all instructors in CSE department.
11. Find the names of courses in CSE department which have 3 credits.
12. For the student with ID 12345 (or any other value), show all course-id and title of all courses registered for by the student.
13. List all the instructors whose salary is in between 40000 and 90000.

Retrieving records from multiple tables:

14. Display the IDs of all instructors who have never taught a course.
15. Find the student names, course names, and the year, for all students those who have attended classes in room-number 303.

Rename and Tuple Variables (Use as in select and from):

16. For all students who have opted courses in 2015, find their names and course id's with the attribute course title replaced by c-name.
17. Find the names of all instructors whose salary is greater than the salary of at least one instructor of CSE department and salary replaced by inst-salary.

String Operations (Use %, _, LIKE):

18. Find the names of all instructors whose department name includes the substring 'ch'.

Built-in Functions:

19. List the student names along with the length of the student names.
20. List the department names and 3 characters from 3rd position of each department name
21. List the instructor names in upper case.
22. Replace NULL with value1(say 0) for a column in any of the table
23. Display the salary and salary/3 rounded to nearest hundred from Instructor.

Add date of birth column (DOB) to Employee Table. Insert appropriate DOB values for different employees and try the exercise problems given below:

24. Display the birth date of all the employees in the following format:
    - 'DD-MON-YYYY'
    - 'DD-MON-YY'
    - 'DD-MM-YY'
25. List the employee names and the year (fully spelled out) in which they are born
    - 'YEAR'
    - 'Year'
    - 'year'
26. List the employee names and the day of the week (fully spelled out) in which they are born
    - 'DAY'
    - 'Day'
27. List the employee names and the month(fully spelled out) in which they are born
    - 'MONTH'
    - 'Month'
28. Find the last day of the month(and its day of the week) in which employee Mr. X is born
29. Find the age of all the employees
30. Find the Saturday following the Employee's 60th birthday
31. List the employees whose birthday falls in the given year X
32. List the employees whose birthday fall between the given years X and Y
33. List the employees who will retire on the given year X.

[Hint: use & with the variable name (e.g. &X) in the SQL query to read the value from the user]

ADDITIONAL EXERCISE:

1. Modify the employee table to check the salary of every employee to be greater than 5000.
2. Find the quarter of year from the given date.
3. Convert seconds to hours, minutes and seconds format.
4. Find the week of the year from the given date.
5. Find the names of all departments with instructor, and remove duplicates.
6. For all instructors who have taught some course, find their names and the course ID of the courses they taught.
7. Find all the instructors with the courses they taught.
8. List all the students with student name, department name, advisor name and the number of courses registered