



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

SECD2523 – Database

Lab 1: DDL

SECTION: 08-SECJH

LECTURER NAME: DR NOOR HIDAYAH BINTI ZAKARIA

SUBMISSION DATE: 12 DECEMBER 2023

| NO. | NAME | MATRIC NUMBER |
|-----|-------------|---------------|
| 1 | LEW SHU BEI | A22EC8002 |

Section 6 Lesson 4 Exercise 1: Data Manipulation Language

Use DML operations to manage database tables (S6L4 Objective 2)

In this exercise you will populate and work with the data that is stored in the database system tables.

Part 1 : Running a script to populate the tables.

You have to consider the order of the tables when populating them. A table that has a foreign key field cannot be populated before the related table with the primary key.

1. Use the table mapping document and list the order that you would use to populate the tables.
 - a) inventory_list
 - b) items
 - c) price_history
 - d) sales_representatives e) sales_rep_addresses f) teams
 - g) customers
 - h) customers_addresses i) orders
 - j) ordered_items

```
1 CREATE TABLE inventory_list (  
2     id VARCHAR2(11) NOT NULL,  
3     cost NUMBER(7,2) NOT NULL,  
4     units NUMBER(4) NOT NULL,  
5     CONSTRAINT inventory_list_pk PRIMARY KEY (id)  
6 )
```

Table created.

```
1 CREATE TABLE items (  
2     itm_number VARCHAR2(10) NOT NULL,  
3     name VARCHAR2 (20) NOT NULL,  
4     description VARCHAR2(50) NOT NULL,  
5     category VARCHAR2 (25) NOT NULL,  
6     color VARCHAR2 (15),  
7     "Size" CHAR (1),  
8     ilt_id VARCHAR2 (11) NOT NULL,  
9     CONSTRAINT item_pk PRIMARY KEY (itm_number )  
10 )
```

Table created.

```
1 CREATE TABLE price_history (  
2     start_date DATE NOT NULL,  
3     start_time DATE NOT NULL,  
4     price NUMBER(7,2) NOT NULL,  
5     end_date DATE,  
6     end_time DATE,  
7     itm_number VARCHAR2(10) NOT NULL,  
8     CONSTRAINT price_history_pk PRIMARY KEY ( itm_number, start_date, start_time ),  
9     CONSTRAINT price_history_items_fk FOREIGN KEY (itm_number ) REFERENCES items (itm_number )  
10 )
```

Table created.

```

1 CREATE TABLE sales_representatives (
2     id VARCHAR2(4) NOT NULL,
3     email VARCHAR2(50) NOT NULL,
4     first_name VARCHAR2(20) NOT NULL,
5     last_name VARCHAR2(30) NOT NULL,
6     phone_number VARCHAR2(11) NOT NULL,
7     commission_rate NUMBER(2) NOT NULL,
8     supervisor_id VARCHAR2(4) NOT NULL,
9     CONSTRAINT sales_representative_pk PRIMARY KEY ( id ),
10    CONSTRAINT sre_email_uk UNIQUE (email)
11 )

```

Table created.

```

1 CREATE TABLE sales_rep_addresses (
2     id VARCHAR2(4) NOT NULL,
3     address_line_1 VARCHAR2(30) NOT NULL,
4     address_line_2 VARCHAR2(30),
5     city VARCHAR2(15) NOT NULL,
6     zip_code VARCHAR2(7) NOT NULL,
7     CONSTRAINT sales_rep_address_pk PRIMARY KEY ( id )
8 )

```

Table created.

```

1 CREATE TABLE teams (
2     id VARCHAR2(4) NOT NULL,
3     name VARCHAR2 (20) NOT NULL,
4     number_of_players NUMBER(2) NOT NULL,
5     discount NUMBER (2),
6     CONSTRAINT team_pk PRIMARY KEY ( id )
7 )

```

Table created.

```

1 CREATE TABLE customers (
2     ctr_number VARCHAR2(6) NOT NULL,
3     email VARCHAR2 (50) NOT NULL,
4     first_name VARCHAR2 (20) NOT NULL,
5     last_name VARCHAR2(30) NOT NULL,
6     phone_number VARCHAR2(11) NOT NULL,
7     current_balance NUMBER (6,2) NOT NULL,
8     sre_id VARCHAR2 (4),
9     tem_id VARCHAR2(4),
10    loyalty_card_number VARCHAR2(6),
11    CONSTRAINT customer_pk PRIMARY KEY ( ctr_number ), CONSTRAINT ctr_email_uk UNIQUE (email),
12    CONSTRAINT ctr_1cn_uk UNIQUE (loyalty_card_number)
13 )

```

Table created.

```

1 ✓ CREATE TABLE customers_addresses (
2     id VARCHAR2(8) NOT NULL,
3     address_line_1 VARCHAR2(30) NOT NULL,
4     address_line_2 VARCHAR2 (30),
5     city VARCHAR2(15) NOT NULL,
6     zip_code VARCHAR2(7) NOT NULL,
7     ctr_number VARCHAR2(6) NOT NULL,
8     CONSTRAINT customer_address_pk PRIMARY KEY ( id )
9 )

```

Table created.

```

1 ✓ CREATE TABLE orders (
2     id VARCHAR2 (9) NOT NULL,
3     odr_date DATE NOT NULL,
4     odr_time DATE NOT NULL,
5     number_of_units NUMBER(2) NOT NULL,
6     ctr_number VARCHAR2(6) NOT NULL,
7     CONSTRAINT orders_pk PRIMARY KEY ( id )
8 )

```

Table created.

```

1 ✓ CREATE TABLE ordered_items (
2     quantity_ordered NUMBER(3) NOT NULL,
3     quantity_shipped NUMBER(3) NOT NULL,
4     itm_number VARCHAR2(10) NOT NULL,
5     odr_id VARCHAR2(9) NOT NULL,
6     CONSTRAINT ordered_item_pk PRIMARY KEY ( itm_number,odr_id )
7 )

```

Table created.

```

1 ✓ ALTER TABLE customers_addresses ADD CONSTRAINT customer_address_customer_fk FOREIGN KEY ( ctr_number )
2 REFERENCES customers ( ctr_number )

```

Table altered.

```

1 ✓ ALTER TABLE customers ADD CONSTRAINT customer_sales_rep_fk FOREIGN KEY ( sre_id )
2 REFERENCES sales_representatives ( id )

```

Table altered.

```

1 ✓ ALTER TABLE customers ADD CONSTRAINT customer_team_fk FOREIGN KEY ( tem_id )
2 REFERENCES teams ( id )

```

Table altered.

```
1 ALTER TABLE items ADD CONSTRAINT item_inventory_list_fk FOREIGN KEY ( ilt_id )
2 REFERENCES inventory_list ( id )
```

Table altered.

```
1 ALTER TABLE orders ADD CONSTRAINT order_customer_fk FOREIGN KEY ( ctr_number )
2 REFERENCES customers ( ctr_number )
```

Table altered.

```
1 ALTER TABLE ordered_items ADD CONSTRAINT ordered_item_item_fk FOREIGN KEY ( itm_number )
2 REFERENCES items ( itm_number )
```

Table altered.

```
1 ALTER TABLE ordered_items ADD CONSTRAINT ordered_item_order_fk FOREIGN KEY ( odr_id )
2 REFERENCES orders ( id )
```

Table altered.

```
1 ALTER TABLE sales_rep_addresses ADD CONSTRAINT sales_rep_add_sales_rep_fk FOREIGN KEY ( id )
2 REFERENCES sales_representatives ( id )
```

Table altered.

```
1 ALTER TABLE sales_representatives ADD CONSTRAINT sales_rep_sales_rep_fk FOREIGN KEY ( supervisor_id ) REFERENCES sales_representatives ( id )
```

Table altered.

```
1 CREATE OR REPLACE TRIGGER fkntm_orders BEFORE
2   UPDATE OF ctr_number ON orders
3   BEGIN
4     raise_application_error(
5       -20225,
6       'Non Transferable FK constraint on table orders is violated'
7     );
8   END;
```

Trigger created.

- Open the “sports data.sql” and look at the order the data is being added there, does your list match? This file can be found in the Section 6 Lesson 4 interaction (sports data.zip) and must first be extracted.

Answer: Yes, it matched.

- Run the “sports data.sql” script in APEX to populate your tables.
- Check that no errors occurred when you ran the script.

| Script: sports data.sql Status: Complete | | View: Detail Summary Rows: 15 Go | |
|--|---------|--|--------------------|
| Number ↑ | Elapsed | Statement | Feedback |
| 46 | 0.00 | INSERT INTO ordered_items (quantity_ordered, quantity_shippe | 1 row(s) inserted. |
| 47 | 0.00 | INSERT INTO ordered_items (quantity_ordered, quantity_shippe | 1 row(s) inserted. |
| Download | | | |
| 47 | | 47 | 0 |
| Statements Processed | | Successful | With Errors |

Part 2- Inserting rows to the system

- Add a new team to the system

| id | name | Number_of_players | discount |
|------|------|-------------------|----------|
| t004 | Jets | 10 | 5 |

Answer: INSERT INTO teams(id, name, Number_of_players, discount) VALUES ('t004', 'Jets', '10', '5')

| TEAMS | | | | |
|--------------|------------|-------------------|-------------|----------------|
| Columns | Data | Indexes | Constraints | Grants |
| Statistics | Triggers | Dependencies | DDL | Sample Queries |
| + Insert Row | Columns... | Filter... | Count Rows | Load Data |
| Download | Refresh | | | |
| ID | NAME | NUMBER_OF_PLAYERS | DISCOUNT | |
| t004 | Jets | 10 | 5 | |
| t001 | Rockets | 25 | 10 | |
| t002 | Celtics | 42 | 20 | |
| t003 | Rovers | 8 | | |

- Add a new Customer with the following details to the system

| ctr number | email | First name | Last name | Phone number | Current balance | Loyalty card number | tem id | sre id |
|------------|----------------------|------------|-----------|--------------|-----------------|---------------------|--------|--------|
| c02001 | brianrog@hootech.com | Brian | Rogers | 01654564898 | -5 | lc4587 | | |

Answer: INSERT INTO customers

(ctr_number,email,first_name,last_name,phone_number,current_balance,loyalty_card_number)

VALUES ('c02001','brianrog@hootech.com','Brian','Rogers','01654564898','-5','lc4587')

| CTR_NUMBER | EMAIL | FIRST_NAME | LAST_NAME | PHONE_NUMBER | CURRENT_BALANCE | SRE_ID | TEM_ID | LOYALTY_CARD_NUMI |
|------------|--------------------|------------|------------|--------------|-----------------|--------|--------|-------------------|
| c02001 | brianrog@hooe... | Brian | Rogers | 01654564898 | -5 | | | lc4587 |
| c00001 | bob.thornberry... | Robert | Thornberry | 01234567898 | 150 | sr01 | t001 | |
| c00012 | Jjones@freemail... | Jennifer | Jones | 01505214598 | 0 | | | lc1015 |
| c00101 | unknown@here.... | John | Doe | 03216547808 | 987.5 | sr01 | t002 | |
| c00103 | MurciaA@global... | Andrew | Murcia | 07715246890 | 85 | | | lc2341 |
| c01986 | margal87@delp... | Maria | Galant | 01442736589 | 125.65 | sr03 | t003 | |

3. This information violates the check constraint that the current balance must not be less than zero. Change the current balance to 50 and rerun the query.

a) Information violates

```

1  ALTER TABLE customers
2  ADD CHECK (CURRENT_BALANCE >= 0)

```

Results Explain Describe Saved SQL History

ORA-02293: cannot validate (WKSP_LEWSHUBEI.) - check constraint violated

b) Change the current balance to 50

```

1  UPDATE customers
2  SET current_balance =50
3  WHERE ctr_number = 'c02001'

```

Results Explain Describe Saved SQL History

1 row(s) updated.

0.04 seconds

c) Data updated

| CTR_NUMBER | EMAIL | FIRST_NAME | LAST_NAME | PHONE_NUMBER | CURRENT_BALANCE | SRE_ID | TEM_ID | LOYALTY_CARD_NUMI |
|------------|--------------------|------------|------------|--------------|-----------------|--------|--------|-------------------|
| c02001 | brianrog@hooe... | Brian | Rogers | 01654564898 | 50 | | | lc4587 |
| c00001 | bob.thornberry... | Robert | Thornberry | 01234567898 | 150 | sr01 | t001 | |
| c00012 | Jjones@freemail... | Jennifer | Jones | 01505214598 | 0 | | | lc1015 |
| c00101 | unknown@here.... | John | Doe | 03216547808 | 987.5 | sr01 | t002 | |
| c00103 | MurciaA@global... | Andrew | Murcia | 07715246890 | 85 | | | lc2341 |
| c01986 | margal87@delp... | Maria | Galant | 01442736589 | 125.65 | sr03 | t003 | |

Section 6 Lesson 4 Exercise 2: Data Manipulation Language

Use DML operations to manage database tables (S6L4 Objective 2)

In this exercise you will populate and work with the data that is stored in the database system.

Part 1- Updating rows to the system

1. Run the following query to view the content of the price_history table:
SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date,
TO_CHAR (end_time, 'HH24:MI')
FROM price_history;

```
1 SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date, TO_CHAR
2 (end_time, 'HH24:MI')
3 FROM price_history;
```

| START_DATE | TO_CHAR(START_TIME,'HH24:MI:SS') | PRICE | END_DATE | TO_CHAR(END_TIME,'HH24:MI') |
|------------|----------------------------------|-------|------------|-----------------------------|
| 06/17/2017 | 09:00:00 | 4.99 | - | - |
| 11/25/2016 | 09:00:00 | 14.99 | 01/25/2017 | 17:00 |
| 01/25/2017 | 17:01:00 | 8.99 | 01/25/2017 | 19:00 |
| 01/26/2017 | 09:00:00 | 15.99 | - | - |
| 02/12/2017 | 12:30:00 | 7.99 | - | - |
| 04/25/2017 | 10:10:10 | 24.99 | - | - |
| 05/31/2017 | 16:35:30 | 149 | - | - |

7 rows returned in 0.04 seconds [Download](#)

2. Obl is going to update the price of the premium bat so you will need to write a query that will close off the current price by adding the system date values to the end_date and end_time fields. To run this query you will need to both match the item number and identify that the end date is null. This ensures that you are updating the latest price.

Answer:

UPDATE price_history

SET end_date = SYSDATE, end_time = SYSDATE

WHERE itm_number = 'im01101045' AND end_date is null;

```
1 UPDATE price_history
2 SET end_date = SYSDATE, end_time = SYSDATE
3 WHERE itm_number = 'im01101045' AND end_date is null;
```

| Results | Explain | Describe | Saved SQL | History |
|---------|---------|----------|-----------|---------|
|---------|---------|----------|-----------|---------|

1 row(s) updated.

0.01 seconds

- Rerun the select statement on the price_history table to ensure that the statement has been executed.

```

1 SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date, TO_CHAR
2 (end_time, 'HH24:MI')
3 FROM price_history;

```

| START_DATE | TO_CHAR(START_TIME,'HH24:MI:SS') | PRICE | END_DATE | TO_CHAR(END_TIME,'HH24:MI') |
|------------|----------------------------------|-------|------------|-----------------------------|
| 06/17/2017 | 09:00:00 | 4.99 | - | - |
| 11/25/2016 | 09:00:00 | 14.99 | 01/25/2017 | 17:00 |
| 01/25/2017 | 17:01:00 | 8.99 | 01/25/2017 | 19:00 |
| 01/26/2017 | 09:00:00 | 15.99 | 12/16/2023 | 11:09 |
| 02/12/2017 | 12:30:00 | 7.99 | - | - |
| 04/25/2017 | 10:10:10 | 24.99 | - | - |
| 05/31/2017 | 16:35:30 | 149 | - | - |

7 rows returned in 0.01 seconds [Download](#)

- Insert a new row that will use the current date and time to set the new price of the premium bat to be 99.99.

Answer:

**INSERT INTO price_history (start_date, start_time, price, itm_number)
VALUES(SYSDATE, SYSDATE, 99.99, 'im01101048');**

```

1 INSERT INTO price_history (start_date, start_time, price, itm_number)
2 VALUES(SYSDATE, SYSDATE, 99.99, 'im01101048');

```

| START_DATE | TO_CHAR(START_TIME,'HH24:MI:SS') | PRICE | END_DATE | TO_CHAR(END_TIME,'HH24:MI') |
|------------|----------------------------------|-------|------------|-----------------------------|
| 06/17/2017 | 09:00:00 | 4.99 | - | - |
| 11/25/2016 | 09:00:00 | 14.99 | 01/25/2017 | 17:00 |
| 01/25/2017 | 17:01:00 | 8.99 | 01/25/2017 | 19:00 |
| 01/26/2017 | 09:00:00 | 15.99 | 12/16/2023 | 11:09 |
| 02/12/2017 | 12:30:00 | 7.99 | - | - |
| 04/25/2017 | 10:10:10 | 24.99 | - | - |
| 05/31/2017 | 16:35:30 | 149 | - | - |
| 12/16/2023 | 11:10:36 | 99.99 | - | - |

1 row(s) inserted.

- Rerun the select statement on the price_history table to ensure that the statement has been executed.

```

1 SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date, TO_CHAR
2 (end_time, 'HH24:MI')
3 FROM price_history;

```

| START_DATE | TO_CHAR(START_TIME,'HH24:MI:SS') | PRICE | END_DATE | TO_CHAR(END_TIME,'HH24:MI') |
|------------|----------------------------------|-------|------------|-----------------------------|
| 06/17/2017 | 09:00:00 | 4.99 | - | - |
| 11/25/2016 | 09:00:00 | 14.99 | 01/25/2017 | 17:00 |
| 01/25/2017 | 17:01:00 | 8.99 | 01/25/2017 | 19:00 |
| 01/26/2017 | 09:00:00 | 15.99 | 12/16/2023 | 11:09 |
| 02/12/2017 | 12:30:00 | 7.99 | - | - |
| 04/25/2017 | 10:10:10 | 24.99 | - | - |
| 05/31/2017 | 16:35:30 | 149 | - | - |
| 12/16/2023 | 11:10:36 | 99.99 | - | - |

8 rows returned in 0.00 seconds [Download](#)

Part 2: Deleting rows from the system

1. Bob Thornberry has contacted Obl to ask that the 83 Barrhill Drive address be removed from the system as he can no longer receive parcels at this address. Write a SQL statement that will remove this address from the system.

```
1 DELETE FROM customers_addresses
2 WHERE id = 'ca0101';
```

| Results | Explain | Describe | Saved SQL | History |
|-------------------|---------|----------|-----------|---------|
| 1 row(s) deleted. | | | | |
| 0.00 seconds | | | | |

2. Run a select statement on the customers_addresses table to ensure that the statement has been executed.

```
1 SELECT * FROM customers_addresses
```

| ID | ADDRESS_LINE_1 | ADDRESS_LINE_2 | CITY | ZIP_CODE | CTR_NUMBER |
|--------|----------------------|----------------|-----------|----------|------------|
| ca0102 | 17 Gartsquare Road | Starford | Liverpool | LP89JHK | c00001 |
| ca0103 | 54 Ropehill Crescent | Georgetown | Star | ST45AGV | c00101 |
| ca0104 | 36 Watercross Lane | - | Jump | JP23YTH | c01986 |
| ca0105 | 63 Acacia Drive | Skins | Liverpool | LP83JHR | c00001 |

4 rows returned in 0.02 seconds [Download](#)