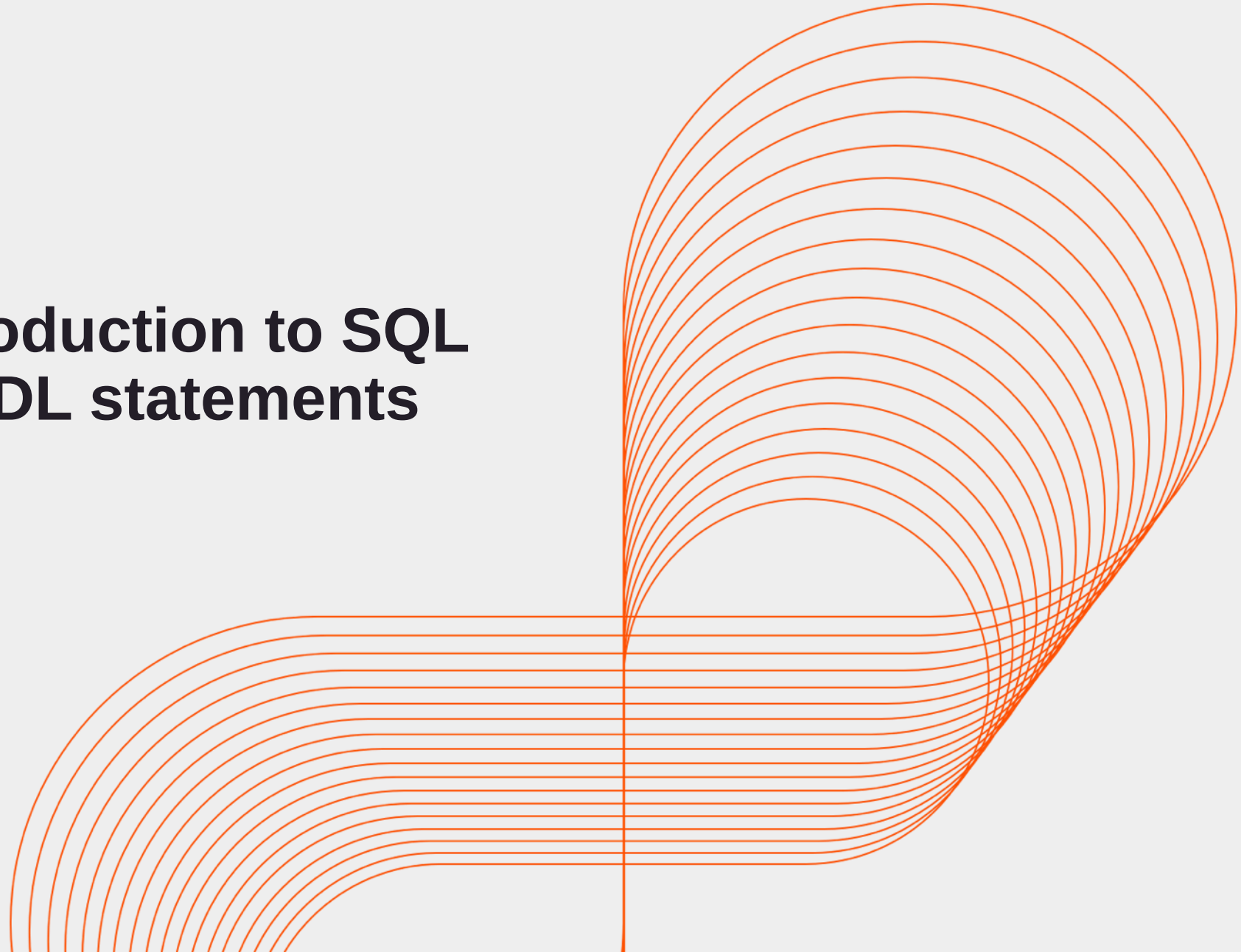
Persistent

Nugget 2: Introduction to SQL and various DDL statements

Persistent University



Key Learning Points

1. Basics of SQL
2. Types of SQL statements
3. DDL statements in detail

Persistent

Basics of SQL

- SQL stands for **Structured Query Language**. It is a special purpose language designed for managing data in Relational database managements systems (RDBMS).
- It is a common language for variety of relational databases.
- It is basically used to communicate with a database.
- It is ANSI standard language; however there are variations that extend the ANSI subset. They are T-SQL, PLSQL, Sybase, Ingres, MySQL etc.

Advantages of SQL

- Efficient
- Easy to learn and use
- Functionality complete(With SQL, user can define, manipulate, retrieve data in the tables)
- Universal
- Scalable
- Data integrity (SQL has many functionalities to maintain the integrity and consistency of data)

Types of SQL statements

Type	Description	Statement	
DATA RETRIEVAL	Retrieves data from the database	SELECT	
DATA MANIPULATION LANGUAGE (DML)	used to manage data within schema objects	INSERT	insert data into a table
		UPDATE	updates existing data within a table
		DELETE	deletes records from a table, the space for the records remain
		MERGE	UPSERT operation (insert or update)
DATA DEFINITION LANGUAGE (DDL)	used to define the database structure or schema	CREATE	to create objects in the database
		ALTER	alters the structure of the database
		DROP	delete objects from the database
TRANSACTION CONTROL (TCL)	Used to manage the changes made by DML statements. Changes to the data can be grouped together into logical transaction.	COMMIT	To save work done
		ROLLBACK	restore database to original since the last COMMIT
		SAVEPOINT	identify a point in a transaction to which you can later roll back
DATA CNTROL LANGUAGE (DCL)	Gives or removes access rights to both the oracle DB and structures within it.	GRANT	gives user's access privileges to database
		REVOKE	withdraw access privileges given with the GRANT command

DDL statements in detail

- **Data Definition Language** (DDL) statements are used to define the database structure or schema. They are used to manage different objects within the database such as Table, View, sequence, index, constraints etc. DDL does an "auto commit”. Oracle implicitly commits the current transaction before and after every DDL statement.

In this session we are going to cover creation and management of Table Object and constraints.

- Naming Rules for Table names and Column names:
  - Must begin with letter
  - Must be 1-30 characters long
  - Must contain only A-Z, a-z, 0-9, _ , $ and #
  - Must not duplicate the name of another object owned by same user.
  - Must not be reserved keyword.
  - Use the descriptive names

CREATE statement

- Syntax:

Create table [schema].table

(column datatype [default expr]

[,……]);

- Schema : name of the owner’s name
- Table: Name of the table
- Column: Name of the Column
- Default expr: specifies the default value for that column, if value is omitted while inserting data. It is a optional part.
- Datatype: column’s data type and length.

CREATE statement

- Example:

```
Create table new_emp
(emp_id number,
last_name varchar2(20),
salary number,
hire_date date);
```

- User can confirm the table creation using DESCRIBE command

  Syntax: DESCRIBE <tablename>

  Example: DESCRIBE new_emp

Data Types

Data type	Description
Varchar2(size)	Variable length character data Min size 1, Max Size 4000
Char[(size)]	Fixed length character data. Min and default size 1, Max size 2000
Number (p,s)	Variable length numeric data. Number having precision p and scale s. Precision is total number of decimal digits and scale is number of digits to the right of decimal point.
Date	Date and time values
Long	Variable length character data up to 2 gigabyte
CLOB	Character data up to 4 gigabytes
RAW and Long RAW	Raw Binary data
BLOB	Binary data up to 4 GBs
BFILE	Binary data stored in an external file; up to 4 GBs
ROWID	A 64 base numbering system representing unique address of the row in a table.

There are some more Date Time data types
TIMESTAMP
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND

CREATE Statement

- Create Table Using Subquery

  Syntax: CREATE TABLE tablename [(column,column1…)]

  AS subquery;

  - This syntax combines creation of table and insertion of rows returned by the subquery.
  - If the column specifications are given, the number of columns must equal the number of subquery columns. The table will be created with specified column names in CREATE TABLE clause.
  - If no column specifications are given, the column names in the table are same as column names in the subquery.
  - Column definition can contain column names and default values
  - The integrity rules are not passed onto the new table, only the column data type definition will be passed.

- Example:

  Create table dept10 as

  Select employee_id,last_name,department_id from employee

  where department_id=10;

ALTER Statement

- The ALTER Statement is used to:

  # Add new column
  # Define default value for the column
  # Add constraint
  # Modify existing column
  # drop a column

- Syntax:

```
ALTER TABLE tablename
ADD  (Column datatype [default expr] [, Column datatype]….);

ALTER TABLE tablename
MODIFY (Column datatype [default expr] [, Column datatype]….);

ALTER TABLE tablename
DROP COLUMN  columnname;
```

ALTER Statement

- Examples:
  - Alter table Job_history

    Add phone_number varchar2(20);

  - Alter table Job_history

    Modify job_id varchar2(30);

  - Alter table Job_history

    drop column phone_number;

ALTER Statement

- Guidelines for Modifying a column
  - User can increase the width of numeric or character column.
  - User can decrease the width of a column only if the column contains only null values or table has no rows.
  - User can change the data type only if the column contains null values.
  - User can convert CHAR column to VARCHAR2 data type or VARCHAR2 column to CHAR data type only if the column contains null values or if you do not change the size.
  - A change to the default value of a column affects only subsequent insertions to the table.

- Guidelines for dropping a column
  - The column may or may not contain data.
  - Only one column can be dropped at a time
  - Table must have at least one column remaining in it after it is altered.
  - Once a column is dropped, it can not be recovered.

DROP Statement

- Syntax:

```
DROP TABLE Tablename;
```

  - All data and structure in the table is deleted.
  - Any pending transactions are committed.
  - All Indexes are dropped.
  - User can not rollback the DROP TABLE statement.

- Example:

```
Drop table job_history;
```

TRUNCATE Statement

- Syntax:

```
TRUNCATE TABLE Tablename;
```

  - Removes all rows from a table
  - Releases the storage space used by the table
  - You can not rollback row removal
  - Alternatively, you can remove rows using DELETE statement

- Example:

```
Truncate table dept10;
```

RENAME Statement

- Syntax:

  RENAME Tablename to New_tablename

  - User can use Rename statement to rename, table, view, sequence and synonym.
  - User must be the owner of the object.

- Example:

  RENAME new_emp to new_emp1;

COMMENT Statement

- Syntax:

  COMMENT ON TABLE tablename | COLUMN table.columnname IS ‘text’

- Examples:

  COMMENT ON TABLE EMPLOYEE IS 'It contains employee details’;

  COMMENT ON COLUMN EMPLOYEE.job_id IS 'contains job type details';

- Tables ALL_COL_COMMENTS and ALL_TAB_COMMENTS can be used to view the comments in data dictionary

Creating Constraints

- As we have already seen in last session,
  - constraint can be created at column level or table level.
  - constraint can be created at the time of table creation or after the table has been created.

- Syntax for defining constraints while table creation

```
Create table tablename
        (column datatype [default expr] [ column level constraint ],
        column datatype,
        ……,
        [table level constraint [, ….] );
```

Creating Constraints

- Table level constraint: References one or more columns and is defined separately from the definition of the columns. At table level, User can define any type of constraint; except NOT NULL

  Syntax:

  Column1,

  Column2,….

  [CONSTRAINT constraint_name] constraint_type (column1)

- Column level constraint: References a single column and is defined within a specification for the owning column. User can define any type of constraint at column level.

  Syntax:

  Column [CONSTRAINT constraint_name] constraint_type,

Creating Constraints

- Examples:

  Primary key at Column level

```
CREATE TABLE Employees(
Emp_id number(6) PRIMARY KEY,
First_name varchar(15),
…. );
```

  Note: Unique index is automatically created for Primary Key column(Emp_id) with system generated name.

  Primary key at Table level

```
CREATE TABLE Dept(
Deptid number(6) ,
Dept_name varchar(15),
…. ,
Constraint dept_idpk PRIMARY KEY(deptid) );
```

Creating Constraints

- Examples:

  Foreign key at Column level

```
CREATE TABLE Employees(
Emp_id number(6),
Job_id varchar(15) NOT NULL,                    -- NOT NULL at column level
Deptid number(6) constraint emp_dptid_fk references dept(deptid);
```

  Foreign key at Table level

```
CREATE TABLE Employees(
Emp_id number(6),
Job_id varchar(15) NOT NULL,
Deptid number(6),
Constraint emp_dptid_fk foreign key (deptid) references dept(deptid);
```

Creating Constraints

- Examples:

  Check constraint

```
CREATE TABLE Employees (
Emp_id number(6),
First_name varchar(15),
….
Job_id varchar(15) NOT NULL,                    -- NOT NULL at column level
Salary number(10)   constraint emp_sal_min      check (salary > 0) );
```

Managing Constraints

- Managing constraints using ALTER statement after the table has been created.
  - Syntax to add the constraint:

    Alter table tablename

    Add constraint constraint_name Constraint_type(column)

  - Syntax to Modify constraint:

    Alter table tablename

    Modify column_name datatype

    constraint constraint_name Constraint_type(column)

  - Syntax to remove a constraint:

    Alter table tablename

    drop constraint constraint_name

Managing Constraints

- Examples:
  - Alter table employees

    Add constraint emp_man_fk

    Foreign key (manager_id) references employees(employee_id);

    -- This syntax is used to create table level constraint

  - Alter table employees

    Modify salary number(6) constraint emp_sal_chk NOT NULL;

    -- This syntax is used to create column level constraint after table creation.

  - Alter table employees

    Drop constraint emp_man_fk;

Managing Constraints

- Disabling Constraints: Disable clause of the ALTER statement is used to deactivate integrity constraint.
  - Syntax:

    Alter table tablename

    Disable/Enable Constraint constrantName [cascade]

  - Example:

    Alter table employees

    Disable constraint emp_emp_id_pk;

Reference Material: Sites

www.sqlbasics.com

http://www.orafaq.com/faq/what_are_the_difference_between_ddl_dml_and_dcl_commands

http://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_1001.htm

http://www.oracle-dba-online.com/sql/oracle_data_definition_language.htm

http://www.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10759/statements_1001.htm

Session 2: Summary

With this we have come to an end of our second session where we discussed concept of SQL, type of SQL statements and various DDL statements in detail.

- At the end of Nugget 2, we see that you are now able to answer following questions:
  - Explain “Meaning and advantages of SQL”.
  - What are the types of SQL statements
  - Explain “Various DDL statements with example”
  - Explain “Table level and column level constraints with examples”

Persistent

Thank you!

Persistent University