IT2201 / IT2601 / IT2564 / IT2621 / IT2521 / IT2323

Database Management Systems

Unit 8 (Part C)

Structured Query Language (Insert, Update, Delete, DDL)

1

Inserts/Updates/Deletes in SQL

- INSERT statement ;
  - Add a new row into a table.
  - Copy data from one table to another.
- UPDATE statement ;
  - Modify attribute values of existing rows.
- DELETE statement ;
  - Drop any number of rows from a table.
- COMMIT ;
  - To commit the changes, i.e. make the change permanent
- ROLLBACK ;
  - To rollback (discard) the changes

2

INSERT Statement

- Create a single row

INSERT INTO *table-name*
[(*column1*, ... , *columnN*)]
VALUES (*value1*, ..., *valueN*) ;

- *column1, column2, …* is optional.
  - If omitted, SQL assumes a list of all columns in the table.
- *value list* must match *column list* as follows:
  - Number of items in each list must be the same.
  - Must be direct correspondence in position of items in two lists.
  - Data type of each item in *value list* must be compatible with data type of corresponding column.

3

INSERT Statement

Example 1

INSERT INTO PRODUCT
VALUES (115, 'PRC', 108, '6/box') ;

The product table has the following columns:-

- PROD_NUM number(3)
- SUPPL_CODE varchar2(3)
- UNIT_PRICE number(6, 2)
- REMARKS varchar2(15)

4

INSERT Statement

Example 2

```
INSERT INTO
 ORDERS ( ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, PO_NUM )
 VALUES ( 1024, '20-May-1994', 101, '12345' );
```

Note:

Must supply values for those columns defined as 'NOT NULL', and with no default value defined.

Result:

ORDER_NUM	ORDER_DATE	CUSTOMER_NUM	SHIP_INSTRUCT
1024	20-MAY-94	101	

BACKLOG	PO_NUM	SHIP_DATE	SHIP_WEIGHT	SHIP_CHARGE	PAID_DATE
	12345				

5

INSERT Statement

- Create a group of new rows using data selected from other tables.

```
INSERT INTO   table-name
              [(column1, ..., columnN)]
SELECT expression ;
```

Example 3

```
INSERT INTO CUST1
  SELECT * FROM CUSTOMER ;
```

6

INSERT Statement

Example 4

Suppose a follow-up call is required for every order that has been paid but not shipped.
Create those rows in the CUST_CALLS table.

```
INSERT INTO CUST_CALLS
            ( CUSTOMER_NUM, CALL-DESCR )
SELECT      CUSTOMER_NUM, ORDER_NUM
FROM        ORDERS
WHERE       PAID_DATE IS NOT NULL
AND         SHIP_DATE IS NULL ;
```

- Restrictions on the INSERT - SELECT
  - cannot contain an **ORDER BY** clause.
  - cannot refer to the table into which rows are inserted.

7

UPDATE Statement

- Format/Syntax

```
UPDATE   table-name
SET      column-name1 = data-value1
         [,column-name = data-value2...]
[WHERE   condition] ;
```

- **SET** clause specifies column(s) that are to be updated, and the corresponding new value(s) to be set for the column(s).
- **WHERE** clause is optional:
  - If omitted, named columns are updated for all rows in table.
  - If specified, only those rows that satisfy the condition are updated.
- New data-value(s) must be compatible with data type for corresponding column.

8

UPDATE Statement

Example 1

```
UPDATE      CUSTOMER
  SET       FNAME = 'Barnaby',
            LNAME = 'Dorfler'
  WHERE     CUSTOMER_NUM = 113;
```

Example 2

Write an update statement to show that the supplier "HRO" has raised all prices by 5%.

```
UPDATE      PRODUCT
SET         UNIT_PRICE = UNIT_PRICE * 1.05
WHERE       SUPPL_CODE = 'HRO';
```

9

UPDATE Statement

Example 3

Suppose that the ANZA Corporation issues a safety recall of their tennis balls. As a result, any unshipped orders that include product number 6 from supplier "ANZ" must be backlogged.

```
UPDATE      ORDERS
SET         BACKLOG = 'Y'
WHERE       SHIP_DATE IS NULL AND
            ORDER_NUM IN
           (1005, 1006, 1010, 1013, 1022);
```

10

DELETE Statement

- Format/Syntax

DELETE FROM table-name
[**WHERE** condition] ;

- **WHERE** clause is optional:
  - If omitted, all rows are deleted from table. This does not delete the table.
  - If *condition* is specified, only those rows that satisfy the *condition* are deleted.

11

DELETE Statement

Example 1 (to delete all rows)

```
DELETE  FROM  CUSTOMER;
```

Example 2 (to delete a specific row)

```
DELETE  FROM CUSTOMER
   WHERE      CUSTOMER_NUM = 175;
```

Example 3 (to delete specific rows)

```
DELETE  FROM CUSTOMER
   WHERE      COMPANY = 'Druid Cyclery';
```

12

DELETE Statement

Example 4

Suppose some rows of product table contain incorrect supplier codes (non-existent). Write a DELETE statement to delete these rows so that they can be re-entered.

Solution:

One way to develop a DELETE **statement with a complicated condition is first to develop a** SELECT **statement that returns precisely the rows to be deleted.**

```
SELECT *  FROM  PRODUCT p
WHERE     ( SELECT COUNT(*)
              FROM    SUPPLIER  S
              WHERE  S.SUPPL_CODE = p.SUPPL_CODE
          ) = 0;
```

Change the SELECT * **to** DELETE **after testing.**

13

Data Definition Language (DDL)

- Subtopics for DDL:
  - Define table structure, know the data types supported by the Oracle DBMS
  - Define integrity constraints using SQL, including :
    - Primary key (Entity integrity)
    - Foreign key (Referential integrity)
    - Required data constraint (Not Null)
    - Default constraint
    - Domain constraint
  - Define indexes, understand usage of indexes
  - Define views
  - Alter table structure
  - Delete tables, views, indexes

14

Data Definition Language (DDL)

- SQL DDL allows database objects such as tables, views, and indexes to be created and destroyed.

- Main SQL DDL statements are:

CREATE/ALTER TABLE	**DROP TABLE**
CREATE INDEX	**DROP INDEX**
CREATE VIEW	**DROP VIEW**

How to change index/view?

15

Create Table

```
CREATE TABLE  TableName
    {(colName dataType [NOT NULL] [UNIQUE]
    [DEFAULT defaultOption] [CHECK searchCondition] [,...]}

    [CONSTRAINT constraint_name
     PRIMARY KEY (listOfColumns),]

    {[UNIQUE (listOfColumns),] [...,]}

    {[CONSTRAINT constraint_name
     FOREIGN KEY (listOfFKColumns)
    REFERENCES ParentTableName [(listOfCKColumns)],
            [ON UPDATE referentialAction]
            [ON DELETE referentialAction ]] [,...]}

    {[CHECK searchCondition)] [,...] } );
```

16

Creating Tables

- A Table definition
  - Consists of a list of definitions of columns that make up a row in the table.

```
CREATE TABLE  table-name
  ( column1      data-type [NULL | NOT NULL] ,
    ...,
    ...,
    columnN      data-type [NULL | NOT NULL] ) ;
```

  - Creates a table with one or more *columns* of the specified *data type*.
  - **NULL** (default) indicates whether column can contain *nulls*.
  - With **NOT NULL**, system rejects any attempt to insert a null in the column.
  - Primary keys should always be specified as **NOT NULL**.

17

Creating Tables

- Partial list of Oracle data types
  - Varchar2(n) – stores variable length character data (up to 4000 bytes)
  - Number (l, d) – stores numbers of length l with d decimal digits; e.g number(5,2) – stores data of up to 999.99
  - Date – stores data from 1 Jan 4712 BC to 31Dec 4712 AD

Example (create table)

```
CREATE TABLE CUST_TEST
(CUST_NUM      NUMBER(5)       NOT NULL,
LAST_NAME      VARCHAR2(15)    NULL,
FIRST_NAME     VARCHAR2(8)     NULL,
STATE          VARCHAR2(2)     NULL );
```

18

Creating Constraints

- What is a constraint
  - A constraint implements a business rule that restricts values stored in a table.
  - Constraints are implemented as either:
    - **Table constraints** that apply to the entire table or
    - **Column constraints** that apply to a single column.

19

Creating Constraints

- If applicable, create the following constraints
  - **Primary Key constraint**
    - identifies the primary key of a table.
  - **Foreign Key constraint**
    - if a value exists, it must be a primary key in the referenced table.
  - **Not Null constraint**
    - specifies that null values are not allowed.
  - **Unique constraint**
    - prevents duplicate values for a column or group of columns.
  - **Default constraint**
    - if value for a column is not specify during an insert, it will take the default given.
  - **Domain constraint**
    - to specify the set of allowable values that a column can have.

20

Creating Constraints

Example 1 (create table with Primary Key & Foreign Key constraint)

```
CREATE TABLE CUST_TEST
   (    CUST_NUM          NUMBER(5)    NOT NULL,
        LAST_NAME         VARCHAR2(15),
        FIRST_NAME        VARCHAR2(8),
        STATE             VARCHAR2(2),
        CONSTRAINT cust_num_pk
        PRIMARY KEY ( CUST_NUM ),
        CONSTRAINT state_fk
        FOREIGN KEY( STATE ) REFERENCES STATE(STATE_CODE)
   );
```

Example 2 (create table with NOT NULL constraint)

```
CREATE  TABLE   NEWITEMS
   (    ITEM_NUM          NUMBER(5),
        SUPPL_CODE        VARCHAR2(3)    NOT NULL,
        DESCR             VARCHAR2(30)       );
```

21

Creating Constraints

Example 3 (create a table with Default constraint)

```
CREATE  TABLE   ACCOUNTS
   ( ACC_NUM         NUMBER(12)  DEFAULT 0001,
     ACC_TYPE        VARCHAR2(1) DEFAULT 'A',
     ACC_DESCR       VARCHAR2(30),
     PRIMARY KEY   ( ACC_NUM ) );
```

Example 4 (create table with domain constraints)

```
CREATE TABLE CUST_TEST
   ( CUST_NUM        NUMBER(5)     NOT NULL,
     LAST_NAME       VARCHAR2(15),
     FIRST_NAME      VARCHAR2(8),
     SEX             VARCHAR2(1)CHECK ( SEX IN ('M','F') ),
     BAL             NUMBER(3) CHECK ( BAL BETWEEN 0 AND 999 ),
     STATE           VARCHAR2(2),
      PRIMARY KEY ( CUST_NUM ),
      FOREIGN KEY( STATE ) REFERENCES STATE( STATE_CODE )
);
```

22

Referential Integrity

- A Foreign Key is a column, or set of columns, that links each row in the child table containing the foreign key to the row of the parent table containing the matching primary key value.
- Referential integrity means that, if the foreign key contains a value, that value must refer to an existing row in the parent table.
- ISO standard supports the definition of foreign keys with the FOREIGN KEY clause in the CREATE and ALTER TABLE statements, for example :

  FOREIGN KEY (branchNo) REFERENCES Branch (branchNo)

23

Referential Integrity

- Any INSERT/UPDATE that attempts to create FK value in child table without matching primary key value in parent is rejected.
- Action taken by SQL that attempts to update/delete a primary key value in parent table with matching rows in child is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:
  - CASCADE - Delete row from parent and delete matching rows in child, and so on in cascading manner.
  - SET NULL - Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are not defined as NOT NULL.
  - SET DEFAULT - Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns
  - NO ACTION - Reject delete from parent. Default.

24

Referential Integrity

Example 5 (indirect violation of a constraint)

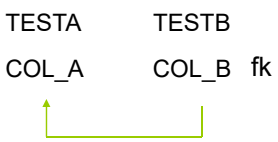
```
CREATE TABLE TESTA
   (COL_A      NUMBER(5)    PRIMARY KEY );

CREATE TABLE TESTB
   (COL_B      NUMBER(5),
     FOREIGN KEY( COL_B ) REFERENCES TESTA ( COL_A ) );

INSERT INTO TESTA VALUES (1);
INSERT INTO TESTB VALUES (1);
```

DELETE FROM TESTA WHERE COL_A = 1; What happens?

Default action for ON DELETE is No Action.
This SQL will fail as there is a child record.



25

Referential Integrity

Example 6 (use the ON DELETE CASCADE option)

```
CREATE TABLE TESTA
   (COL_A      NUMBER(5)    PRIMARY KEY );

CREATE TABLE TESTB
   (COL_B      NUMBER(5),
     FOREIGN KEY( COL_B ) REFERENCES TESTA ( COL_A )
     ON DELETE CASCADE );

INSERT INTO TESTA VALUES (1);
INSERT INTO TESTB VALUES (1);
```

DELETE FROM TESTA WHERE COL_A = 1; What happens?

The matching child record will also be deleted.

26

Creating Indexes

- What is an Index
  - **Indexes** are structures which points to the data rows for faster retrieval.

- Format/Syntax

  CREATE [UNIQUE] INDEX index-name
  ON table-name (column-name [ASC | DESC][, …]);

Examples

CREATE UNIQUE INDEX LNAME_IX ON CUSTOMER (LNAME);

CREATE INDEX LNAME_IX ON CUSTOMER (LNAME DESC);

27

Creating Indexes

- How do indexes improve retrieval?

Suppose the following query is executed very frequently :-
SELECT LNAME FROM CUSTOMER ORDER BY LNAME ASC;

Example

Performance can be improved by creating an index on the ORDER BY **column.**

CREATE INDEX LNAME_IX ON CUSTOMER (LNAME ASC);

The database server traverse the pages of the LNAME_IX from left to right and retrieves the key values from smallest key to the largest key.

LNAME_IX	Row_id
Ang	100
Boo	76
Chua	20
Lim	30
…	

28

Creating Views

- What is a view
  - A view like a base table, has a name, rows and columns.
  - A **virtual table** that is derived from other base tables or views.
- Views are defined by
  - a name
  - a list of attribute names
  - a query that selects rows and columns from underlying tables.

29

Creating Views

Example (a view to list all customers' names)

```
CREATE VIEW NAME_ONLY AS
   SELECT CUSTOMER_NUM, FNAME, LNAME FROM CUSTOMER;
```

Example (a view to list customers' full address)

```
CREATE VIEW FULL_ADDR AS
   SELECT CUSTOMER_NUM, ADDRESS1, ADDRESS2,
          CITY, S.STATE_NAME, ZIPCODE
      FROM CUSTOMER C
      INNER JOIN STATE S ON C.STATE_CODE = S.STATE_CODE;
```

QUESTION:

i) Without using the view, write a query to retrieve the customer number, address, city, state name and zipcode of customers.

ii) Rewrite (i) using the view created :

Answer : select * from full_addr ;

30

Creating Views

- Advantages of Views
  - Data independence
  - Improved security
  - Reduced complexity
- Disadvantages of views
  - Update restriction
  - Structure restriction
  - Performance

31

Drop Statements

- Removing any elements in Database
  - The **DROP** command can be used to remove any database elements.
  - Format/Syntax
    - **DROP TABLE** table-name;
    - **DROP VIEW** view-name;
    - **DROP INDEX** index-name;

32

Altering Tables

- Changing Table Definitions
  - The definition of a base table can be changed by using the **ALTER TABLE** command.
  - The alter table action includes:
    - **Add a new column to a table.**
    - **Drop a column from a table.**
    - **Add a new table constraint.**
    - **Drop a table constraint.**
    - **Set a default for a column.**
    - **Drop a default for a column.**

33

Altering Tables

- Format/Syntax

ALTER TABLE tablename

[ADD [COLUMN] columnName dataType **[NOT NULL] [UNIQUE]**
[DEFAULT defaultOption] [CHECK (searchCondition)]

[DROP [COLUMN] columnName [RESTRICT | CASCADE]]

[ADD [CONSTRAINT [ConstraintName]] tableConstraintDefinition]

[DROP CONSTRAINT ConstraintName [RESTRICT | CASCADE]]

[ALTER [COLUMN] SET DEFAULT defaultOption]

[ALTER [COLUMN] DROP DEFAULT];

34

Altering Tables

Example (add a new column)

```
ALTER TABLE   ITEMS
ADD  ( ITEM_WEIGHT NUMBER (6,2)   NOT NULL ) ;
```

Example (drop constraint, add a new constraint)

```
ALTER TABLE order_detail
DROP CONSTRAINT suppl_code_fk;

ALTER TABLE order_detail
ADD (
   CONSTRAINT  suppl_code_fk
   FOREIGN KEY (suppl_code)
   REFERENCES SUPPLIER (suppl_code)
) ;
```

35

Altering Tables

Example (change column definition)

```
ALTER TABLE  staff
   ALTER        position DROP DEFAULT;

ALTER TABLE staff
   ALTER        sex SET DEFAULT 'F' ;
```

36