

Dominic Grant

Final submission for SQL/NoSQL (MongoDB) Project:

The SQL portion I chose from old projects.

1. Table Patients(Patient_number INTEGER,
First_name char(30),
Last_name char(30),
Address char(30),
City char(30),
State char(30),
Zip char(5),
Balance INTEGER,
PRIMARYKEY(Patient_number));
2. Table Therapists(therapist_ID INTEGER,
First_name char(30),
Last_name char(30),
Street char(30),
City char(30),
State char(30),
Zip char(5),
PRIMARYKEY(therapist_ID));
3. Table Therapies(therapist_code INTEGER,
description char(30),
billable_unit char(30),
PRIMARYKEY(therapist_code));
4. Table session(therapist_id INTEGER,
Session_date DATE,
Session_length char(30),
Session_number INTEGER,
FOREIGNKEY(Therapist_id) REFERENCE therapist(therapist_id),
FOREIGNKEY(Patient_number) references to patients(patient_number);

I used the mongodb extension for VS code. I made my own dummy data as well.

The screenshot displays the MongoDB Playground interface within VS Code. The left pane shows the MongoDB Playground editor with a JavaScript script for inserting dummy data into a database. The right pane shows the Playground Result, displaying the JSON documents inserted into the database.

Playground Script:

```
14 t_id3 = ObjectId()//time searching for the therapist id if we
15 db.therapists.insertMany([
16   {'_id': t_id1,'therapist_id': 1, 'First_Name': 'John', 'Last
17   {'_id': t_id2,'therapist_id': 2, 'First_Name': 'Abe', 'Last
18   {'_id': t_id3,'therapist_id': 3, 'First_Name': 'Erik', 'Last
19 ])
20 p_id1 = ObjectId()//for adding new sessions later on, we would
21 p_id2 = ObjectId()//to the session
22 p_id3 = ObjectId()
23 db.patients.insertMany([
24   {'_id': p_id1,'patient_number_id': 1, 'First_Name': 'Sociopa
25   {'_id': p_id2,'patient_number_id': 2, 'First_Name': 'Mental'
26   {'_id': p_id3,'patient_number_id': 3, 'First_Name': 'Seek',
27 ])
28 db.therapies.insertMany([
29   {'therapist_code': 1, 'Description': 'What happened.', 'Billa
30   {'therapist_code': 2, 'Description': 'What happened.', 'Billa
31   {'therapist_code': 3, 'Description': 'What happened.', 'Billa
32 ])
33 db.sessions.insertMany([//i just set each therapist to be in a
34   {'therapist_id': t_id1,'patient_number_id': p_id1,'Session_D
35   {'therapist_id': t_id2,'patient_number_id': p_id2,'Session_D
36   {'therapist_id': t_id3,'patient_number_id': p_id3,'Session_D
37 ])
38 db.sessions.find()//prints out all the sessions. |
39 //This code is deleting the databases completely and putting t
40 //if you wanted to, you could remove the drop codes and you co
41
```

Playground Result:

```
1 {
2   {
3     "_id": {
4       "$oid": "6059114476defe30e4fc90a3"
5     },
6     "therapist_id": {
7       "$oid": "6059114376defe30e4fc909a"
8     },
9     "patient_number_id": {
10      "$oid": "6059114476defe30e4fc909d"
11    },
12    "Session_Date": {
13      "$date": "2021-03-22T21:51:00.533Z"
14    },
15    "Session_Length": "2 hours",
16    "Session_number": 1
17  },
18  {
19    "_id": {
20      "$oid": "6059114476defe30e4fc90a4"
21    },
22    "therapist_id": {
23      "$oid": "6059114376defe30e4fc909b"
24    },
25    "patient_number_id": {
26      "$oid": "6059114476defe30e4fc909e"
27    }
28  }
29 }
```

Database Interface:

The MongoDB extension interface shows the database **schoolNOSQLversion** with a size of 1.76KB, index size of 80KB, and 4 total collections. The collections are listed in a table:

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
patients	3	540B	180B	1	20KB	20KB
sessions	3	447B	149B	1	20KB	20KB
therapies	3	294B	98B	1	20KB	20KB
therapists	3	521B	174B	1	20KB	20KB

patients

sessions

therapies

therapists

FILTER { filter : example }

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("6059114476defe30e4fc909d")
patient_number_id: 1
First_Name: "Sociopath"
Last_Name: "Evil"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

```
_id: ObjectId("6059114476defe30e4fc909e")
patient_number_id: 2
First_Name: "Mental"
Last_Name: "Issues"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

```
_id: ObjectId("6059114476defe30e4fc909f")
patient_number_id: 3
First_Name: "Seek"
Last_Name: "Help"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

patients

sessions

therapies

therapists

FILTER { filter : example }

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("6059114476defe30e4fc90a3")
therapist_id: ObjectId("6059114376defe30e4fc909a")
patient_number_id: ObjectId("6059114476defe30e4fc909d")
Session_Date: 2021-03-22T21:51:00.533+00:00
Session_Length: "2 hours"
Session_number: 1
```

```
_id: ObjectId("6059114476defe30e4fc90a4")
therapist_id: ObjectId("6059114376defe30e4fc909b")
patient_number_id: ObjectId("6059114476defe30e4fc909e")
Session_Date: 2021-03-22T21:51:00.533+00:00
Session_Length: "2 hours"
Session_number: 2
```

```
_id: ObjectId("6059114476defe30e4fc90a5")
therapist_id: ObjectId("6059114376defe30e4fc909c")
patient_number_id: ObjectId("6059114476defe30e4fc909f")
Session_Date: 2021-03-22T21:51:00.533+00:00
Session_Length: "2 hours"
Session_number: 3
```

patients

sessions

therapies

therapists

FILTER { filter : example }

QUERY RESULTS 1-3 OF 3

`_id: ObjectId("6059114476defe30e4fc90a0")`
`therapist_code: 1`
`Description: "What happened."`
`Billable_Unit: "500$"`

`_id: ObjectId("6059114476defe30e4fc90a1")`
`therapist_code: 2`
`Description: "What happened."`
`Billable_Unit: "500$"`

`_id: ObjectId("6059114476defe30e4fc90a2")`
`therapist_code: 3`
`Description: "What happened."`
`Billable_Unit: "500$"`

sessions

therapies

therapists

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("6059114376defe30e4fc909a")
therapist_id: 1
First_Name: "John"
Last_Name: "Smith"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

```
_id: ObjectId("6059114376defe30e4fc909b")
therapist_id: 2
First_Name: "Abe"
Last_Name: "Newell"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

```
_id: ObjectId("6059114376defe30e4fc909c")
therapist_id: 3
First_Name: "Erik"
Last_Name: "Fielder"
Street: "Middle of nowhere 12345"
City: "Dustville"
State: "Oklahoma"
Zip: "12345"
```

I research queries and the modifiers that both the SQL and NoSQL use. I made several different queries that do the same thing for both programs.

I first print out all 4 tables completely. Next, I print out sessions that have a therapist_id that's equal to 1. Then I print out therapists that have a Last_name that has a e in it. That's followed by a query that prints out patients that have a patient_number of 1 or 3. Then another query that prints out patients that have a Balance between 1000 and 1500. Then finally, one more query that prints out patients with their Balance in ascending order.

I did these in both programs making use of their various syntaxes. Here are them below.

SQL:

```
String query = "SELECT * FROM therapists";
query = "SELECT * FROM patients";
query = "SELECT * FROM therapies";
query = "SELECT * FROM sessions";
query = "SELECT * FROM sessions WHERE therapist_id = 1";
query = "SELECT * FROM therapists WHERE Last_name LIKE '%e%'";
query = "SELECT * FROM patients WHERE patient_number IN (1,3)";
query = "SELECT * FROM patients WHERE Balance BETWEEN 1000 AND 1500";
query = "SELECT * FROM patients ORDER BY Balance";
```

NoSQL (MongoDB):

```
const myCursor = db.therapists.find();
myCursor = db.patients.find();
myCursor = db.therapies.find();
myCursor = db.sessions.find();
myCursor = db.sessions.find({'therapist_id': t_id1});
myCursor = db.therapists.find({'Last_Name': { $regex: /e/}});
myCursor = db.patients.find({'patient_number_id': { $in: [1,3]}});
myCursor = db.patients.find({'Balance': { $gte: 1000, $lte: 1500}});
myCursor = db.patients.find().sort({'Balance': 1});
```

I put commented info in the code for some of the ones that aren't as straightforward and obvious in their use.

I also looked into triggers as asked. I got a simple trigger that adds in the therapies table once it's detected that the patients table has been inserted into for my SQL code.

```
why.execute("CREATE TRIGGER trig1 AFTER INSERT ON patients REFERENCING NEW AS upd
atedrow FOR EACH ROW MODE DB2SQL INSERT INTO therapies VALUES(updatedrow.patient_
number,'What happened.','500$')");
    System.out.println("Created trigger that does the therapies table after a
new row is inserted into patients.");
```

Basically upon a insertion into the patient table it takes the patient_number from that inserted information, and puts that as the therapy number in the therapies insertion, along with the generic info and cost. More commented info in the code.

I did have a bug with implementing my MongoDB trigger however. MongoDB's triggers make use of serverside code that you customize to detect when something happens, and to trigger the trigger so to speak, so it's not like the SQL where you just put it in the same code.

My MongoDB trigger works when it has a test run done, and inputs the patient data, but when I run my MongoDB playground code, it acts like it doesn't see that. I have gotten in touch with MongoDB support to see if they can help, but as of now they haven't gotten back with info on why this is happening.

Operation Type

This trigger will only execute on these operations.

☒ Insert ☐ Update ☐ Delete ☐ Replace

Full Document

By turning on Full Document, you will receive the document created or modified in your [change event](#). For Delete operations, the full document will not exist.



✓ FUNCTION

Select An Event Type

[Learn how to use Amazon EventBridge with Triggers](#)

Function

EventBridge

Function

```
1 exports = function(changeEvent) {  
2   const collection = context.services.get('Cluster0').db('schoolINOSQLversion').collection('therapies');  
3   collection.insertOne({therapist_code: 1, Description: 'What happened.', Billable_Unit: '500$'});  
4   collection.insertOne({therapist_code: 2, Description: 'What happened.', Billable_Unit: '500$'});  
5   collection.insertOne({therapist_code: 3, Description: 'What happened.', Billable_Unit: '500$'});  
6 }
```

