Creating and Modifying Database Tables

Practical 9

Reminder:

Intellectual Property

□ Copyright must be seriously protected. The University takes a strong stand against any illegal photocopying and distributing of all materials provided to students. Students are forewarned of the consequences and the penalty that may be meted out if they are "caught in the act".

□ All the materials provided to student **SHOULD NOT** be posted/distributed at any online platform or any other ways possible without the permission.

Lesson Objectives

- ☐ Become acquainted with DDL.
- ☐ Learn how to define/create Oracle database tables.
- ☐ Review the table structure.
- ☐ List the data types that are available for columns.

DO NOT run any of the script for this practical!

Introduction to SQL

- Structured Query Language (SQL): The standard query language for relational databases.
  - Data Query Language (DQL)
    - View database data – Select.
  - **Data Definition Language (DDL)**
    - **Create new database objects.**
    - **Modify or delete existing database objects.**
  - Data Manipulation Language (DML)
    - Insert, update, delete database data.
  - Data Control Language (DCL)
    - Grant privileges and assign storage area to user.
  - Transaction Control Language (TCL)
    - Statement used to manage the changes made by DML.
    - COMMIT, ROLLBACK, SAVEPOINT.

Defining Database Tables

- To create a new table specify the:
  - table name
  - name of each data field
  - data type and size of each data field
- Oracle Naming Standard:
  - Series of rules Oracle Corporation established for naming all database objects
    - Objects must be from 1 to 30 characters long
    - Can contain letters, numbers, and the special symbols $, _, and #
    - Must begin with a character/letter
    - Must not be an Oracle reserved word
    - Must not duplicate the name of another object owned by the same user

Creating a Table

CREATE TABLE tablename
(column1 data_type,
column2 data_type,
…);

Data Types

- **Data type** specifies the kind of data that a field stores.

- Assigning a data type provides a means for error checking.

- Data types enable the DBMS to use storage space more efficiently by internally storing different types of data in different ways.

- Each column can hold only one type of data.

Data Types

- For character values:
    - CHAR/NCHAR (fixed size, max 2000 characters)
    - VARCHAR2/ NVARCHAR2 (variable size, max 4000 characters)
- For number values:
    - NUMBER (variable size, max precision 38 digits)
- For date and time values:
    - DATE, TIMESTAMP, INTERVAL
- For binary values (multimedia: JPG, WAV, MP3):
    - RAW (variable size, max 2000 bytes)
    - BLOB (variable size, max 4 Gig)
    - CLOB (variable size, max 4 billion characters)

Character Data Types

VARCHAR2	CHAR
Stores **variable-length** character data up to a maximum of 4,000 characters	**Fixed-length** character data up to a maximum size of 2,000 characters
Values in different records can have a different number of characters	Data values for different records all have the same number of characters
No trailing blank spaces is added	DBMS **adds trailing blank spaces** to the end of the entry to make the entry fill the maximum_size value
fieldname VARCHAR2(maximum_size)	fieldname CHAR[(maximum_size)]

Character Data Types

□ NVARCHAR2 and NCHAR

- Analogous to VARCHAR2 and CHAR but use Unicode rather than ASCII.

- Used to hold character data in languages other than English.

Number Data Types

□ Stores negative, positive, fixed, and floating point numbers between 10^{-130} and 10^{125}, with precision up to 38 decimal places.

□ General Syntax: *fieldname NUMBER [([precision,] [scale])]*

□ Integer: *fieldname NUMBER(precision)*

□ Fixed Point: *fieldname NUMBER[([precision],[scale])]*

□ Floating Point: *fieldname NUMBER*

The value 7,456,123.89 will display as follows

- NUMBER(9)
- NUMBER(9,1)
- NUMBER(*,1)
- NUMBER(9,2)
- NUMBER(6)
- NUMBER(7,-2)
- NUMBER
- FLOAT

The value 7,456,123.89 will display as follows

NUMBER(9)	7456124
NUMBER(9,1)	7456123.9
NUMBER(*,1)	7456123.9
NUMBER(9,2)	7456123.89
NUMBER(6)	[not accepted exceeds precision]
NUMBER(7,-2)	7456100
NUMBER	7456123.89
FLOAT	7456123.89

Date and Time Data Types

- DATE
  - Dates from December 31, 4712 BC to December 31, 9999 AD
  - Default format DD-MON-RR
  - Default time format HH:MI:SS A.M.
  - Eg. 21-AUG-03 17:25:30 A.M.
  - *fieldname DATE*
- TIMESTAMP
  - Similar to DATE but stores fractional seconds (precise time values).
  - *fieldname TIMESTAMP (fractional_seconds_precision)*
  - *Eg. 15-AUG-06 09:26:01:123975 a.m.*

Constraints

- Rules that restrict the data values that you can enter into a field in a database table.

- Types of constraint:
  - **Integrity Constraints** (Entity Integrity, Referential Integrity)
    - Define primary and foreign keys.
  - **Value/Field /Domain Constraint** (Domain Integrity)
    - Limits the value that can be placed in a specific field, irrespective of values that exist in other table records.

Constraint Naming Convention

☐ *tablename_ fieldname_constraintID*

Constraint Type	ConstraintID Abbreviation	Oracle9i Constraint Type Identifier
PRIMARY KEY	pk	P
FOREIGN KEY	fk	R
CHECK CONDITION	cc	C
NOT NULL	nn	N
UNIQUE	uk	U

Table 2-2 Common constraintID abbreviations

Integrity Constraints

☐ Define **primary key** fields

COLUMN LEVEL	TABLE LEVEL
Loc_id Number(6) PRIMARY KEY @ Loc_id Number(6) CONSTRAINT location_loc_id_pk PRIMARY KEY	CONSTRAINT location_loc_id_pk PRIMARY KEY (loc_id)

☐ Specify **foreign keys** and their corresponding table and column references

COLUMN LEVEL	TABLE LEVEL
Locid number(6) CONSTRAINT faculty_loc_id_fk REFERENCES location(loc_id)	CONSTRAINT faculty_loc_id_fk FOREIGN KEY (locid) REFERENCES location(loc_id)

☐ Specify **composite keys**

COLUMN LEVEL	TABLE LEVEL
Cannot be done at column level	CONSTRAINT cust_agent_id_pk PRIMARY KEY (cust_id, agent_id)

Value Constraints

☐ **Check conditions**: field value must be a specific value or fall within a range of values

COLUMN LEVEL	TABLE LEVEL
s_class CHAR(2) CONSTRAINT student_s_class_cc CHECK s_class in('FR', 'SO','JR', 'SR')	CONSTRAINT student_s_class_cc CHECK s_class in('FR', 'SO','JR', 'SR')

☐ **NOT NULL constraints**: specify whether a field value can be NULL

COLUMN LEVEL	TABLE LEVEL
job_id VARCHAR2(10) NOT NULL	*Cannot be done at table level*

☐ **Unique constraints**: specify that a field must have a unique value for every table record

COLUMN LEVEL	TABLE LEVEL
email varchar2(50) CONSTRAINT emp_email_uk UNIQUE	CONSTRAINT emp_email_uk UNIQUE(email)

Violating Constraints

□ When you have constraints in place on columns, an error is returned if you try to violate the constraint rules.

Default Option

- Specify a default value for a column during an insert

- Literal values, SQL functions are legal values

- Another column's name or pseudocolumn are illegal values

- The default data type must match the column data type.
  - hire_date DATE DEFAULT SYSDATE, …
  - s_state CHAR(2) DEFAULT 'PK'

Creating Constraints

- **At column level**
  - References a single column

```
Create table client
(cl_no NUMBER(4) CONSTRAINT client_cl_no_pk PRIMARY KEY,
cl_first VARCHAR2(20),
cl_last VARCHAR2(20)
);
```

- **At table level**
  - Are listed after all the table columns have been defined.

```
Create table client
(cl_no NUMBER(4),
cl_first VARCHAR2(20),
cl_last VARCHAR2(20),
CONSTRAINT client_clno_pk PRIMARY KEY(cl_no),
CONSTRAINT uniqfnln unique(cl_first,cl_last)
);
```

Basic Rules For Constraints

- UNIQUE, PRIMARY KEY, FOREIGN KEY and CHECK constraints can be defined at either the *column* or *table* level.

- Constraints that refer to more than one column must be defined at the *table* level.

- The NOT NULL constraint can be specified only at the *column* level.

- If the word CONSTRAINT is used, you must give the constraint a name.

Create a Table

CREATE TABLE location

(loc_id NUMBER(6),

bldg_code VARCHAR2(10),

room VARCHAR2(6) CONSTRAINT loc_room_nn
   NOT NULL,

capacity NUMBER(5),

CONSTRAINT location_loc_id_pk PRIMARY KEY
   (loc_id));

Create Tables

- CREATE TABLE for faculty, student

CREATE TABLE faculty
(f_id NUMBER(6),
f_last VARCHAR2(30),
f_first VARCHAR2(30),
f_mi CHAR(1),
loc_id NUMBER(6),
f_phone VARCHAR2(10),
f_rank VARCHAR2(8),
f_pin NUMBER(4),
f_image BLOB,
CONSTRAINT faculty_f_id_pk PRIMARY KEY(f_id),
CONSTRAINT faculty_loc_id_fk FOREIGN KEY(loc_id) REFERENCES
location(loc_id));

```sql
CREATE TABLE student
(s_id NUMBER(6),
s_last VARCHAR2(30),
s_first VARCHAR2(30),
s_mi CHAR(1),
s_add VARCHAR2(25),
s_city VARCHAR2(20),
s_state CHAR(2),
s_zip VARCHAR2(9),
s_phone VARCHAR2(10),
s_class CHAR(2) CONSTRAINT student_s_class_cc CHECK ((s_class = 'FR')
OR (s_class = 'SO') OR (s_class = 'JR') OR (s_class = 'SR')),
s_dob DATE,
s_pin NUMBER(4),
f_id NUMBER(6),
time_enrolled INTERVAL YEAR TO MONTH,
CONSTRAINT student_s_id_pk PRIMARY KEY (s_id),
CONSTRAINT student_f_id_fk FOREIGN KEY (f_id) REFERENCES faculty(f_id));
```

Checking Table Constraints

SELECT constraint_name, constraint_type

FROM user_constraints

WHERE table_name = 'FACULTY';

* Take note of the result.

Removing Tables

- To remove table:
  - DROP TABLE tablename
  - Use with caution.
  - To delete foreign key constraints, add "*cascade constraints*".

DROP TABLE location;

DROP TABLE location CASCADE CONSTRAINTS;

Table Constraints

SELECT constraint_name, constraint_type

FROM user_constraints

WHERE table_name = 'FACULTY';

* Observe the result. Explain.

Identify the lines with errors. For each error, **explain** and **provide the fix**.

Line no.	
1	CREATE TABLE STUD ENROL(
2	s_id NUMBER(6),
3	c_sec_id NUMBER(6) CONSTRAINT enr_pk PRIMARY KEY (s_id, c_sec_id),
4	grade CHAR(1)
5	CONSTRAINT enr_csecid_nn NOT NULL,
6	CONSTRAINT enr_sid_fk FOREIGN KEY (s_id) REFERENCES student(s_id),
7	CONSTRAINT enr_csecid_fk FOREIGN KEY (c_sec_id) REFERENCE course_section (c_sec_id),
8	CONSTRAINT enr_grade_cc CHECK ((grade = 'A') AND (grade = 'B') AND (grade = 'C') AND (grade = 'D') AND (grade = 'F')
9	;

- Try the exercise given.