Database Management Report

Southern New Hampshire University

April 20, 2019

Jose Piña

mysql> SHOW SCHEMAS;

+ - - - - - - - - - - - - - - - - - - - - +

| Database |

+ - - - - - - - - - - - - - - - - - - - - +

| information\_schema |

| event |

| flight\_reservation |

| hr |

| messaging |

| mysql |

| performance\_schema |

| scheduling |

| test |

+ - - - - - - - - - - - - - - - - - - - - +

9 rows in set (0.00 sec)

mysql> USE messaging;

Database changed

mysql> SHOW TABLES;

+ - - - - - - - - - - - - - - - - - - - - - +

| Tables\_in\_messaging |

+ - - - - - - - - - - - - - - - - - - - - - +

| contact\_list |

| message |

| person |

+ - - - - - - - - - - - - - - - - - - - - - +

3 rows in set (0.00 sec)

I used the above queries to look at the different schemas, then to use the messaging schema, and then to see all the tables available in the messaging schema.

Task 1: Insert Record to the Person Table

Construct the SQL statement to add yourself to the Person table. Note: You are required to add yourself as a new record in the “person” table. Use your first name and last name for one of the new records that you are inserting.

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- +

| person\_id | first\_name | last\_name |

+ ----------- + ------------ + ----------- +

| 1 | Michael | Phelps |

| 2 | Katie | Ledecky |

| 3 | Usain | Bolt |

| 4 | Allyson | Felix |

| 5 | Kevin | Durant |

| 6 | Diana | Taurasi |

+ ----------- + ------------ + ----------- +

6 rows in set (0.00 sec)

mysql> INSERT INTO person (person\_id, first\_name, last\_name)

-> VALUES (7, "Jose", "Pina");

Query OK, 1 row affected (0.03 sec)

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- +

| person\_id | first\_name | last\_name |

+ ----------- + ------------ + ----------- +

| 1 | Michael | Phelps |

| 2 | Katie | Ledecky |

| 3 | Usain | Bolt |

| 4 | Allyson | Felix |

| 5 | Kevin | Durant |

| 6 | Diana | Taurasi |

| 7 | Jose | Pina |

+ ----------- + ------------ + ----------- +

7 rows in set (0.00 sec)

I used the queries above to first see all the data in the person table, next, I used a INSERT INTO query to add myself to the person table, then I used a SELECT query to confirm the change in the person table.

Task 2: Alter the Person Table

Construct the SQL statement to alter the table named “person”. The columns, column data types, and column notes are provided in the previous section. You need to alter the table to include an additional column of your choice. This column should represent some property of a person. You can choose the data type for the column and any constraints on the column.

mysql> SHOW COLUMNS FROM person;

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

| Field | Type | Null | Key | Default | Extra |

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

| person\_id | int(8) | NO | PRI | NULL | auto\_increment|

| first\_name | varchar(25) | NO | | NULL | |

| last\_name | varchar(25) | NO | | NULL | |

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

3 rows in set (0.00 sec)

mysql> ALTER TABLE person

-> ADD gender VARCHAR(1) NOT NULL;

Query OK, 7 rows affected (0.26 sec)

Records: 7 Duplicates: 0 Warnings: 0

mysql> SHOW COLUMNS FROM person;

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

| Field | Type | Null | Key | Default | Extra |

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

| person\_id | int(8) | NO | PRI | NULL | auto\_increment|

| first\_name | varchar(25) | NO | | NULL | |

| last\_name | varchar(25) | NO | | NULL | |

| gender | varchar(1) | NO | | NULL | |

+ ------------ + ------------- + ------ + ----- + --------- + ---------------- +

4 rows in set (0.00 sec)

In the above queries, first I used a SHOW COLUMNS query to see all the columns, data types, and notes for the person table. Next, I used a ALTER TABLE query to add a new column of my choice to the person table. Then, I used a SHOW COLUMNS query to confirm the change in the person table.

Task 3: Update Records in the Person Table

Construct the SQL statement to update the existing record in the “person” table to use the new column that you created. Update your record (the record with your first and last name) in the Person table by setting some value to your new property.

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- + -------- +

| person\_id | first\_name | last\_name | gender |

+ ----------- + ------------ + ----------- + -------- +

| 1 | Michael | Phelps | |

| 2 | Katie | Ledecky | |

| 3 | Usain | Bolt | |

| 4 | Allyson | Felix | |

| 5 | Kevin | Durant | |

| 6 | Diana | Taurasi | |

| 7 | Jose | Pina | |

+ ----------- + ------------ + ----------- + -------- +

7 rows in set (0.00 sec)

mysql> UPDATE messaging.person

-> SET gender = 'M'

-> WHERE person\_id % 2 = 1;

Query OK, 4 rows affected (0.04 sec)

Rows matched: 4 Changed: 4 Warnings: 0

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- + -------- +

| person\_id | first\_name | last\_name | gender |

+ ----------- + ------------ + ----------- + -------- +

| 1 | Michael | Phelps | M |

| 2 | Katie | Ledecky | |

| 3 | Usain | Bolt | M |

| 4 | Allyson | Felix | |

| 5 | Kevin | Durant | M |

| 6 | Diana | Taurasi | |

| 7 | Jose | Pina | M |

+ ----------- + ------------ + ----------- + -------- +

7 rows in set (0.00 sec)

mysql> UPDATE messaging.person

-> SET gender = 'F'

-> WHERE person\_id % 2 = 0;

Query OK, 3 rows affected (0.03 sec)

Rows matched: 3 Changed: 3 Warnings: 0

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- + -------- +

| person\_id | first\_name | last\_name | gender |

+ ----------- + ------------ + ----------- + -------- +

| 1 | Michael | Phelps | M |

| 2 | Katie | Ledecky | F |

| 3 | Usain | Bolt | M |

| 4 | Allyson | Felix | F |

| 5 | Kevin | Durant | M |

| 6 | Diana | Taurasi | F |

| 7 | Jose | Pina | M |

+ ----------- + ------------ + ----------- + -------- +

7 rows in set (0.00 sec)

In the above query, first, I used a SELECT query to see all the data in the person table. Next, I used a UPDATE query to update the gender column to “M” for all the males. Next, I used a SELECT query to confirm the changes. Next, I used a UPDATE query to update the gender column to “F” for all the females. Then, I used a SELECT query to confirm the changes.

Task 4: Delete Records from Person Table

Construct the SQL statement to delete the record(s) from the “person” table where the first name is “Diana” and the last name is “Taurasi.”

mysql> DELETE FROM person

-> WHERE first\_name = 'Diana' AND last\_name = 'Taurasi';

Query OK, 1 row affected (0.04 sec)

mysql> SELECT \* FROM person;

+ ----------- + ------------ + ----------- + -------- +

| person\_id | first\_name | last\_name | gender |

+ ----------- + ------------ + ----------- + -------- +

| 1 | Michael | Phelps | M |

| 2 | Katie | Ledecky | F |

| 3 | Usain | Bolt | M |

| 4 | Allyson | Felix | F |

| 5 | Kevin | Durant | M |

| 7 | Jose | Pina | M |

+ ----------- + ------------ + ----------- + -------- +

6 rows in set (0.00 sec)

In the queries above, first, I used a DELETE query to delete a record from the person table. Then, I used a SELECT query to confirm the changes in the person table.

Task 5: Alter the Contact List Table

Construct the SQL statement to alter the table named “contact\_list”. The columns, column data types, and column notes are provided in the previous section. You need to alter the table to include an additional column named “favorite” with a data type of “varchar(10)”. This column is not required.

mysql> SHOW COLUMNS FROM contact\_list;

+ --------------- + -------- + ------ + ----- + --------- + ---------------- +

| Field | Type | Null | Key | Default | Extra |

+ --------------- + -------- + ------ + ----- + --------- + ---------------- +

| connection\_id | int(8) | NO | PRI | NULL | auto\_increment|

| person\_id | int(8) | NO | | NULL | |

| contact\_id | int(8) | NO | | NULL | |

+ --------------- + -------- + ------ + ----- + --------- + ---------------- +

3 rows in set (0.01 sec)

mysql> ALTER TABLE messaging.contact\_list

-> ADD favorite VARCHAR(10) DEFAULT NULL;

Query OK, 14 rows affected (0.32 sec)

Records: 14 Duplicates: 0 Warnings: 0

mysql> SHOW COLUMNS FROM contact\_list;

+ --------------- + ------------- + ------ + ----- + --------- + ---------------- +

| Field | Type | Null | Key | Default | Extra |

+ --------------- + ------------- + ------ + ----- + --------- + ---------------- +

| connection\_id | int(8) | NO | PRI | NULL | auto\_increment|

| person\_id | int(8) | NO | | NULL | |

| contact\_id | int(8) | NO | | NULL | |

| favorite | varchar(10)| YES | | NULL | |

+ --------------- + ------------- + ------ + ----- + --------- + ---------------- +

4 rows in set (0.00 sec)

In the queries above, first, I used a SHOW COLUMNS query to see all the columns, data types, and notes for the contact\_list table. Next, I used a ALTER TABLE query to add a new column to the table name favorite. Then, I used a SHOW COLUMNS query to confirm the change in the contact\_list table.

Task 6: Update Records in the Contact List Table

Construct the SQL statement to update the existing records in the “contact\_list” table to use the new column that you created. Update the record(s) in the table by setting Michael Phelps as everyone's favorite contact (contact\_id = 1). The value for the “favorite” column should be set to “y” for these records.

mysql> UPDATE contact\_list

-> SET favorite = "y"

-> WHERE contact\_id = 1;

Query OK, 3 rows affected (0.03 sec)

Rows matched: 3 Changed: 3 Warnings: 0

mysql> SELECT \* FROM contact\_list;

+ --------------- + ----------- + ------------ + ---------- +

| connection\_id | person\_id | contact\_id | favorite |

+ --------------- + ----------- + ------------ + ---------- +

| 1 | 1 | 2 | NULL |

| 2 | 1 | 3 | NULL |

| 3 | 1 | 4 | NULL |

| 4 | 1 | 5 | NULL |

| 5 | 1 | 6 | NULL |

| 6 | 2 | 1 | y |

| 7 | 2 | 3 | NULL |

| 8 | 2 | 4 | NULL |

| 9 | 3 | 1 | y |

| 10 | 3 | 4 | NULL |

| 11 | 4 | 5 | NULL |

| 12 | 4 | 6 | NULL |

| 13 | 5 | 1 | y |

| 14 | 5 | 6 | NULL |

+ --------------- + ----------- + ------------ + ---------- +

14 rows in set (0.00 sec)

In the queries above, first, I used a UPDATE query to set Michael Phelps as everyone’s favorite contact WHERE the contact\_id was equal to 1. Then, I used a SELECT query to confirm the changes in the contact\_list table.

Task 7: Update Records in the Contact List Table

Construct the SQL statement to update the existing records in the “contact\_list” table to use the new column that you created. Update the remaining record(s) in the table by setting every contact who is NOT Michael Phelps (contact\_id <> 1) to not be a favorite. The value for the “favorite” column should be set to “n” for these records.

mysql> UPDATE contact\_list

-> SET favorite = "n"

-> WHERE contact\_id <> 1;

Query OK, 11 rows affected (0.03 sec)

Rows matched: 11 Changed: 11 Warnings: 0

mysql> SELECT \* FROM contact\_list;

+ --------------- + ----------- + ------------ + ---------- +

| connection\_id | person\_id | contact\_id | favorite |

+ --------------- + ----------- + ------------ + ---------- +

| 1 | 1 | 2 | n |

| 2 | 1 | 3 | n |

| 3 | 1 | 4 | n |

| 4 | 1 | 5 | n |

| 5 | 1 | 6 | n |

| 6 | 2 | 1 | y |

| 7 | 2 | 3 | n |

| 8 | 2 | 4 | n |

| 9 | 3 | 1 | y |

| 10 | 3 | 4 | n |

| 11 | 4 | 5 | n |

| 12 | 4 | 6 | n |

| 13 | 5 | 1 | y |

| 14 | 5 | 6 | n |

+ --------------- + ----------- + ------------ + ---------- +

14 rows in set (0.02 sec)

In the queries above, first, I used a UPDATE query to set everyone who isn’t Michael Phelps as not a favorite in the contact\_list table. Then, I used a SELECT query to confirm the changes in the contact\_list table.

Task 8: Insert Records to Contact List Table

Construct the SQL statement to insert at least 3 new records in the “contact\_list” table. Make sure that you use the new column that you created in the previous step. Note: You are required to add at least 3 new records with yourself as a new contact in the “contact\_list” table. Make sure that you provide a value (y or n) for the new “favorite” column.

mysql> INSERT INTO contact\_list (connection\_id, person\_id, contact\_id, favorite)

-> VALUES (15, 7, 1, "y"),

-> (16, 7, 2, "n"),

-> (17, 7, 4, "n");

Query OK, 3 rows affected (0.03 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT \* FROM contact\_list;

+ --------------- + ----------- + ------------ + ---------- +

| connection\_id | person\_id | contact\_id | favorite |

+ --------------- + ----------- + ------------ + ---------- +

| 1 | 1 | 2 | n |

| 2 | 1 | 3 | n |

| 3 | 1 | 4 | n |

| 4 | 1 | 5 | n |

| 5 | 1 | 6 | n |

| 6 | 2 | 1 | y |

| 7 | 2 | 3 | n |

| 8 | 2 | 4 | n |

| 9 | 3 | 1 | y |

| 10 | 3 | 4 | n |

| 11 | 4 | 5 | n |

| 12 | 4 | 6 | n |

| 13 | 5 | 1 | y |

| 14 | 5 | 6 | n |

| 15 | 7 | 1 | y |

| 16 | 7 | 2 | n |

| 17 | 7 | 4 | n |

+ --------------- + ----------- + ------------ + ---------- +

17 rows in set (0.00 sec)

In the above queries, first, I used a INSERT INTO query to add three new records including myself as one of the records with the favorite column as ‘y’ or ‘n’. Then, I used a SELECT query to confirm all the changes made in the contact\_list table.

Task 9: Create the Image Table

Construct the SQL statement to create a table named “image”. The columns, column data types, and column notes are provided here. Create the “image” table according to these specifications.

mysql> CREATE TABLE IF NOT EXISTS image (

-> image\_id INT(8) NOT NULL AUTO\_INCREMENT,

-> image\_name VARCHAR(50) NOT NULL,

-> image\_location VARCHAR(250) NOT NULL,

-> PRIMARY KEY (image\_id)

-> ) AUTO\_INCREMENT = 1;

Query OK, 0 rows affected (0.09 sec)

mysql> SHOW COLUMNS FROM image;

+ ---------------- + -------------- + ------ + ----- + --------- + ---------------- +

| Field | Type | Null | Key | Default | Extra |

+ ---------------- + -------------- + ------ + ----- + --------- + ---------------- +

| image\_id | int(8) | NO | PRI | NULL | auto\_increment|

| image\_name | varchar(50) | NO | | NULL | |

| image\_location| varchar(250) | NO | | NULL | |

+ ---------------- + -------------- + ------ + ----- + --------- + ---------------- +

3 rows in set (0.00 sec)

In the queries above, first, I used a CREATE TABLE query to create a table named image with the specified data types and column notes. Then, I used a SHOW COLUMNS query to confirm the changes in the image table.

Task 10: Create the Message-Image Intersection Table

Construct the SQL statement to create an intersection table named “message\_image”. The columns, column data types, and column notes are provided here. Create the “message\_image” table according to these specifications.

mysql> CREATE TABLE IF NOT EXISTS message\_image (

-> message\_id INT(8) NOT NULL,

-> image\_id INT(8) NOT NULL,

-> PRIMARY KEY (message\_id, image\_id));

Query OK, 0 rows affected (0.09 sec)

mysql> SHOW COLUMNS FROM message\_image;

+ ------------ + -------- + ------ + ----- + --------- + ------- +

| Field | Type | Null | Key | Default | Extra |

+ ------------ + -------- + ------ + ----- + --------- + ------- +

| message\_id | int(8) | NO | PRI | NULL | |

| image\_id | int(8) | NO | PRI | NULL | |

+ ------------ + -------- + ------ + ----- + --------- + ------- +

2 rows in set (0.01 sec)

In the queries above, first, I used a CREATE TABLE query to create a table named message\_image with the specified data types and column notes. Then, I used a SHOW COLUMNS query to confirm the changes made to the message\_image table.

Task 11: Insert Records to Image Table

Construct the SQL statement to insert 5 new records in the “image” table.

mysql> INSERT INTO messaging.image (image\_id, image\_name, image\_location)

-> VALUES

-> (1, "Picture 1", "New York City"),

-> (2, "Picture 2", "Boston"),

-> (3, "Picture 3", "Seattle"),

-> (4, "Picture 4", "Miami"),

-> (5, "Picture 5", "Las Vegas");

Query OK, 5 rows affected (0.04 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT \* FROM image;

+ - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - - - - - +

| image\_id | image\_name | image\_location |

+ - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - - - - - +

| 1 | Picture 1 | New York City |

| 2 | Picture 2 | Boston |

| 3 | Picture 3 | Seattle |

| 4 | Picture 4 | Miami |

| 5 | Picture 5 | Las Vegas |

+ - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - - - - - +

5 rows in set (0.00 sec)

In the queries above, first, I used a INSERT INTO query to add five new records to the image table. Then, I used a SELECT query to confirm the changes made to the image table.

Task 12: Insert Records to Message-Image Table

Construct the SQL statement to insert 5 new records in the “message\_image” intersection table. Note: You are required to add at least one record where at least one of Michael Phelp's messages includes at least one image. Also, you are required to add at least one message that has multiple images.

mysql> INSERT INTO messaging.message\_image (message\_id, image\_id)

-> VALUES

-> (1, 4),

-> (1, 1),

-> (2, 5),

-> (3, 2),

-> (5, 3);

Query OK, 5 rows affected (0.04 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT \* FROM message\_image;

+ ------------ + ---------- +

| message\_id | image\_id |

+ ------------ + ---------- +

| 1 | 1 |

| 1 | 4 |

| 2 | 5 |

| 3 | 2 |

| 5 | 3 |

+ ------------ + ---------- +

5 rows in set (0.00 sec)

In the queries above, first, I used a INSERT INTO query to add five new records to the message\_image table. Then, I used a SELECT query to confirm the changes made to the message\_image table.

Task 13: Find All of the Messages that Michael Phelps Sent

Construct the SQL statement to find all of the messages that Michael Phelps sent. Note: You must use the WHERE clause to set the conditions for this query. Display the following columns: - Sender's first name

- Sender's last name

- Receiver's first name

- Receiver's last name

- Message ID

- Message

- Message Timestamp

mysql> SELECT

-> pSender.first\_name AS "Sender's first name",

-> pSender.last\_name AS "Sender's last name",

-> pReceiver.first\_name AS "Receiver's first name",

-> pReceiver.last\_name AS "Receiver's last name",

-> m.message\_id AS "Message ID",

-> m.message AS "Message",

-> m.send\_datetime AS "Message Timestamp"

-> FROM message m

-> JOIN person pSender ON pSender.person\_id = m.sender\_id

-> JOIN person pReceiver ON pReceiver.person\_id = m.receiver\_id

-> WHERE pSender.person\_id = 1;

+---------------------+--------------------+-----------------------+----------------------+-----

-------+--------------------------------------------+---------------------+

| Sender's first name | Sender's last name | Receiver's first name | Receiver's last name | Mess

age ID | Message | Message Timestamp |

+---------------------+--------------------+-----------------------+----------------------+-----

-------+--------------------------------------------+---------------------+

| Michael | Phelps | Katie | Ledecky | 1 | Congrats on winning the 800m Freestyle! | 2016-12-25 09:00:00 |

| Michael | Phelps | Usain | Bolt | 4 | Thanks! You're the greatest sprinter ever | 2016-12-25 09:04:00 |

| Michael | Phelps | Allyson | Felix | 5 | Good luck on your race | 2016-12-25 09:05:00 |

+---------------------+--------------------+-----------------------+----------------------+-----

-------+--------------------------------------------+---------------------+

3 rows in set (0.01 sec)

In the above query, I used a SELECT/ FROM/ JOIN/ ON/ WHERE query to find all the messages that Michael Phelps sent.

Task 14: Find the Number of Messages Sent for Every Person

Construct the SQL statement to find the number of messages sent for every person. Note: You must use the WHERE clause to set the conditions for this query. Display the following columns: - Count of messages

- Person ID

- First Name

- Last Name

mysql> SELECT

-> COUNT(message.message) AS "Count of messages",

-> person.person\_id AS "Person ID ",

-> person.first\_name AS "First Name",

-> person.last\_name AS "Last Name"

-> FROM message

-> JOIN person

-> ON message.sender\_id = person.person\_id

-> WHERE person.person\_id BETWEEN 1 AND 5

-> GROUP BY person.person\_id;

+ - - - - - - - - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - +

| Count of messages | Person ID | First Name | Last Name |

+ - - - - - - - - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - +

| 3 | 1 | Michael | Phelps |

| 1 | 2 | Katie | Ledecky |

| 1 | 3 | Usain | Bolt |

+ - - - - - - - - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - - + - - - - - - - - - - - +

3 rows in set (0.00 sec)

In the above query, I used a SELECT/ COUNT/ FROM/ JOIN/ ON/ WHERE/ GROUP BY query to find the number of messages sent for every person.

Task 15: Find All of the Messages that Have At Least One Image Attached Using INNER JOINs Construct the SQL statement to find all of the messages that have at least one image attached using INNER JOINs. Note: For messages with multiple images, display only the first image for the message. Display the following columns:

- Message ID

- Message

- Message Timestamp

- First Image Name

- First Image Location

mysql> SELECT

-> message.message\_id AS "Message ID",

-> message.message AS "Message",

-> message.send\_datetime AS "Message Timestamp",

-> image.image\_name AS "First Image Name",

-> image.image\_location AS "First Image Location"

-> FROM message

-> JOIN message\_image ON message.message\_id = message\_image.message\_id

-> JOIN image ON message\_image.message\_id = image.image\_id

-> WHERE message\_image.message\_id BETWEEN 1 AND 5

-> GROUP BY message\_image.message\_id;

+------------+-----------------------------------------+---------------------+------------------

+----------------------+

| Message ID | Message | Message Timestamp | First Image Name | First Image Location |

+------------+-----------------------------------------+---------------------+------------------

+----------------------+

| 1 | Congrats on winning the 800m Freestyle! | 2016-12-25 09:00:00 | Picture 1

| New York City |

| 2 | Congrats on winning 23 gold medals! | 2016-12-25 09:01:00 | Picture 2

| Boston |

| 3 | You're the greatest swimmer ever | 2016-12-25 09:02:00 | Picture 3

| Seattle |

| 5 | Good luck on your race | 2016-12-25 09:05:00 | Picture 5

| Las Vegas |

+------------+-----------------------------------------+---------------------+------------------

+----------------------+

4 rows in set (0.00 sec)

In the query above, I used a SELECT/ FROM/ JOIN/ ON/ WHERE/ GROUP BY query to find all the messages that have at least one image attached.