LO4 and LO5 – Table Creation with DDL

DDL – Data Definition Language (LO4 and LO5)

1. We are changing the structure of the database.
2. Commands which create new or modify existing database objects.
3. When you execute a DDL command, the database immediately changes
4. Ex: CREATE, ALTER, DROP, RENAME

DML – Data Manipulation Language (LO6)

1. We are manipulating the contents of the database.
2. When you execute a DML command, you must explicitly save the command to make the new data values commit.
3. Ex: INSERT, UPDATE, DELETE, and SELECT(LO7).

SQL statements are embedded in scripting languages or programming languages.

# Data Definition Language

CREATE TABLE statement

* Syntax:  
  CREATE TABLE TableName(  
   columnName datatype[constraint]  
   {, columnName datatype [constraint]}…  
   {, [constraints]}…

);

* Each column definition has 3 parts
  + Name of the column
  + Data type of the column
  + Column constraints (optional)

DROP TABLE statement

* Syntax:  
  DROP TABLE[schema.]tableName[CASCADE CONSTRAINTS]
* Cascade constraints allows a parent table to be dropped even if child table exists.

# Oracle Data Types

* A data type of a column specifies what kind of information the column will hold (characters, numbers, dates, etc.) so that the system will know how the data is to be physically stored
* It is very important to pick the correct datatype because it is usually difficult to change it later. Often you will need to use some kind of conversion function.
* Oracle does not use INT or TINYINT – instead we use NUMBER with an appropriate size. INTEGER and SMALLINT are the same as NUMBER(38).
* Oracle does not reasonably support DECIMAL (it lets you use DECIMAL, but ends up treating it as if it were a NUMBER(38) – with no decimal portion). Instead, use NUMBER(size, #decimals) or DECIMAL(size, #decimal).
* Oracle stores date and time values using DATE datatype. The default format is DD-MON-YY.

# Number Data Types

* Stores negative, positive, fixed, and floating point numbers between 10^-130 and 10^125 with a precision up to 38 decimal places.
* columnName NUMBER [([precision,][scale])]
  + precision stores the total number of digits
  + Scale is the number of digits on the right of the decimal point
* Subtypes:
  + Integer numbers
    - columnName NUMBER(precision)
    - NUMBER(3) – three digits with no decimal
    - Decimals are rounded, error generated if inserting too large of a number
  + Fixed-point numbers
    - columnName NUMBER(6,2)
    - Fixed number of digits, and decimal places
    - Example above 6 digits, 4 on the left, and 2 on the right
  + Floating-point numbers
    - columnName NUMBER
    - Decimal can appear anywhere from before the first digit to after the last digit or be omitted entirely.

# Character Datatypes

* CHAR
  + columnName CHAR [(max size)]
  + stores fixed-length character data up to a maximum of 2000 characters
  + max size is optional – default of 1
  + More efficient than VARCHAR2
* VARCHAR2
  + columnName VARCHAR2(maxSize)
  + Always uses VARCHAR2 and not VARCHAR
  + Stores variable length character data up to 4000 characters
  + Must specify a max size
* NVARCHAR2 and NCHAR
  + Only allows ASCII character set. Similar to VARCHAR2 and CHAR.

# Date and Time Datatypes

* Datatypes that store date and time values include **datetime** data subtypes, which store actual date and time values, and the **interval** data subtypes, which store an elapsed time interval between two date/time values. The main datetime subtypes are DATE and TIMESTAMP. The interval subtypes include INTERVAL YEAR TO MONTH and INTERVAL DAY TO SECOND.
  + DATE
    - columnName DATE (More common)
    - Stores dates from December 31, 4712 BC to December 31, 4712 AD.
    - Stores the century, year, month, day, hour, minute, and second
    - Default format is ‘DD-MON-YY’ 05-Oct-19
    - Default time format is HH:MI:SS AM – using the 12 hour clock
  + TIMESTAMP
    - columnName TIMESTAMP (fractional seconds precision)
    - stores the date values similar to DATE datatype, except it also stores fractional seconds.
    - The default is 6 decimal places for fractional seconds precision

# Large Object (LOB) Datatypes

* columnName LOB\_data\_type
* Used to store binary data, such as digitized sounds or images or references to binary files from a word processor or spreadsheet. –Stores a lot of data in one field,
* There are 4 LOB datatypes:

1. BLOB – Binary LOB stores up to 4GB of data in the database
2. BFILE – Binary files stores a reference to a binary file located outside the database in a file maintained by the OS
3. CLOB – Character LOB stores up to 4GB of character data in the database
4. NCLOB – Character LOB that supports 2-byte character codes, stores in the database – up to 4GB

# Constraints

* Constraints are rules that restrict the data values that you can enter into a column in a database table.
* There are two types of constraints:
  + Integrity constraints: Which define primary and foreign keys
  + Value constraints: Which define specific data values or date ranges that must be followed, or whether a column can be null, etc.
* There are two levels of constraints
  + Table constraints: Table constraints restrict the data with respect to all other values in the table. – All values in the table
    - Ex: Primary key constraint – specifies that a column value must be unique and cannot appear in more than one record.
  + Column constraints: limit the value that can be placed in a specific column, regardless of values that exist in other records. – Do not care about other columns or rows ( ex: age is an integer and not a varchar2)
* You can place constraint definitions at the end of the CREATE TABLE command, after you declare all of the columns.
* OR you can place each constraint definition within the column definition.
* Each constraint must have a unique constraint name. You can define your own or omit them and Oracle will name your constraint. Max of 30 characters.

|  |  |
| --- | --- |
| **Constraint Type** | **Constraint ID Abbreviation** |
| Primary | pk |
| Foreign | fk |
| Check | cc |
| Null / Not Null | nn |
| Unique | uk |

Use TableName\_columnName\_constraintIDAbbreviation

Ex: Project\_projectID\_pk

* Types of constraints
  + Integrity constraints
    - Defines primary key columns and specifies foreign keys corresponding table and column references.
    - Primary Keys
      * Syntax:
        + Within a column

columnName datatype CONSTRAINT constraintName\_pk PRIMARY KEY

* + - * + At the end of the CREATE TABLE

CONSTRAINT constraintName\_pk PRIMARY KEY (columnName)

* + - Foreign Keys
      * Syntax:
        + Within a column

columnName datatype CONSTRAINT constraintName\_fk REFERENCES parentTable(columnNameInParent)

* + - * + At the end of the CREATE TABLE

CONSTRAINT constraintName\_fk FOREIGN KEY(columnNameInTable) REFERENCES parentTable(columnNameInParent)

* + - * + Note: Before you can create a Foreign Key constraint the parent table must already exist.
    - Composite Key
      * You will need to list all of the columns involved.
      * Syntax (only creatable at the end of the CREATE TABLE statement):
        + CONSTRAINT constraintName\_pk PRIMARY KEY (columnName, columnName2, columnName3, …
      * Defining composite foreign key works the same way
  + Value Constraints
    - Column-level constraints that restrict the data values that users can enter into a given column
    - Check, Not Null, and Default
    - Check constraints
      * Specify that a column value must be a specific value or fall within a range of values.
      * Make sure you specify all conditions because once the table is created and populated it is difficult to modify the constraint.
      * DBMS must be able to evaluate each check condition to either true or false. You can combine expressions using logical operators AND and OR.
      * Syntax:
        + CONSTRAINT constraintName\_cc CHECK (expression)
      * Examples:
        + CONSTRAINT course\_grade\_cc CHECK ((grade > 0) AND (grade <= 100))
    - NOT NULL Constraints
      * Specify whether the user MUST enter a value for a specific field or whether the value can be NULL.
      * NULL is the default for fields except PK
      * Primary keys automatically NOT NULL
      * Syntax (must be in the column definition):
        + columnName datatype CONSTRAINT constraintName\_nn NOT NULL
    - Default Constraints
      * Specifies that a particular column has a default value that the DBMS automatically inserts for every record, unless the user specifies an alternate value.
      * These must be created in the column definition and do not begin with the CONSTRAINT keyword – many systems do not consider default a constraint, instead it is a field property.
      * Syntax:
        + columnName datatype DEFAULT defaultValue
    - Unique Constraints
      * Specifies that a column must have a unique value for every record
      * All primary keys are automatically assigned a unique constraint
      * Null values are allowed
      * Syntax:
        + columnName datatype CONSTRAINT constraintName\_uk UNIQUE
        + CONSTRAINT constraintName\_uk UNIQUE (columnName)

\*\*\*\* Stop here and do examples up to “—data dictionary views”, then continue from here

# System Tables

We can retrieve information about a variety of database objects using different data dictionary views. These views simply contain data (structure and contents) about our data; this Metadata can be queried, and the result is a “view”.

|  |  |
| --- | --- |
| Object Name | Object Type |
| User\_objects | All schema objects |
| User\_tables | Schema tables |
| User\_indexes | Indexes created for keys and improving query performance |
| User\_views | Schema views |
| User\_sequences | Sequences used to create auto numbers |
| User\_constraints | Constraints placed on columns |
| Cons\_columns | Columns that have constraints |
| Tab\_columns | All schema columns |

\*\*\* Stop here and do examples up to “--Alter Table”, then continue from here

## Unrestricted Actions and Restricted Actions

Various actions involving the database schema are classified as restricted or unrestricted.

* Modifying and Deleting Database Tables

1. Unrestricted Actions – specifics that can always be changed – making the DB easier to use, not changing or breaking the structure – harmless for the most part
   * + Renaming a table
     + Adding new fields
     + Deleting non key fields
     + Making a field larger
     + Deleting constraints
2. Restricted Action – changing structure in a way that it could break things

|  |  |
| --- | --- |
| **Restricted Action** | **Restriction** |
| Deleting a table from a user schema | Allowed only if the table does not contain any fields that other tables reference as foreign keys |
| Changing an existing field’s data type | Only if existing data in the field is compatible with the data type |
| Decreasing the size of a field | Only if all existing data is null or field has no values |
| Adding a Foreign Key constraint to an existing field | Allowed only if current field values are NULL or existing in the referenced table |
| Adding a Primary Key constraint to an existing field | Allowed only if current field values are not null and unique |
| Adding a Unique constraint to an existing field | Only if existing values are already Unique |
| \*\*Adding a Check constraint | Unrestricted – but constraint applies only to new changes to the table. |
| \*\*Changing a field’s Default value | Unrestricted – applies only to rows added after the change |

## Renaming Existing Tables

Syntax: RENAME oldTableName TO newTableName - Bad practice to do this to a DB that is in place

* The DBMS automatically transfers to the new name the integrity constraints, indexes, and privileges that referenced the old name.
* Objects such as functions, procedures, views, etc. (as well as external code or reports) are not automatically updated – this can cause problems.

## Alter Table - Making Changes to Existing Tables – Know this

Always start with ALTER TABLE tableName

Is similar to CREATE TABLE syntax

* After a table has been created, its structure, properties, or constraints can be changed using the ALTER statement.
* Altering an empty table usually poses no difficulties. Altering a populated table may require ninja DBA skills to fix the data before the alteration can occur.

### Adding Fields to an Existing Table

* Unrestricted action

Syntax: ALTER TABLE tableName ADD ( columnName datatype Constraints…);

* Basically a column definition inside the ADD brackets

### Deleting Fields from Existing Table

* In many cases the DB will automatically delete the associated constraints on the column, but it depends on the DBMS.

Syntax: ALTER TABLE tableName DROP COLUMN columnName;

* A column can be renamed with ALTER TABLE tableName RENAME COLUMN old TO newName

### Adding and Deleting Constraints Using Alter

* Adding a new constraint is a restricted (see above)
* Deleting constraints is unrestricted
* Syntax:
  + ALTER TABLE tableName ADD CONSTRAINT constrantName constraintDefinition;
  + constraintDefinition would be in same form as adding at the end of the CREATE TABLE
  + ALTER TABLE tableName DROP CONSTRAINT constraintName;

### Modifying Existing Fields Using Alter Table

* Can modify only if existing values are compatible with the new data type
* Making a field size larger is unrestricted, but making it smaller is restricted
* ALTER TABLE tableName MODIFY (columnName new\_data\_declaration)

\*\* Do remainder of examples in script