# CSE–302

# DATABASE MANAGEMENT SYSTEMS SESSIONAL

# ABSTRACT DATA TYPE

# Abstract Data Type

- Sometimes we may want a type of data which holds all types of data including numbers, chars and special characters something like this.
- We can not achieve this using pre-defined types.
- In Oracle 7 there was no way to extend the data typing.

# Abstract Data Type (Contd.)

- C++ Class Concept:

```
Class Employee
{
    int Emp_ID;
    int Room_no;
    int House_no;
    int Street_no;
    string City;
};
```

# Abstract Data Type (Contd.)

```
Class Address
{
        int House_no;
        int Street_no;
        string City;
        };

Class Employee
{
    int Emp_ID;
    int Room_no;
    Address Addr;
};
```

# Abstract Data Type (Contd.)

- For example, to select all of the address information from a table, we have to specify all of the columns in the group.

```
CREATE TABLE STUDENT
(
ID NUMBER(2),
NAME VARCHAR2(20),
HOUSENO NUMBER(5),
STREETNO NUMBER(5),
CITY VARCHAR(10)
);
```

These data columns can be group together by data type

# Abstract Data Type (Contd.)

```
CREATE TYPE ADDR AS OBJECT
(
        HOUSENO NUMBER(3),
        STREETNO NUMBER(5),
        City VARCHAR(10)
);
```

The data type ADDR is created

**Now we can create table using new abstract data types that we have created:**

```
CREATE TABLE STUDENT
(
        ID NUMBER(2),
        NAME VARCHAR2(20),
        ADDRESS ADDR
);
```

**Now we can reference ADDRESS in our sql as if it were a primitive data type.**

# Abstract Data Type (Contd.)

- We can't insert data into ADDR. The reason is straightforward.
- A data type describes data, it does not store data.
- We cannot store data in a NUMBER data type, so we cannot store data in a data type that we define, either.
- To store data, you have to create a table that uses your data type.

# Abstract Data Type (Contd.)

```
INSERT INTO STUDENT
 VALUES
(
        1, ‘A', ADDR(111, 43, ‘DHAKA’)
);
```

# Abstract Data Type (Contd.)

- Show House No and City of all students

SELECT S.ADDRESS.HOUSENO, S.ADDRESS.CITY
FROM STUDENT S;

| ADDRESS.HOUSENO | ADDRESS.CITY |
|---|---|
| 111 | DHAKA |
| 234 | KHULNA |

# Abstract Data Type (Contd.)

- Show House No of the students who live in a City which has a name starting with D

```
SELECT S.ADDRESS.HOUSENO FROM STUDENT S
WHERE  S.ADDRESS.CITY LIKE 'D%' ;
```

| ADDRESS.HOUSENO |
| --- |
| 111 |

# Abstract Data Type (Contd.)

- UPDATING ADT TABLES

```
UPDATE STUDENT S SET S.ADDRESS.CITY = 'RAJSHAHI'
WHERE S.ADDRESS.HOUSENO = 111;
```

- DELETE FROM ADT TABLES

```
DELETE STUDENT S
WHERE S.ADDRESS.HOUSENO = 111;
```

- DROPPING ADT

```
DROP TYPE ADDR;
```

# Nesting In Abstract Data Type

```
CREATE TYPE ADDRESS_TY AS OBJECT  (
        STREET VARCHAR2(20),
        CITY VARCHAR2(10),
        PIN NUMBER
);

CREATE TYPE PERSON_TY AS OBJECT
(
        NAME VARCHAR2 (20),
        ADDRESS ADDRESS_TY
);
```

- Now PERSON_TY contains Name and address of a person we can use this to create table.

# Nesting In Abstract Data Type (Contd.)

```
CREATE TABLE CUSTOMER
(
CUSTOMER_ID NUMBER,
PERSON PERSON_TY
);
```

- To Insert rows into CUSTOMER do following.

```
INSERT INTO CUSTOMER
VALUES
(1, PERSON_TY ('SANAN', ADDRESS_TY ('102 Dhanmondi', 'DHAKA', 10101))
);
```

# Nesting In Abstract Data Type (Contd.)

- To select data from customer table:

```
SELECT CUSTOMER_ID,
C.PERSON.NAME,
C.PERSON.ADDRESS.STREET
FROM CUSTOMER C;
```

# GRANT AND REVOKE

# Data Control Language Statements

- Data Control Language Statements are used to grant privileges on tables, views, sequences, synonyms, procedures to other users or roles.
- GRANT :Use to grant privileges to other users or roles.
  REVOKE :Use to take back privileges granted to other users and roles.

- Privileges are of two types :
  - **System Privileges:** System Privileges are normally granted by a DBA to users. Examples of system privileges are CREATE SESSION, CREATE TABLE, CREATE USER etc.

  - **Object privileges:** Object privileges means privileges on objects such as tables, views, synonyms, procedure. These are granted by owner of the object.

# GRANT AND REVOKE

- Object privileges are:

| | |
|---|---|
| ALTER | Change the table definition with the ALTER TABLE statement. |
| DELETE | Remove rows from the table with the DELETE statement.<br>Note: You must grant the SELECT privilege on the table along with the DELETE privilege. |
| INDEX | Create an index on the table with the CREATE INDEX statement. |
| INSERT | Add new rows to the table with the INSERT statement. |
| REFERENCES | Create a constraint that refers to the table. You cannot grant this privilege to a role. |
| SELECT | Query the table with the SELECT statement. |
| UPDATE | Change data in the table with the UPDATE statement. |
| | Note: You must grant the SELECT privilege on the table along with the UPDATE privilege. |

# User creation

- create user user2 identified by "user2";
- grant create session to user2;
- grant unlimited tablespace to user2;.

# GRANT AND REVOKE (Contd.)

## GRANT:

Grant is use to grant privileges on tables, view, procedure to other users or roles

## SYNTAX:

GRANT < Privilege List> ON <Relation name or View name> TO <User / role list>;

# GRANT AND REVOKE (Contd.)

## GRANT:

- Suppose user1 owns employee table. Now s/he wants to grant select, update, insert privilege on this table to other user "user2".

  **GRANT SELECT, UPDATE, INSERT ON EMPLOYEE TO USER2;**

- Suppose user1 wants to grant all privileges on employee table to user2.

  **GRANT ALL ON EMPLOYEE TO USER2;**

- If user1 wants to grant select privilege on employee to all other users of the database.

  **GRANT SELECT ON EMPLOYEE TO PUBLIC;**

# GRANT AND REVOKE (Contd.)

- Suppose user1 wants to grant update and insert privilege on only certain columns not on all the columns then include the column names in grant statement. For example, if s/he wants to grant update privilege on emp_name column only and insert privilege on emp_no and emp_name columns only, then the following statement will do that job:

```
GRANT UPDATE (EMP_NAME), INSERT
(EMP_NO, EMP_NAME) ON EMPLOYEE
 TO USER2;
```

# GRANT AND REVOKE (Contd.)

- To grant select statement on employee table to user2 and to make user2 be able further pass on this privilege you have to give WITH GRANT OPTION clause in GRANT statement like this.

**GRANT SELECT ON EMPLOYEE TO USER2 WITH GRANT OPTION;**

# GRANT AND REVOKE (Contd.)

## REVOKE:

Revoke is used to revoke privileges already granted to other users.

## SYNTAX:

REVOKE < Privilege List> ON <Relation name or View name>
FROM <User / role list>;

# GRANT AND REVOKE (Contd.)

## REVOKE:

- To revoke select, update, insert privilege user1 has granted to user2:

  **REVOKE SELECT, UPDATE, INSERT ON EMPLOYEE FROM USER2;**

- To revoke select statement on employee granted to public give the following command.

  **REVOKE SELECT ON EMPLYEE FROM PUBLIC;**

- To revoke update privilege on emp_name column and insert privilege on emp_no and emp_name columns give the following revoke statement.

  **REVOKE UPDATE, INSERT ON EMPLOYEE FROM USER2;**

# GRANT AND REVOKE (Contd.)

REVOKE:

- **Note : You cannot take back column level privileges. Suppose you just want to take back insert privilege on emp_name column then you have to take back the whole insert privilege.**

# GRANT AND REVOKE (Contd.)

ROLES:

- A role is a group of Privileges. A role is very handy in managing privileges, Particularly in such situation when number of users should have the same set of privileges.

- **For example:** you have four users : user1, user2, user3, user4 in the database. To these users you want to grant select ,update privilege on employee table, select, delete privilege on department table. To do this first create a role by giving the following statement

  **CREATE ROLE CLERKS**

# GRANT AND REVOKE (Contd.)

ROLES:

- Then grant privileges to this role.

```
GRANT SELECT,UPDATE ON EMPLOYEE TO
CLERKS;

GRANT SELECT,DELETE ON
DEPTARTMENT TO CLERKS;
```

- Now grant this clerks role to users like this

```
GRANT CLERKS TO USER1, USER2, USER3, USER4 ;
```

- Now all 4 users have all the privileges to granted on clerks role.

# GRANT AND REVOKE (Contd.)

- Suppose after one month you want grant delete on privilege on employee table all these users then just grant this privilege to clerks role and automatically all the users will have the privilege.

  **GRANT DELETE ON EMPLOYEE TO CLERKS;**

- If you want to take back update privilege on employee table from these users just take it back from clerks role.

  **REVOKE UPDATE ON EMPLOYEE FROM CLERKS;**

- To Drop a role

  **DROP ROLE CLERKS;**

# LISTING INFORMATION ABOUT PRIVILEGES

- To see which table privileges are granted by you to other users.

  SELECT * FROM USER_TAB_PRIVS_MADE

- To see which table privileges are granted to you by other users

  SELECT * FROM USER_TAB_PRIVS_RECD;

- To see which column level privileges are granted by you to other users.

  SELECT * FROM USER_COL_PRIVS_MADE

- To see which column level privileges are granted to you by other users

  SELECT * FROM USER_COL_PRIVS_RECD;

- To see which privileges are granted to roles

  SELECT * FROM USER_ROLE_PRIVS;

# ORACLE SYNONYM

# Oracle Synonym

- A synonym is an alias for a schema object.
- Synonyms can provide a level of security by masking the name and owner of an object and by providing location transparency for remote objects of a distributed database.
- Also, they are convenient to use and reduce the complexity of SQL statements for database users.

# Oracle Synonym (Contd.)

- Synonyms can be created as in the database. PRIVATE (by default) or PUBLIC.
- Public synonyms are available to all users
- Private synonyms exist only in specific user schema
- The syntax for a synonym is as follows:
- To create private synonym in your own schema

**CREATE SYNONYM SYNONYM_NAME FOR OBJECT_NAME**

- To create public synonym in your own schema

**CREATE PUBLIC SYNONYM SYNONYM_NAME FOR OBJECT_NAME**

# Oracle Synonym (Contd.)

- Assume we have a database with a schema called USER1. This schema contains a table called ORDERS.

```
CREATE PUBLIC SYNONYM ORDERS_DATA FOR USER1.ORDERS;
```

- From now on, USER1 or any other user who has the access can query this table using the synonym:

```
SELECT * FROM ORDERS_DATA;
```

# Oracle Synonym (Contd.)

- To create a private synonym in your own schema, you must have the CREATE SYNONYM system privilege.
- To create a PUBLIC synonym, you must have the CREATE PUBLIC SYNONYM system privilege.

**GRANT CREATE SYNOYM, CREATE PUBLIC SYNONYM TO USERNAME;** // by database administrator

# Synonyms for Synonyms

- A curious feature of a synonym is that each may have its own synonym or many synonyms.
- Let's assume that we have a table with a formal name such as USER1_PURCHASE_TABLE. Now we want to create a synonym PURCHASE_TABLE

**CREATE SYNONYM PURCHASE_TABLE FOR USER1_PURCHASE_TABLE.**

- Now let us create a synonym PT for the synonym PURCHASE_TABLE

**CREATE SYNONYM PT FOR PURCHASE_TABLE;**

# Virtual Column

- A virtual column is a column whose value is computed as a reaction for a defined operation.
- This means that they are computed “on the fly” – only when needed.
- It’s accessible like any other column except that there is no physical space associated with it.
- If a SQL query doesn’t reference a virtual column, the value is not calculated.
- Of course, it’s impossible to insert a value into a virtual column. The attempt will cause a SQL error.
- It is a feature of Oracle 11g.

# Virtual Column (Contd.)

SYNTAX:

```
CREATE TABLE TABLE_NAME
(....
 .....
 ......
 COLUMN_NAME [DATATYPE] [GENERATED ALWAYS] AS
 (EXPRESSION) [VIRTUAL]
 );
```

- Keyword **AS** is sufficient to create a virtual column.
- Others: GENERATED ALWAYS and VIRTUAL are optional as well as datatype. If datatype is omitted, the virtual column datatype is based on the result of the expression.

# Virtual Column (Contd.)

**SYNTAX:**

- CREATE TABLE Product
  (
  ID NUMBER,
  Name VARCHAR2(30),
  Price NUMBER,
  VAT Number,
  PRICE_INCLUDING_VAT **AS (Price+Price*VAT)**
  );

- INSERT INTO Product VALUES (1, 'BOOK', 20, 0.07);
- SELECT * FROM Product

| ID | Name | Price | VAT | PRICE_INCLUDING_VAT |
|---|---|---|---|---|
| 1 | book | 20 | 0.07 | 21.4 |

# Virtual Column (Contd.)

- To get information about an expression that produces the value of a virtual column, check the data_default column in view USER_TAB_COLS:

```
SELECT column_name, data_type, data_default
FROM USER_TAB_COLS
WHERE  table_name = 'PRODUCT';
```

| | COLUMN NAME | DATA_TYPE | DATA_DEFAULT |
|---|---|---|---|
| 1 | ID | NUMBER | (NULL) |
| 2 | NAME | VARCHAR2 | (NULL) |
| 3 | PRICE | NUMBER | (NULL) |
| 4 | VAT | NUMBER | (NULL) |
| 5 | PRICE_INCLUDING_VAT | NUMBER | PRICE + PRICE * VAT |

# Truncate Table Command

- The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.
- You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.
- Syntax:

**TRUNCATE TABLE TABLE_NAME;**

# Truncate Table Command

- There are some arguments on the difference between truncate and delete command.  As they can be used to do the same thing like deleting information. But some of the differences are:
- We can use where clause in DELETE command but can't use it with Truncate command.
- Delete creates a log in the transaction log for every row of the table. Truncate just removes pages used by the table and just creates a log for the pages. So truncate works a bit faster.
- For  detail ed concept :
- https://www.mssqltips.com/sqlservertip/1080/deleting-data-in-sql-server-with-truncate-vs-delete-commands/

# THANK YOU