Alicia Bonavita

Database Programming: Sections 12, 13

12-1 INSERT Statements

- Vocabulary
  - Someone doing “real Work” with the computer, using it as a means rather than an end
    - **End User**
  - Consists of a collection of DML statements that form a logical unit of work
    - **Transaction**
  - Fully and clearly expressed; leaving nothing implied
    - **Explicit**
  - Adds a new row to a table
    - **Insert**
- Try and solve
  - 1.Give two examples of why it is important to be able to alter the data in a database.
    - **It is important to be able to alter the data in a database because sometimes information may need to be updated or changed for any reason. We would need the ability to alter the data in order to keep normalization within the database.**
  - 2.DJs on Demand just purchased four new CDs. Use an explicit INSERT statement to add each CD to the copy_d_cds table. After completing the entries, execute a SELECT * statement to verify

```
INSERT INTO copy_d_cds (cd_id, title, artist, genre, release_date)
VALUES (101, 'Greatest Hits', 'Artist A', 'Pop', '2024-10-01');

INSERT INTO copy_d_cds (cd_id, title, artist, genre, release_date)
VALUES (102, 'Classic Rock', 'Artist B', 'Rock', '2024-10-02');

INSERT INTO copy_d_cds (cd_id, title, artist, genre, release_date)
VALUES (103, 'Jazz Essentials', 'Artist C', 'Jazz', '2024-10-03');

INSERT INTO copy_d_cds (cd_id, title, artist, genre, release_date)
VALUES (104, 'Country Hits', 'Artist D', 'Country', '2024-10-04');

-- Verify the insertions:
SELECT * FROM copy_d_cds;
```

- 3.DJs on Demand has two new events coming up. One event is a fall football party and the other event is a sixties theme party. The DJs on Demand clients requested the songs shown in the table for their events. Add these songs to the copy_d_songs table using an implicit INSERT statement.

  INSERT INTO copy_d_songs
  SELECT song_id, title, artist, genre
  FROM song_requests
  WHERE event_name IN ('Fall Football Party', 'Sixties Theme Party');

- 4.Add the two new clients to the copy_d_clients table. Use either an implicit or an explicit INSERT.

  INSERT INTO copy_d_clients (client_id, client_name, contact_info)
  VALUES (201, 'Client A', 'clienta@example.com');

  INSERT INTO copy_d_clients (client_id, client_name, contact_info)
  VALUES (202, 'Client B', 'clientb@example.com');

- 5. Add the new client's events to the copy_d_events table. The cost of each event has not been determined at this date.

  INSERT INTO copy_d_events (event_id, client_id, event_name, event_date, cost)
  VALUES (301, 201, 'Fall Football Party', '2024-11-10', NULL);

  INSERT INTO copy_d_events (event_id, client_id, event_name, event_date, cost)
  VALUES (302, 202, 'Sixties Theme Party', '2024-11-15', NULL);

- 6.Create a table called rep_email using the following statement:

  - CREATE TABLE rep_email (id NUMBER(3) CONSTRAINT rel_id_pk PRIMARY KEY,first_name VARCHAR2(10),last_name VARCHAR2(10),
    email_address VARCHAR2(10))

    CREATE TABLE rep_email (
    id NUMBER(3) CONSTRAINT rep_id_pk PRIMARY KEY,
    first_name VARCHAR2(10),
    last_name VARCHAR2(10),
    email_address VARCHAR2(30)

);

Populate this table by running a query on the employees table that includes only those employees who are REP’s

```
INSERT INTO rep_email (id, first_name, last_name, email_address)
SELECT employee_id, first_name, last_name, email
FROM employees
WHERE job_title = 'REP';
```

12-2 Updating Column Values and Deleting Rows

- Vocabulary
  - Modifies existing rows in a table
    - **Update**
  - Retrieves information from one table and uses the information update another table
    - **Merge**
  - Ensures that the data adheres to a predefined set of rules
    - **Constraint**
  - Deletes information on a linked table based on what was deleted on the other table
    - **On delete cascade**
  - Removes existing rows from a table
    - **delete**
- Try it / Solve it
  - 1.Monique Tuttle, the manager of Global Fast Foods, sent a memo requesting an immediate change in prices. The price for a strawberry shake will be raised from $3.59 to $3.75, and the price for fries will increase to $1.20. Make these changes to the copy_f_food_items table.

    ```
    UPDATE copy_f_food_items
    SET price = 3.75
    WHERE item_name = 'Strawberry Shake';

    UPDATE copy_f_food_items
    SET price = 1.20
    WHERE item_name = 'Fries';
    ```

  - 2. Bob Miller and Sue Doe have been outstanding employees at Global Fast Foods. Management has decided to reward them by increasing their overtime pay. Bob Miller will receive an additional $0.75 per hour and Sue Doe will receive an additional $0.85 per hour. Update the copy_f_staffs table to show these new values. (Note: Bob Miller currently doesn’t get overtime pay. What function do you need to use to convert a null value to 0?)

UPDATE copy_f_staffs
SET overtime_pay = NVL(overtime_pay, 0) + 0.75
WHERE staff_name = 'Bob Miller';

UPDATE copy_f_staffs
SET overtime_pay = overtime_pay + 0.85
WHERE staff_name = 'Sue Doe';

- 3.Add the orders shown to the Global Fast Foods copy_f_orders table:

  INSERT INTO copy_f_orders (order_id, customer_id, order_date, amount)
  VALUES (101, 1, '2024-11-10', 25.50);

- 4. Add the new customers shown below to the copy_f_customers table. You may already have added Katie Hernandez. Will you be able to add all these records successfully?

  INSERT INTO copy_f_customers (customer_id, customer_name, contact_info)
  SELECT 201, 'John Smith', '123-456-7890' FROM DUAL
  WHERE NOT EXISTS (SELECT 1 FROM copy_f_customers
  WHERE customer_name = 'John Smith');

- 5.Sue Doe has been an outstanding Global Foods staff member and has been given a salary raise. She will now be paid the same as Bob Miller. Update her record in copy_f_staffs.

  UPDATE copy_f_staffs
  SET salary = (SELECT salary FROM copy_f_staffs WHERE staff_name = 'Bob Miller')
  WHERE staff_name = 'Sue Doe';

- 6. Global Fast Foods is expanding their staff. The manager, Monique Tuttle, has hired Kai Kim. Not all information is available at this time, but add the information shown here.

  INSERT INTO copy_f_staffs (staff_id, staff_name, position, salary)
  VALUES (301, 'Kai Kim', 'New Hire', 0);

- 7. Now that all the information is available for Kai Kim, update his Global Fast Foods record to include the following: Kai will have the same manager as Sue Doe. He does not qualify for overtime. Leave the values for training, manager budget, and manager target as null.

  ```
  UPDATE copy_f_staffs
  SET manager_id = (SELECT manager_id FROM copy_f_staffs
  WHERE staff_name = 'Sue Doe'),
     overtime_qualify = 'No'
  WHERE staff_name = 'Kai Kim';
  ```

- 8.Execute the following SQL statement. Record your results.

  ```
  DELETE from departments WHERE department_id = 60;
  SELECT * FROM departments WHERE department_id = 60;
  ```

- 9.Kim Kai has decided to go back to college and does not have the time to work and go to school.Delete him from the Global Fast Foods staff. Verify that the change was made.

  ```
  DELETE FROM copy_f_staffs WHERE staff_name = 'Kim Kai';

  -- Verify deletion:
  SELECT * FROM copy_f_staffs WHERE staff_name = 'Kim Kai';
  ```

- 10. Create a copy of the employees table and call it lesson7_emp;Once this table exists, write a correlated delete statement that will delete any employees from the lesson7_employees table that also exist in the job_history table.

  ```
  DELETE FROM lesson7_emp
  WHERE EXISTS (
     SELECT 1 FROM job_history
     WHERE job_history.employee_id = lesson7_emp.employee_id
  );
  ```

12-3 DEFAULT Values, MERGE, and Multi-Table Inserts

- Try it/ Solve it
  - 1. When would you want a DEFAULT value?

    Default value should be used when you are trying to achieve column starting consistency without wanting to manually enter a value each time

  - 2.Currently, the Global Foods F_PROMOTIONAL_MENUS table START_DATE column does not have SYSDATE set as DEFAULT. Your manager has decided she would like to be able to set the starting date of promotions to the current day for some entries. This will require three steps:

- a.In your schema, Make a copy of the Global Foods F_PROMOTIONAL_MENUS table using the following SQL statement:

  CREATE TABLE copy_f_promotional_menus
  *AS (SELECT * FROM f_promotional_menus)*

  CREATE TABLE copy_f_promotional_menus
  AS (SELECT * FROM f_promotional_menus);

- b.Alter the current START_DATE column attributes using:
  ALTER TABLE copy_f_promotional_menus MODIFY(start_date DATE DEFAULT SYSDATE)

  ALTER TABLE copy_f_promotional_menus
  MODIFY (start_date DATE DEFAULT SYSDATE);

- c.INSERT the new information and check to verify the results.
  INSERT a new row into the copy_f_promotional_menus table for the manager's new promotion. The promotion code is 120. The name of the promotion is 'New Customer.' Enter DEFAULT for the start date and '01-Jun-2005' for the ending date. The giveaway is a 10% discount coupon. What was the correct syntax used?

  INSERT INTO copy_f_promotional_menus (promotion_code, promotion_name, start_date, end_date, giveaway)
  VALUES (120, 'New Customer', DEFAULT, TO_DATE('01-Jun-2005', 'DD-Mon-YYYY'), '10% discount coupon');

  SELECT * FROM copy_f_promotional_menus WHERE promotion_code = 120;

○ 3.Allison Plumb, the event planning manager for DJs on Demand, has just given you the following list of CDs she acquired from a company going out of business. She wants a new updated list of CDs in inventory in an hour, but she doesn't want the original D_CDS table changed. Prepare an updated inventory list just for her.

- A. Assign new cd_numbers to each new CD acquired.
- B. Create a copy of the D_CDS table called manager_copy_d_cds. What was the correct syntax used?

- C. INSERT into the manager_copy_d_cds table each new CD title using an INSERT statement.
  - Make up one example or use this data:
    20, 'Hello World Here I Am', 'Middle Earth Records', '1998' What was the correct syntax used?

- D. Use a merge statement to add to the manager_copy_d_cds table, the CDs from the original table. If there is a match, update the title and year. If not, insert the data from the original table.What was the correct syntax used?

```
MERGE INTO manager_copy_d_cds AS target
USING D_CDS AS source
ON target.cd_number = source.cd_number
WHEN MATCHED THEN
   UPDATE SET target.title = source.title,
           target.year = source.year
WHEN NOT MATCHED THEN
   INSERT (cd_number, title, label, year)
   VALUES (source.cd_number, source.title, source.label,
source.year);
```

- 4.Run the following 3 statements to create 3 new tables for use in a Multi-table insert statement. All 3 tables should be empty on creation, hence the WHERE 1=2 condition in the WHERE clause.

```
CREATE TABLE sal_history (employee_id, hire_date, salary)
AS SELECT employee_id, hire_date, salary
FROM employees
WHERE 1=2;
```

```
CREATE TABLE sal_history (employee_id, hire_date, salary)
AS SELECT employee_id, hire_date, salary
FROM employees
WHERE 1=2;
```

```
CREATE TABLE mgr_history (employee_id, manager_id, salary)
AS SELECT employee_id, manager_id, salary
FROM employees
WHERE 1=2;
```

```
CREATE TABLE special_sal (employee_id, salary)
AS SELECT employee_id, salary
FROM employees
WHERE 1=2;
```

```
CREATE TABLE special_sal (employee_id, salary)
AS SELECT employee_id, salary
FROM employees
WHERE 1=2;
```

CREATE TABLE special_sal (employee_id, salary)
AS SELECT employee_id, salary
FROM employees
WHERE 1=2;

13-1 Creating Tables

- Vocabulary
  - Create and maintained by the oracle server and contains information about the database
    - **Data Dictionary**
  - A collection of objects that are the logical structures that directly refer to the data in the database
    - **Schema**
  - Specifies a preset value if a value is omitted in the INSERT statement
    - **Default Value**
  - Stores data ; basic unit of storage composed of rows and columns
    - **Table**
  - Command use to make a new table
    - **CREATE TABLE**
- Try It/ Solve it
  - 1. Complete the following table
    -

Column Name	Student ID	Last_name	First_name	credits	graduation_date
Key type	PK				
Nulls/Unique	Not Null	Not Null	Not Null	Nullable	Nullable
FK column				FK	
Datatype	Number	VARCHAR2	VARCHAR2	Number	Date

Length	6	30	30	3	

- 2. Write the syntax to create the grad_candidates table.

```
CREATE TABLE grad_candidates (
   student_id NUMBER(6) PRIMARY KEY,
   last_name VARCHAR2(30) NOT NULL,
   first_name VARCHAR2(30) NOT NULL,
   credits NUMBER,
   graduation_date DATE,
   FOREIGN KEY (credits) REFERENCES Credits(credits_id) --
Replace 'Credits' and 'credits_id' if needed
);
```

- 3.Confirm creation of the table using DESCRIBE.

```
DESCRIBE grad_candidates;
```

- 4.Create a new table using a subquery. Name the new table your last name -- e.g., smith_table. Using a subquery, copy grad_candidates into smith_table.

```
CREATE TABLE smith_table AS
SELECT *
FROM grad_candidates;
```

- 5.Insert your personal data into the table created in question 4.

```
INSERT INTO smith_table (student_id, last_name, first_name,
credits, graduation_date)
VALUES (123456, 'Smith', 'John', 120, TO_DATE('2024-05-15',
'YYYY-MM-DD'));
```

- 6.Query the data dictionary for each of the following:
  - A. USER_TABLES
    SELECT * FROM USER_TABLES;
  - B. USER_OBJECTS
    SELECT * FROM USER_OBJECTS;
  - C. USER_CATALOG or USER_CAT
    SELECT * FROM USER_CATALOG;

- In separate sentences, summarize what each query will return
  a. **This query will return information about the current user to which the table is owned by**

b. **This query will return information about database objects that are also owned by the current user such details like the view, table, sequence and other objects related to the table.**
c. **This query will return a catalog view that can be used to view all object that the current table owner/user has in order to review various data objects , types, and names**

13-2 Using Data Types

- Vocabulary
  - Allows time to be stored as an interval of years and months
    - **Interval Year to Month**
  - When a column is selected in a SQL statement the time is automatically converted to the user's timezone
    - **Timestamp with local time zone**
  - Binary large object data up to 4 gigabyte
    - **BLOB**
  - Stores a timezone value as a displacement for universal coordinated Time or UCT
    - **Timestamp with local time zone**
  - Allows time to be stored as an interval or days to hours, minutes. And seconds
    - **Interval Day to Seconds**
  - Character data up to 4 gigabytes
    - **CLOB**
  - Allows the time to be stored as a date with fractional seconds
    - **Timestamp**
- Try it / Solve it
  - 1.Create tables using each of the listed time-zone data types, use your time-zone and one other in your examples. Answers will vary.
    - a.TIMESTAMP WITH LOCAL TIME ZONE

```
CREATE TABLE local_time_event (
   event_id NUMBER PRIMARY KEY,
   event_timestamp TIMESTAMP WITH LOCAL TIME ZONE
);

INSERT INTO local_time_event (event_id, event_timestamp)
VALUES (1, TIMESTAMP '2024-11-12 08:30:00
America/New_York');

INSERT INTO local_time_event (event_id, event_timestamp)
VALUES (2, TIMESTAMP '2024-11-12 15:30:00 Europe/London');
```

- b.INTERVAL YEAR TO MONTH

  ```
  CREATE TABLE project_duration (
     project_id NUMBER PRIMARY KEY,
     duration INTERVAL YEAR TO MONTH
  );

  INSERT INTO project_duration (project_id, duration)
  VALUES (1, INTERVAL '1-6' YEAR TO MONTH);  -- 1 year and 6
  months

  INSERT INTO project_duration (project_id, duration)
  VALUES (2, INTERVAL '2-3' YEAR TO MONTH);  -- 2 years and 3
  months
  ```

- c.INTERVAL DAY TO SECOND

  ```
  CREATE TABLE task_duration (
     task_id NUMBER PRIMARY KEY,
     duration INTERVAL DAY TO SECOND
  );

  INSERT INTO task_duration (task_id, duration)
  VALUES (1, INTERVAL '5 12:30:45' DAY TO SECOND);  -- 5 days,
  12 hours, 30 minutes, and 45 seconds

  INSERT INTO task_duration (task_id, duration)
  VALUES (2, INTERVAL '3 08:15:00' DAY TO SECOND);  -- 3 days,
  8 hours, and 15 minutes
  ```

- 2.Execute a SELECT * from each table to verify your input.

  ```
  SELECT * FROM local_time_event;

  SELECT * FROM project_duration;

  SELECT * FROM task_duration;
  ```

- 3.Give 3 examples of organizations and personal situations where it is important to know to which time zone a date-time value refer

  1. **Airlines**

        a. **Its importance that airlines have a good and accurate data with an exact schedule in order to ensure that flights departures and arrivals to not get mixed up when communicating with pilots**
    2. **Financial/Banking**
        a. **Financial analytics for money related data such as stocks and bones often requires accurate time zone information.**
        So that traders or people analyzing the stocks and market have a good understanding of transactional time stamping of when these stocks are open or closed, or if there is a price change. Which is crucial when making financial decisions.
    3. **Technology Companies**
        a. **Companies like Google, Apple, and more require coordination amongst several regions when trying to maintain technology, or to meet expected deadlines for a product or service.**
        Therefore tech companies need reliable and accurate time zone data to ensure that work across different countries, areas, or regions run together effectively and efficiently. To ensure that updates and project correspondence are kept up.

13-3 Modifying a Table

- Try it / Solve it
    - 1. Why is it important to be able to modify a table?
      IT is important to be able to modify a table because your requirements for The data change over time. You'll want to be able to adapt the table to meet those new requirements or expectations when adding or updating any type of data into the table.

    - 2. CREATE a table called Artists.

```
CREATE TABLE Artists (
  artist_id NUMBER,
  first_name VARCHAR2(50),
  last_name VARCHAR2(50),
  band_name VARCHAR2(100),
  email VARCHAR2(100),
  hourly_rate NUMBER
);
```

        - A. Add the following to the table:

```
INSERT INTO Artists (artist_id, first_name, last_name, band_name,
email, hourly_rate)
SELECT artist_id, first_name, last_name, band_name, email,
hourly_rate FROM d_songs WHERE artist_id = <artist_id>;
```

    - B. INSERT one artist from the d_songs table.
      INSERT INTO Artists (artist_id, first_name, last_name, band_name, email, hourly_rate)
      SELECT artist_id, first_name, last_name, band_name, email, hourly_rate FROM d_songs WHERE artist_id = <artist_id>;

    - C. INSERT one artist of your own choosing.
      INSERT INTO Artists (artist_id, first_name, last_name, band_name, email, hourly_rate)
      VALUES (2, 'John', 'Doe', 'The Explorers', 'jdoe@example.com', 45);

- d.Give an example how each of the following may be used on the table that you have created:

    - 1) ALTER TABLE
      ALTER TABLE Artists ADD (genre VARCHAR2(50));
    - 2) DROP TABLE
      DROP TABLE Artists;
    - 3) RENAME TABLE
      ALTER TABLE Artists RENAME TO Musicians;
    - 4) TRUNCATE
      TRUNCATE TABLE Artists;
    - 5) COMMENT ON TABLE
      COMMENT ON TABLE Artists IS 'This table contains artist information.';

- 3.In your o_employees table, enter a new column called "Termination." The datatype for the new column should be VARCHAR2. Set the DEFAULT for this column as SYSDATE to appear as character data in the format: February 20th, 2003.
  ALTER TABLE o_employees ADD (Termination VARCHAR2(30) DEFAULT TO_CHAR(SYSDATE, 'Month DDth, YYYY'));

- 4. Create a new column in the o_employees table called start_date. Use the TIMESTAMP WITH LOCAL TIME ZONE as the datatype.
  ALTER TABLE o_employees ADD (start_date TIMESTAMP WITH LOCAL TIME ZONE);

- 5.Truncate the o_jobs table. Then do a SELECT * statement. Are the columns still there? Is the data still there?
  TRUNCATE TABLE o_jobs;
  SELECT * FROM o_jobs;
  Yes there are still tables and there still is data with the following query.

- 6.What is the distinction between TRUNCATE, DELETE, and DROP for tables?
    - **TRUNCATE - removes all rows without logging individuals rows or resetting**
    - **DELETE - removes specific rows on conditions can be reverted back**
    - **DROP - Is a COMPLETE deletion with no rollback availability and removes the tables and its entire structure**

- 7.List the changes that can and cannot be made to a column.
  What CAN be made - renaming, adding values, modifying data types
  What CANNOT be made - reduce length, changing the data type , or changing column constraints

- 8.Add the following comment to the o_jobs table: "New job description added"View the data dictionary to view your comments.
  COMMENT ON TABLE o_jobs IS 'New job description added';

- 9.Rename the o_jobs table to o_job_description.
  ALTER TABLE o_jobs RENAME TO o_job_description;

- 10. F_staffs table exercises:
    - A. Create a copy of the f_staffs table called copy_f_staffs and use this copy table for the remaining labs in this lesson.
      CREATE TABLE copy_f_staffs AS SELECT * FROM f_staffs;

    - B. Describe the new table to make sure it exists.
      DROP TABLE copy_f_staffs;

    - C. Drop the table.
      DROP TABLE copy_f_staffs;
    - D. Try to select from the table.
      SELECT * FROM copy_f_staffs;

    - e.Investigate your recycle bin to see where the table went.
      SHOW RECYCLEBIN;
    - F. Try to select from the dropped table by using the value stored in the OBJECT_NAME column. You will need to copy and paste the name as it is exactly, and enclose the new name in " "(double quotes). So if the dropped name returned to you is BIN$Q+x1nJdcUnngQESYELVIdQ==$0, you need to write a query that refers to "BIN$Q+x1nJdcUnngQESYELVIdQ==$0".
      SELECT * FROM "BIN$Q+x1nJdcUnngQESYELVIdQ==$0";

    - G. Undrop the table.
      FLASHBACK TABLE copy_f_staffs TO BEFORE DROP;

- 11. Still working with the copy_f_staffs table, perform an update on the table.

- A. Issue a select statement to see all rows and all columns from the copy_f_staffs table;
  UPDATE copy_f_staffs SET salary = 12 WHERE first_name = 'Sue' AND last_name = 'Doe';
  COMMIT;

- B. Change the salary for Sue Doe to 12 and commit the change.
  SELECT * FROM copy_f_staffs VERSIONS BETWEEN TIMESTAMP MINVALUE AND MAXVALUE;

- C. Issue a select statement to see all rows and all columns from the copy_f_staffs table;
  UPDATE copy_f_staffs SET salary = <original_salary> WHERE first_name = 'Sue' AND last_name = 'Doe';
  COMMIT;

- d.For Sue Doe, update the salary to 2 and commit the change.
  UPDATE copy_f_staffs
  SET salary = 2
  WHERE first_name = 'Sue' AND last_name = 'Doe';
  COMMIT;

- e.Issue a select statement to see all rows and all columns from the **copy_f_staffs table;**
  SELECT * FROM copy_f_staffs;

- f.Now, issue a FLASHBACK QUERY statement against the copy_f_staffs table, so you can see all the changes made.
  SELECT *
  FROM copy_f_staffs
  VERSIONS BETWEEN TIMESTAMP MINVALUE AND MAXVALUE;

- g.Investigate the result of f), and find the original salary and update the copy_f_staffs table salary column for Sue Doe back to her original salary
  UPDATE copy_f_staffs
  SET salary = 10
  WHERE first_name = 'Sue' AND last_name = 'Doe';
  COMMIT;