Chapter 7 – Section 7.2 & 7.3

Objectives: to learn
• The basic commands and functions of SQL
  – use to define tables and indexes
  – to use insert, delete, & update table rows
• About database transactions and using the commit & rollback commands
• About using nested queries for inserting and copying data.

Outline
– Review CREATE & DROP tables
– CREATE INDEX
– User’s data dictionary
– INSERT: adding rows
• Database Transactions
  • COMMIT
  • ROLLBACK
– DELETE: deleting rows
– UPDATE: changing the row’s data
– Copying a table with a nested select
– Renaming a table

Review - Creating a Table Structure

• Create Table with attributes & constraints
  – On Delete Restrict is the default & not specified for foreign key constraints.

```
CREATE TABLE PRODUCT
    ( P_CODE VARCHAR2(10) NOT NULL,
      P_DESCRIPT VARCHAR2(35) NOT NULL,
      P_INDATE DATE DEFAULT SYSDATE,
      P_MIN NUMBER(8) NOT NULL
        CONSTRAINT CK_P_MIN CHECK(P_MIN > 0),
      P_PRICE NUMBER(8,2) NOT NULL,
      V_CODE NUMBER,
    CONSTRAINT PK_PRODUCT PRIMARY KEY (P_CODE),
    CONSTRAINT FK_PRODUCT FOREIGN KEY (V_CODE)
      REFERENCES VENDOR(V_CODE) ON DELETE CASCADE );
```

Review - Deleting a Table Structure

• Syntax: DROP TABLE <tablename> PURGE;
  Example:
  DROP TABLE CUSTOMER PURGE;
• the keyword PURGE
  – If used, the table is permanently deleted.
  – If not used the table is sent to the Recyclebin.
  – May also use PURGE to permanently delete a table previously sent to the Recyclebin. This can follow a DROP statement.
  Example:
  Drop TABLE Old_Table;
  Purge TABLE Old_Table;

Deleting a Table Structure

• Restrictions
  – a table cannot be dropped if it is the “one” side of any relationship, i.e., is being referenced by a foreign key in another table.
  – Using the cascade constraints clause will cause the foreign key constraints in the other table to be deleted, allowing the table to be dropped.
• Syntax:
  DROP TABLE <tablename> CASCADE CONSTRAINTS.
  Example:
  Drop table vendor cascade constraints;
SQL Table Indexes

- **CREATE INDEX command**, allows additional SQL indexes can be created based on any selected attribute or set of attributes – 'a composite index'.
  - When primary key is declared, DBMS automatically creates unique index
- The two main reasons for additional indexes
  1. Used to define a secondary key for speed/efficiency of data retrieval
  2. Used to prevent data duplication-specified as unique

Syntax:  
```
CREATE INDEX <index name>
ON <table name>(<attribute name> ASC|DESC);
```

- The Index may be specified as Unique, otherwise the default is non-unique.
- Standard naming practice puts an X at the end of the index name.
- Examples:
  ```
  create index p_priceXon product (p_price) desc;
  create unique index p_codeXon product(p_code);
  create unique index ratingXon earnedrating (emp_num, rtg_code, earnrtg_date);
  ```

Example Duplicate Data Problem (without an Index)

- Rule - Should only allow the test to be taken on differing days.
- Example Table containing testing records.
  - Example not using an index, thereby allowing duplicate data – i.e., 111 taking same test WEA on 18-feb-2003.

```
<table>
<thead>
<tr>
<th>EXAM NO</th>
<th>TEST NAME</th>
<th>TEST CODE</th>
<th>TEST DATE</th>
<th>TEST SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>130</td>
<td>1</td>
<td>WEA</td>
<td>15-Mar-2005</td>
<td>83</td>
</tr>
<tr>
<td>130</td>
<td>2</td>
<td>WEA</td>
<td>15-Mar-2005</td>
<td>87</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>HRG</td>
<td>14-Dec-2005</td>
<td>91</td>
</tr>
<tr>
<td>111</td>
<td>2</td>
<td>YEA</td>
<td>18-Feb-2006</td>
<td>91</td>
</tr>
<tr>
<td>111</td>
<td>3</td>
<td>WEA</td>
<td>18-Feb-2006</td>
<td>91</td>
</tr>
<tr>
<td>112</td>
<td>1</td>
<td>YREM</td>
<td>17-Aug-2005</td>
<td>87</td>
</tr>
</tbody>
</table>
```

Inserting Rows into a Table

- **INSERT DML Command**: Used to add data to a table
- Syntax:
  ```
  INSERT INTO <table name> [Column,...] VALUES
  (attribute1 value, attribute2 value, ... etc.);
  ```
- Examples:
  ```
  INSERT INTO VENDOR
  VALUES('21225, Bryson, Inc.', 'Smithson', '615', '223-3234', 'TN', 'Y');
  INSERT INTO PRODUCT
  VALUES('11 QER/31', 'Power painter, 15 psi. 3-nozzle', '2-DEC-96', 85, 109.99, 0.00, 25595);
  ```
Inserting Rows into a Table

- There are two methods of inserting rows with only partial columns
  1. By naming the columns.
     \[
     \text{INSERT INTO NEW\_PRODUCT (p\_code, p\_descript) VALUES ('ABC-234', 'New item');}
     \]
  2. Insert rows with missing columns by using 'NULL'.
     \[
     \text{INSERT INTO VENDOR VALUES(21225, 'Bryson, Inc.', null, null, null, 'TN', 'Y');}
     \]
- Attributes without a value must be declared
  - having a Default value or
  - No 'Not Null' constraint.

Review Inserting Rows

- When entering values, notice that:
  - Row contents are entered between parentheses
  - Character and date values are entered between apostrophes
  - Numerical entries are not enclosed in apostrophes
  - Attribute entries are separated by commas
  - A value is required for each column
- Describe the table to make sure unlisted columns
  - have a default value or
  - are without a 'not null' constraint

Deleting Table Rows

- DELETE : Purpose - Deletes a table row
  - Syntax:
    \[
    \text{DELETE FROM tablename [WHERE conditionlist];}
    \]
  - WHERE condition is optional
  - If WHERE condition is not specified, all rows from specified table will be deleted

Examples

\[
\text{DELETE FROM PRODUCT WHERE P\_CODE = '2238/QPD';}
\]
\[
\text{DELETE FROM PRODUCT WHERE P\_MIN = 5;}
\]
\[
\text{DELETE FROM PRODUCT;}
\]

Saving Table Changes

- Changes made to table contents are not physically saved on disk until:
  - Database is closed
  - Program is closed
  - COMMIT command is used
- Syntax:
  - COMMIT [WORK];
- Will permanently save any changes made to the data in all tables in the database
Saving & Restoring the Table

- **TRANSACTIONS**
  - Logical unit of work containing all statements between commits or rollbacks
- **ROLLBACK**;
  - Used to restore the database to its previous condition
  - Only applicable if COMMIT command has not been used to permanently store changes in the database
- **COMMIT**;
  - Must be used to finalize any changes.
  - Oracle automatically commits when exiting normally (SQL+, & other interfaces)

Restoring Table Contents

- COMMIT and ROLLBACK only work with data manipulation commands that are used to add, modify, or delete table rows
- Transaction Example:
  - commit;
  - delete from mytable where salary = 0.00;
  - select * from mytable;
  - rollback;
- Changes to the database with DDL commands (Create, Drop, etc.) are committed with the command.

Updating Table Rows

- **UPDATE** : Purpose - to Modify data in a table
  - Syntax:
    ```sql
    UPDATE tablename
    SET columnname = expression [, columnname = expression]
    WHERE conditionlist;
    ```
  - If more than one attribute is to be updated in the row, separate with commas
  - **UPDATE** command updates only data in existing rows
  - A ROLLBACK command undoes changes made by the UPDATE statements to the previous Commit.

Example Table Rows Updates

- **UPDATE PRODUCT**
  - SET P_SALECODE = '2'
  - WHERE P_CODE = '1546-QQ2';
- **UPDATE PRODUCT**
  - SET P_SALECODE = '1'
  - WHERE P_INDATE BETWEEN '15-NOV-05' AND '1-DEC-06';
  - AND P_CODE IN ('2232/QWE','2232/QTY');
  - WHERE P_CODE = 'SW-23116';
Using a SELECT Subquery to Insert Rows

- INSERT rows with subquery from another table
- Both tables must be compatible, i.e., the same attributes and attribute type.
  - Uses SELECT subquery
    - **Subquery**: query embedded (or nested or inner) inside another query
    - The subquery is executed first, producing rows to be inserted.
  - Syntax:
    - `INSERT INTO tablename SELECT * FROM tablename;`

Example: Inserting Table Rows with a SELECT Subquery

```
INSERT INTO part SELECT * FROM product;
```

Copying & Renaming a Table

- Copying a table requires:
  - A new table structure to be made
  - The rows from the old table must be inserted.
- Use a CREATE statement with a nested select (subquery) to accomplish that.
  - `CREATE TABLE newjob AS SELECT * FROM job;`
- May select to copy only certain rows by including a WHERE clause on the subquery.
  - `CREATE Table SpringSchedule as select * from AnnualSchedule where term = 'SP11';`
- Renaming a table
  - `RENAME oldtablename TO newtablename;`

Review - The user’s data dictionary

- Know your environment before and after you create tables.
  - Displays tables for current logon
    - `SELECT * FROM tab;`
    - `SELECT table_name FROM user_tables;`
  - Displays all objects for current logon
    - `SELECT object_name, object_type FROM user_objects;`
- Looking at a table’s data dictionary
  - `sqlplus> describe <tablename>`
SQL LABS

- Check out Scripting notes on Website resource page.
  - Include all Creates, Alters, Inserts
  - Describe the tables after any structural change
    - Commands: Create or Alter
  - Display data both before and after changes
    - Commands: Insert, Delete, Update

Summary

- Creating Indexes: secondary keys for a table.
- DML commands allow you to
  - add, modify, and delete rows from tables
  - Create Database transactions: A complete segment of work
- The basic DML commands:
  - SELECT, INSERT, UPDATE, DELETE, COMMIT, and ROLLBACK
- Nested querying used with and Insert or a Create will allow copying rows into another table, or making a complete copy of another table.
- The DDL command RENAME allows changing a table name.