Database Team Based Assignment

Course Title: Enterprise Software Platforms

Professor: Mr. Rakesh Ranjan

Submitted by: Project Team #7

- Amita Vasudev Kamat
- Kedhara Nethra Thiruvuru
- Mohammed Haroon Shareef
- Pavana Srinivasadeshika Achar

Contents:

SQLite

- Installed SQLite Add-ons for Firefox from:

https://addons.mozilla.org/en-us/firefox/addon/sqlite-manager/ (Links to an external site.)

- Designed a sample database for a Purchase order management system.
- Created a sample schema for the designed database:
- Database includes 6 tables created.

Entities:

1. **Employees**
   - Create Query for Employees Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
CREATE TABLE IF NOT EXISTS Employees("EID" INTEGER PRIMARY KEY NOT NULL,"ENAME" VARCHAR NOT NULL,"EMOBILE" INTEGER,"EGENDER" VARCHAR,"ESALESREPID" INTEGER NOT NULL,"ESTORECODE" VARCHAR NOT NULL);
```

Run SQL | Actions | Last Error: not an error

Columns (6)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	EID	INTEGER	1	null	1
1	ENAME	VARCHAR	1	null	0
2	EMOBILE	INTEGER	0	null	0
3	EGENDER	VARCHAR	0	null	0
4	ESALESREPID	INTEGER	1	null	0
5	ESTORECODE	VARCHAR	1	null	0

   - Insert Query for Employees Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
INSERT INTO Employees VALUES ( 123,"Nancy",9999998888,"F",321,"CA1")
```

Run SQL | Actions | Last Error: not an error

- Sample data inserted into the Employee Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
SELECT * FROM Employees
```

Run SQL | Actions | Last Error: not an error

Structure | Browse & Search | Execute SQL | DB Settings

TABLE Employees — Search | Show All | Add | Duplicate | Edit

EID	ENAME	EMOBILE	EGENDER	ESALESREPID
123	Nancy	9999998888	F	321
231	PRAKASH	8250000009	M	909
456	SAI	9250000005	F	654
555	JOHN	5250000009	M	666
789	RAJA	9250000009	M	987

2. Customers

- Create Query for Customers Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
CREATE TABLE IF NOT EXISTS Customers("CID" INTEGER PRIMARY KEY NOT NULL,"CNAME" VARCHAR NOT NULL,"CMOBILE" INTEGER,"CGENDER" VARCHAR,"CSALESREPID" INTEGER NOT NULL,FOREIGN KEY ("CSALESREPID") REFERENCES "Employees" ("EID"))
```

Run SQL | Actions | Last Error: not an error

- Sample data inserted into the Customers Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
SELECT * FROM Customers
```

Run SQL | Actions | Last Error: not an error

CID	CNAME	CMOBILE	CGENDER	CSALESREPID
11	Neths	9999999999	F	123
22	Roger	9999999998	M	231
33	Smith	9999999997	M	456
44	Raj	9999999993	M	555
55	Angel	9999999991	F	789

3. Products

- Create Query for Products Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
CREATE TABLE IF NOT EXISTS Products("PID" INTEGER PRIMARY KEY NOT NULL,"PNAME" VARCHAR NOT NULL,"PSTOCK" INTEGER,"PPRICE" INTEGER,"PVENDOR" VARCHAR);
```

Run SQL | Actions | Last Error: not an error

Columns (5)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	PID	INTEGER	1	null	1
1	PNAME	VARCHAR	1	null	0
2	PSTOCK	INTEGER	0	null	0
3	PPRICE	INTEGER	0	null	0
4	PVENDOR	VARCHAR	0	null	0

- Sample data inserted into the Products Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
SELECT * FROM Products
```

Run SQL | Actions | Last Error: not an error

PID	PNAME	PSTOCK	PPRICE	PVENDOR
1	Rasberry pi 3	5	64	Rasberry
2	IPHONE 7	9	999	Apple
3	Dell Touch	2	999	Dell
4	Modem	50	230	Netgear
5	TV 55inch	3	300	Toshiba

4. **Orders**

- Create Query for Orders Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
CREATE TABLE IF NOT EXISTS Orders("OID" INTEGER PRIMARY KEY NOT NULL,"ODATE" DATE NOT NULL,"SHIPMTDATE" DATE DEFAULT NULL,"OSTATUS" VARCHAR(10) NOT NULL,"CID" INTEGER UNSIGNED NOT NULL ,FOREIGN KEY ("CID") REFERENCES "Customers"("EID"));
```

Run SQL | Actions | Last Error: not an error

Columns (5)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	OID	INTEGER	1	null	1
1	ODATE	DATE	1	null	0
2	SHIPMTD...	DATE	0	NULL	0
3	OSTATUS	VARCHA...	1	null	0
4	CID	INTEGER ...	1	null	0

- Sample data inserted into the Orders Table

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

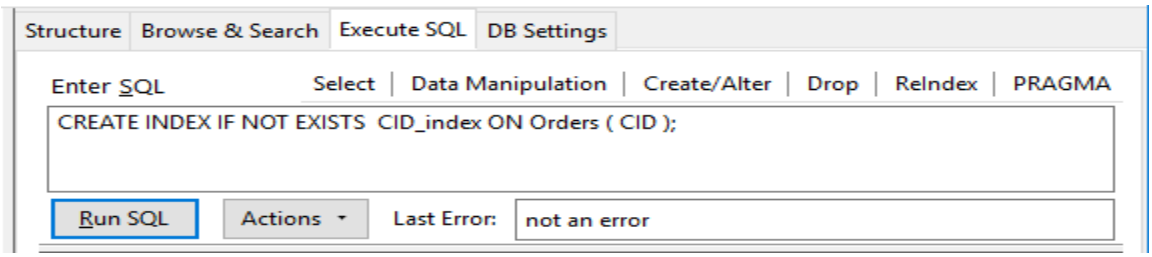
```
SELECT * FROM Orders
```

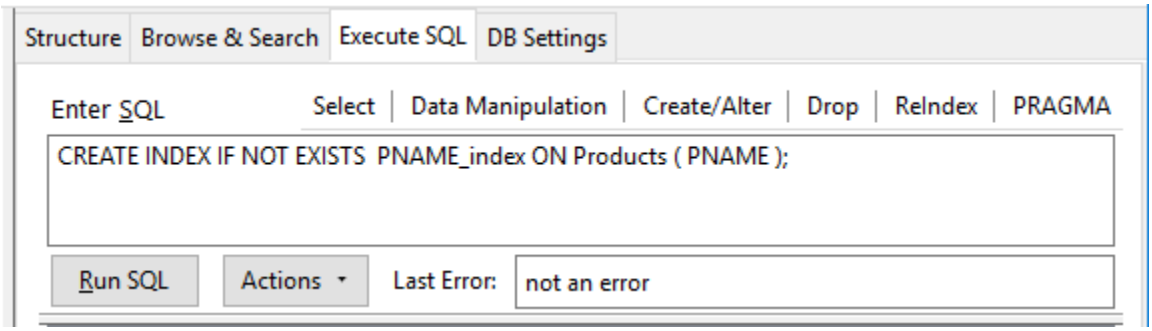
Run SQL | Actions | Last Error: not an error

OID	ODATE	SHIPMTDATE	OSTATUS	CID	PID
34	2/10/2017		SHIPPED	11	3
55	1/10/2017		DELIVERED	22	2
77	2/20/2017		SHIPPED	11	3
86	2/20/2017		PROCESSING	33	4
98	2/14/2017		ONTRAC	44	0

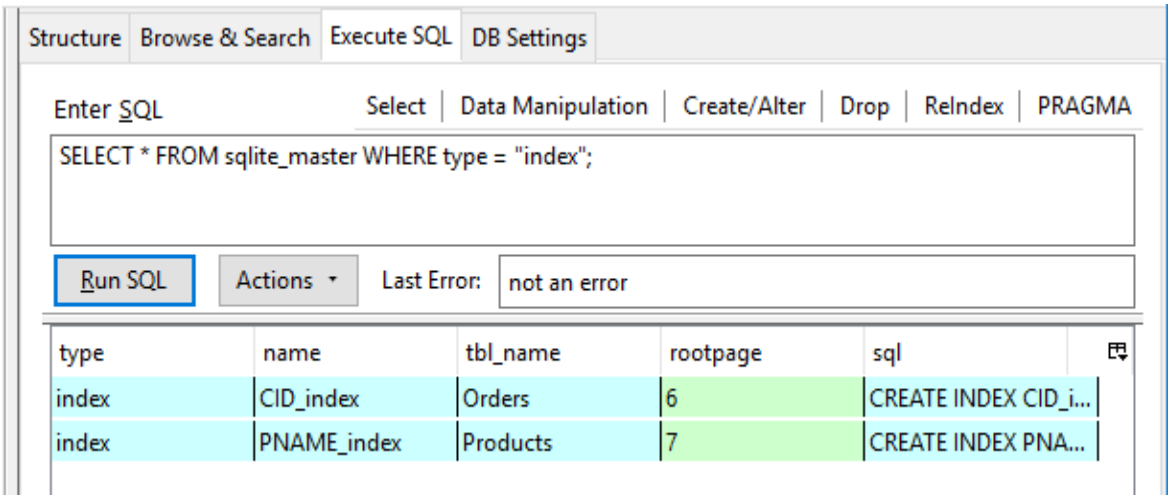
- Queries run on the sample data:

1) Create INDEX for Product and Orders tables





2) Select all the Indexes from the database and list down



3) Select data from customer and order tables using “WHERE AND LEFT OUTER JOIN “

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
select Customers.CID,Customers.CNAME,Orders.OID,Orders.OSTATUS FROM Customers LEFT OUTER JOIN
ORDERS ON Customers.CID>=10 and Orders.OID>=40
```

Run SQL | Actions | Last Error: not an error

CID	CNAME	OID	OSTATUS
11	Neths	55	DELIVERED
11	Neths	86	PROCESSING
11	Neths	98	ONTRAC
22	Roger	55	DELIVERED
22	Roger	86	PROCESSING
22	Roger	98	ONTRAC
33	Smith	55	DELIVERED
33	Smith	86	PROCESSING
33	Smith	98	ONTRAC
44	Raj	55	DELIVERED
44	Raj	86	PROCESSING
44	Raj	98	ONTRAC
55	Angel	55	DELIVERED
55	Angel	86	PROCESSING
55	Angel	98	ONTRAC

4) Fetch the Customer and order details whose Customer id matches with customer id in the order table and product id matches from order table.

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL — Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
Select C.CID,C.CNAME,O.OID,O.OSTATUS,P.PID FROM CUSTOMERS AS C,ORDERS AS O,PRODUCTS AS P WHERE C.CID=O.CID and O.PID = P.PID
```

Run SQL | Actions | Last Error: not an error

CID	CNAME	OID	OSTATUS	PID
11	Neths	34	SHIPPED	3
22	Roger	55	DELIVERED	2
11	Neths	77	SHIPPED	3
33	Smith	86	PROCESSING	4

5) Fetch Customer data whose order is delivered from Order table using table associations and ORDER BY clause

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL

Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
Select C.CID,C.CNAME,O.OID,O.OSTATUS,P.PID FROM CUSTOMERS AS C,ORDERS AS O,PRODUCTS AS P WHERE C.CID=O.CID and O.OSTATUS="DELIVERED" ORDER BY P.PID DESC
```

Run SQL | Actions | Last Error: not an error

CID	CNAME	OID	OSTATUS	PID
22	Roger	55	DELIVERED	5
22	Roger	55	DELIVERED	4
22	Roger	55	DELIVERED	3
22	Roger	55	DELIVERED	2
22	Roger	55	DELIVERED	1

6) Fetch Employees who sold to customers matching in the Customers and Employee schemas

Structure | Browse & Search | Execute SQL | DB Settings

Enter SQL

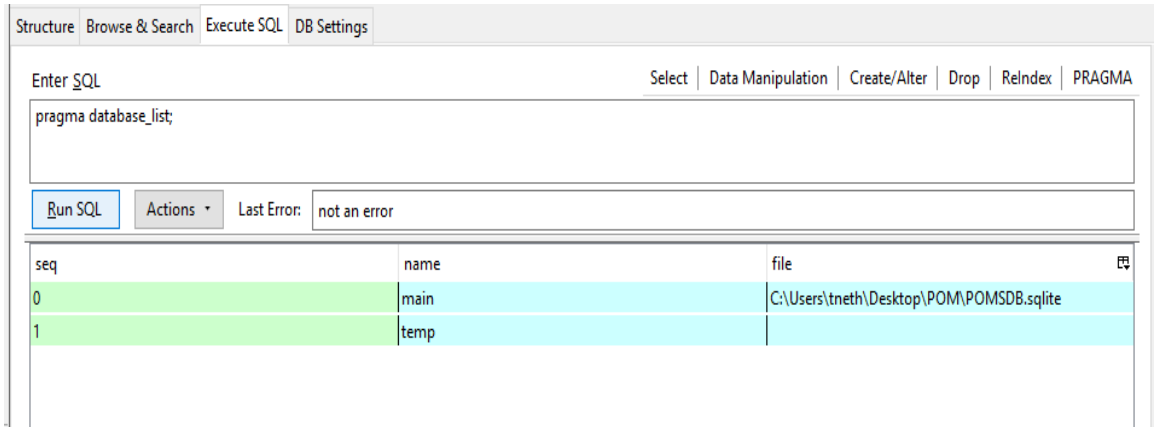
Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
Select E.ENAME,E.ESALESREPID FROM EMPLOYEES AS E,CUSTOMERS AS C WHERE E.ESALESREPID=C.CSALESREPID
```

Run SQL | Actions | Last Error: not an error

ENAME	ESALESREPID
PRAKASH	909
SAI	654
JOHN	666

7) Pragma-Database list



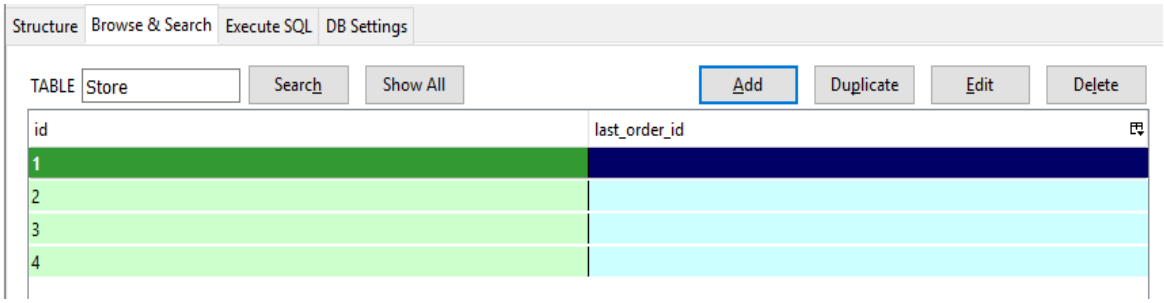
8) Perform Triggers for Store and Sale schema

- **CREATE Table STORE And SALE**

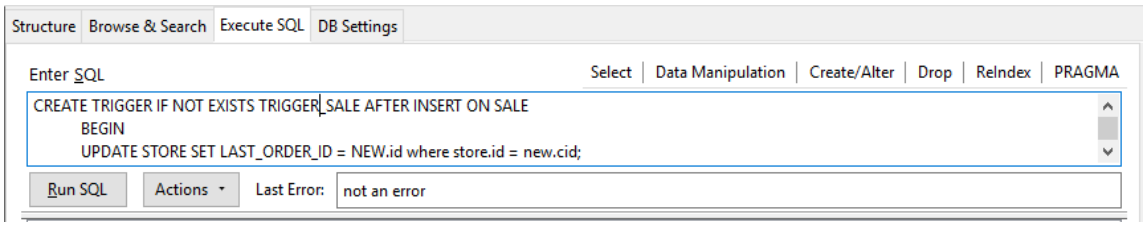
CREATE TABLE "Sale" ("id" INTEGER PRIMARY KEY NOT NULL, "item_id" INTEGER, "cid" INTEGER)

CREATE TABLE "Store" ("id" INTEGER PRIMARY KEY, "last_order_id" INTEGER NOT NULL)

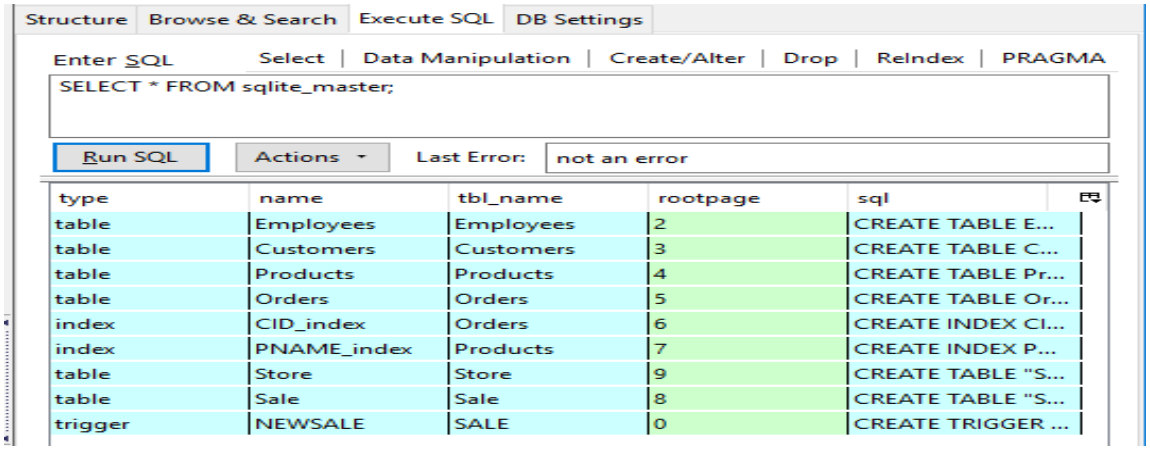
- **INSERT sample data Into STORE Schema - COLUMN ID**



- **Create a trigger -TRIGGER_sale for the Sale schema**

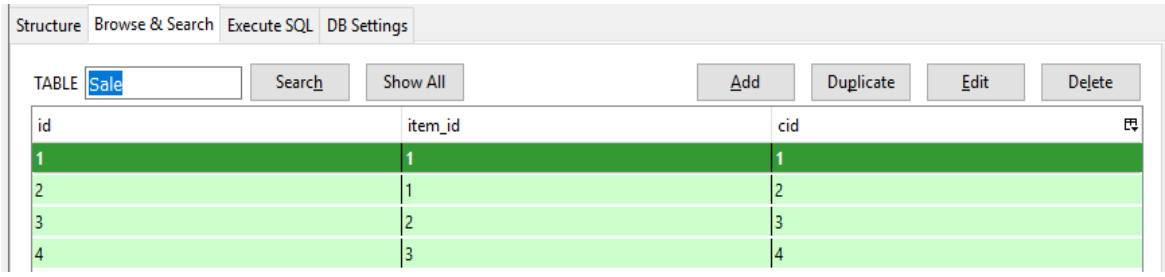


- **Display Tables, index and Triggers created from the Database.**



type	name	tbl_name	rootpage	sql
table	Employees	Employees	2	CREATE TABLE E...
table	Customers	Customers	3	CREATE TABLE C...
table	Products	Products	4	CREATE TABLE Pr...
table	Orders	Orders	5	CREATE TABLE Or...
index	CID_index	Orders	6	CREATE INDEX CI...
index	PNAME_index	Products	7	CREATE INDEX P...
table	Store	Store	9	CREATE TABLE "S...
table	Sale	Sale	8	CREATE TABLE "S...
trigger	NEWSALE	SALE	0	CREATE TRIGGER ...

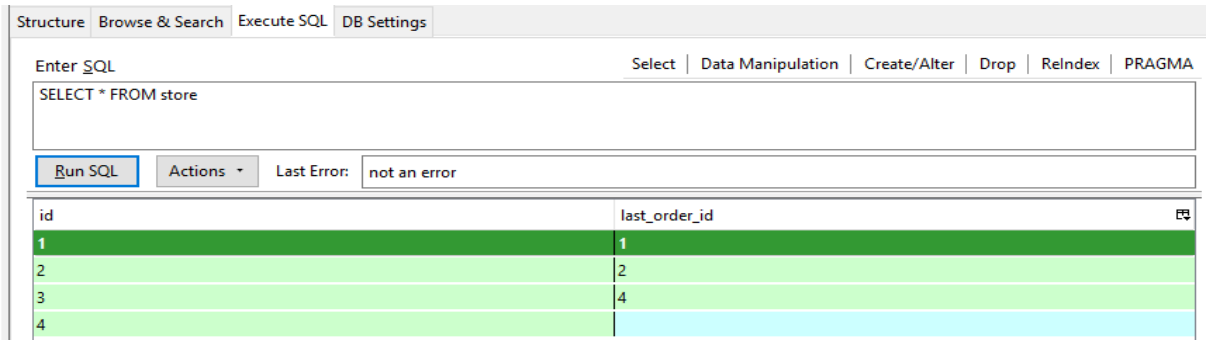
INSERT data INTO SALE Schema



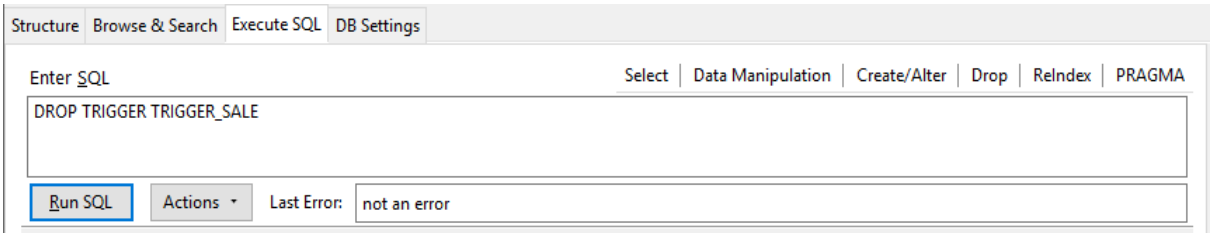
id	item_id	cid
1	1	1
2	1	2
3	2	3
4	3	4

And after each INSERT To SALE Schema, The STORE Schema-Last_order_id COLUMN gets updated

AUTOMATICALLY as per the Trigger created.



To DROP Trigger



DB 2 Express C

- Downloaded DB2 express C from http:// (Links to an external site.)www-03.ibm.com/software/products/en/db2expressc (Links to an external site.)
- Creating a sample database using db2sampl command

  Command used: db2sampl -name sampleDB.

```
Haroon@DESKTOP-DG5PM9H MINGW64 /c/Program Files/IBM/SQLLIB/BIN
$ db2sampl -name sampleDB

  Creating database "sampleDB"...
  Connecting to database "sampleDB"...
  Creating tables and data in schema "HAROON"...
  Creating tables with XML columns and XML data in schema "HAROON"...

  'db2sampl' processing complete.


Haroon@DESKTOP-DG5PM9H MINGW64 /c/Program Files/IBM/SQLLIB/BIN
```

  Connect to database sampleDB:

```
db2 => connect to sampleDB

   Database Connection Information

 Database server        = DB2/NT64 11.1.1.1
 SQL authorization ID   = HAROON
 Local database alias   = SAMPLEDB

db2 => |
```

- Running sample Query:

  QUERY: select workdept,sum(salary) as GROUP_SALARY from emp where salary > 70000 group by workdept order by GROUP_SALARY desc

```
db2 => select workdept,sum(salary) as GROUP_SALARY from emp where salary > 70000 group by work
dept order by GROUP_SALARY desc

WORKDEPT GROUP_SALARY
-------- ---------------------------------
C01                            172050.00
A00                            152750.00
D21                             96170.00
B01                             94250.00
E11                             89750.00
E21                             86150.00
E01                             80175.00
D11                             72250.00

  8 record(s) selected.

db2 => |
```

- Generate Explain plan for query:
  - Set current explain mode explain

```
db2 => set current explain mode explain
DB20000I  The SQL command completed successfully.
db2 => |
```

  - Execute the query to generate explain plan:

```
db2 => select workdept,sum(salary) as GROUP_SALARY from emp where salary > 70000 group by workdept order by GROUP_SALARY
 desc
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604
db2 =>
```

  - Close the explain mode:

```
db2 => set current explain mode no
DB20000I  The SQL command completed successfully.
db2 =>
```

  - Generate explain plan:

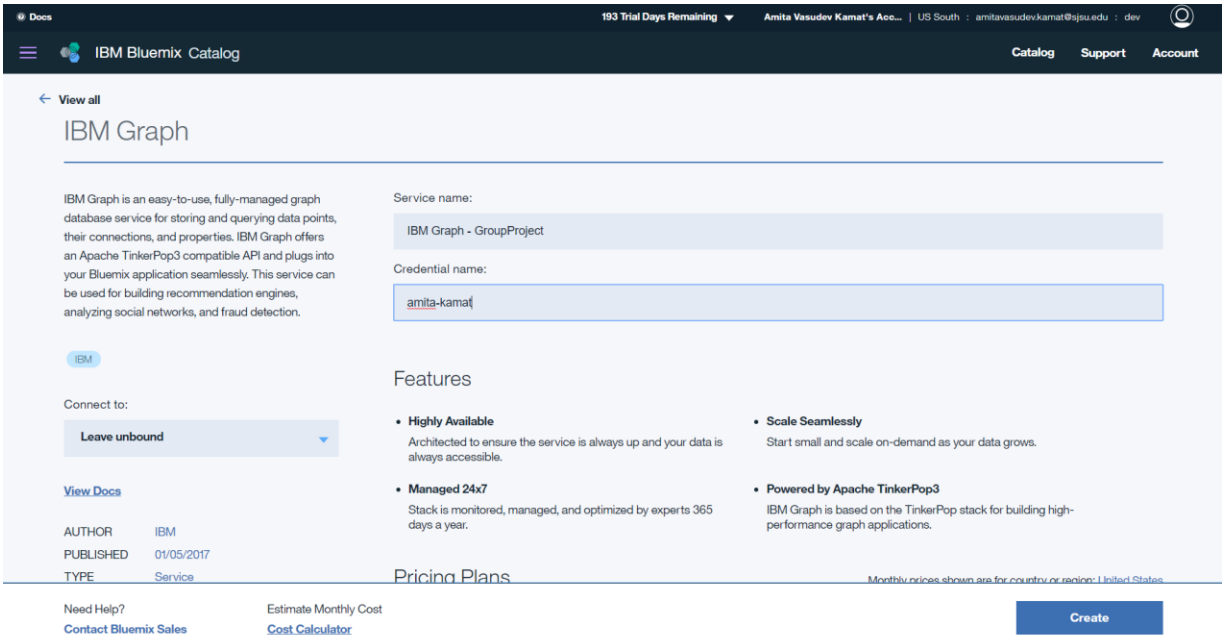
```
$ db2exfmt -d haroon -1 -o explainPlan.txt
Connecting to the Database.
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

Connect to Database Successful.
Output is in explainPlan.txt.
Executing Connect Reset -- Connect Reset was Successful.
```

The output will be stored in explainPlan.txt (included in appendix)

Graph Data Store

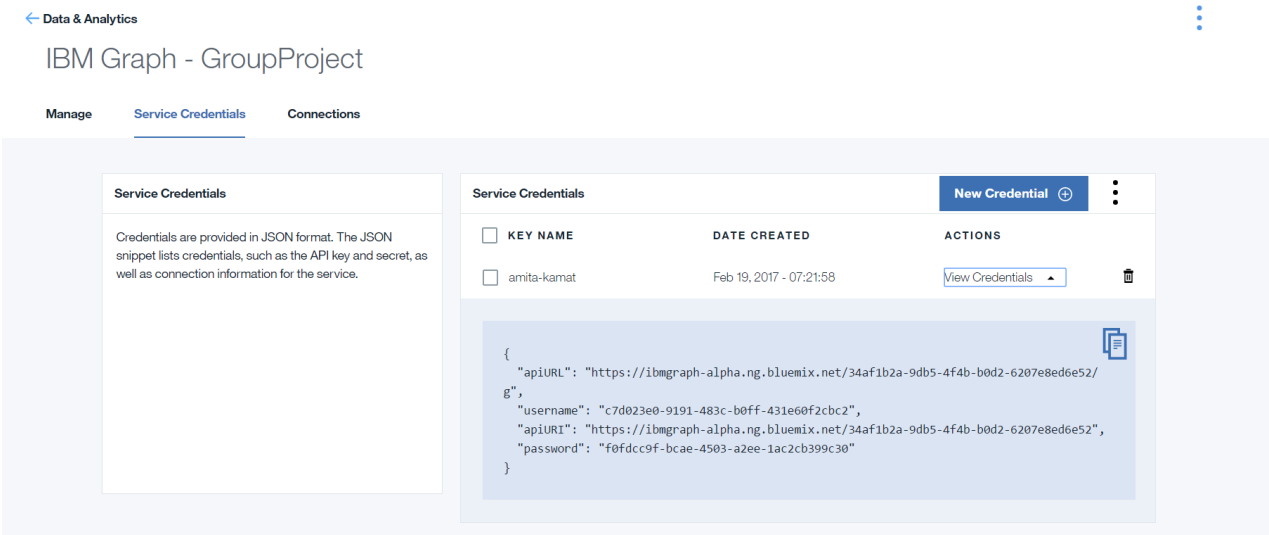
- Signed up for IBM Bluemix at www.bluemix.net (Links to an external site.)
- Navigating the catalog to the Data and Analytics section and creating an IBM graph service:

    1. Provide Service Name and Credential Name and create the service.
    2. Open your services from home menu and click on newly created graph service.



- Once adding an instance of graph service to the bluemix account, three key pieces of information out of four provisioned credentials will be used to connect to the database:
    - apiURL
    - username
    - password

  Base URL can be fetched from apiURL in the credentials by trimming the '/g' at the end.



Save the username, password and base URL in environment variables for simplicity:

amita@LAPTOP-E8TGACIL:~$ export username="c7d023e0-9191-483c-b0ff-431e60f2cbc2"

amita@LAPTOP-E8TGACIL:~$ export password="f0fdcc9f-bcae-4503-a2ee-1ac2cb399c30"

amita@LAPTOP-E8TGACIL:~$ export BASE_URL=”https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52”

```
amita@LAPTOP-E8TGACIL:~$ export username="c7d023e0-9191-483c-b0ff-431e60f2cbc2"
amita@LAPTOP-E8TGACIL:~$ export password="f0fdcc9f-bcae-4503-a2ee-1ac2cb399c30"
amita@LAPTOP-E8TGACIL:~$ export BASE_URL="https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52"
```

- Obtaining a session token to authenticate the connection before using the service:

curl --user "$username:$password" "$BASE_URL/_session"

Response:

{"gds-token":"YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0="}

Note the token generated for creating a graph, defining schema and other database transactions.

```
c-843a-aeb8bf7e362f"}amita@LAPTOP-E8TGACIL:~$ curl --user "$username:$password" "$BASE_URL/_session"
{"gds-token":"YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0="}
```

- Creating a graph:

amita@LAPTOP-E8TGACIL:~$ curl -H "Content-Length: 0" -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -v "$BASE_URL/_graphs"

```
amita@LAPTOP-E8TGACIL: ~
amita@LAPTOP-E8TGACIL:~$ curl -H "Content-Length: 0" -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhW
UTBnZWdMSXpzND0=" -v "$BASE_URL/_graphs"
* Hostname was NOT found in DNS cache
*   Trying 75.126.70.44...
* Connected to ibmgraph-alpha.ng.bluemix.net (75.126.70.44) port 443 (#0)
* successfully set certificate verify locations:
*   CAfile: none
  CApath: /etc/ssl/certs
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server key exchange (12):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
*        subject: C=US; ST=New York; L=Armonk; O=International Business Machines Corporation; CN=*.ng.bluemix.net
*        start date: 2014-09-29 00:00:00 GMT
*        expire date: 2017-11-08 12:00:00 GMT
*        subjectAltName: ibmgraph-alpha.ng.bluemix.net matched
*        issuer: C=US; O=DigiCert Inc; CN=DigiCert SHA2 Secure Server CA
*        SSL certificate verify ok.
> POST /34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/_graphs HTTP/1.1
> User-Agent: curl/7.35.0
> Host: ibmgraph-alpha.ng.bluemix.net
> Accept: */*
> Content-Length: 0
> Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=
>
< HTTP/1.1 201 Created
< X-Backside-Transport: OK OK
< Connection: Keep-Alive
< Transfer-Encoding: chunked
< Request-Id: 2f44189a-e0a0-40d0-a644-34dd79faa5b3
< Set-Cookie: JSESSIONID=%20; Max-Age=3600; Path=/; HttpOnly; Secure
< Content-Type: */*
< Date: Mon, 20 Feb 2017 02:23:46 GMT
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< X-Global-Transaction-ID: 3690566017
<
* Connection #0 to host ibmgraph-alpha.ng.bluemix.net left intact
{"graphId":"af1adaee-895d-4d69-a513-195d00187aaf","dbUrl":"https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf"}amita@LAPTOP-E8TGACIL:~$
```

<u>Response:</u>

{"graphId":"af1adaee-895d-4d69-a513-195d00187aaf","dbUrl":"https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf"}

The response includes graphID and dbURL generated for the created graph. This dbURL here will be used at the time of defining the schema.

- Getting the list of all graphs in the database:

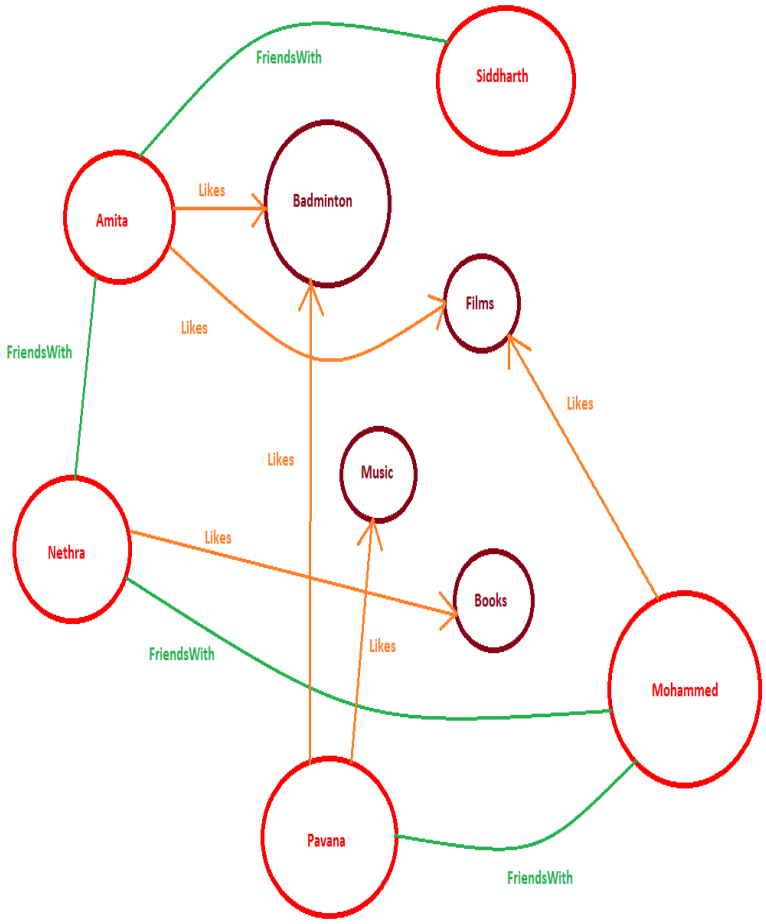
amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs" -X GET -u "$username:$password"

Response:
{"graphs":["14cc0f3d-0fa2-479c-a1ab-ccdb8a88c873","62ec6492-054c-4d85-8b97-d971188a1bbb","67d23268-e5e8-4e5d-8377-dd68de5c6360","g"]}

```
amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs" -X GET -u "$username:$password"
{"graphs":["14cc0f3d-0fa2-479c-a1ab-ccdb8a88c873","62ec6492-054c-4d85-8b97-d971188a1bbb","67d23268-e5e8-4e5d-8377-dd68de5c6360","g"]}amita@LAPTOP-E8TGA
```

The response will be a list of graphs which includes custom created and default graphs loaded at the time of service creation.

The graph diagram below reflects the sample data we intend to connect in a meaningful way by defining a schema to accommodate the relationships and property keys for each graph entity.



- Create schema for the graph

1. Save the schema definition in graph-schema.json format.

```
amita@LAPTOP-E8TGACIL:~$ vim graph-schema.json
amita@LAPTOP-E8TGACIL:~$ cat graph-schema.json

{
  "propertyKeys": [
    {"name": "personName", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "interestName", "dataType": "String", "cardinality": "SINGLE"}
  ],
  "vertexLabels": [
    {"name": "person"},
    {"name": "interests"}
  ],
  "edgeLabels": [
    { "name": "likes" },
    { "name": "friendsWith" }
  ],
  "vertexIndexes": [
    {"name": "vByPersonName", "propertyKeys": ["personName"], "composite": true, "unique": false},
    {"name": "vByInterestName", "propertyKeys": ["interestName"], "composite": true, "unique": false}
  ]
}
```

2. Run the command:

amita@LAPTOP-E8TGACIL:~$ curl -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/schema --data @graph-schema.json

<u>Response:</u>

{"requestId":"215c3702-2d24-4548-9582-55f34a2baae8","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"propertyKeys":[{"name":"personName","dataType":"String","cardinality":"SINGLE"},{"name":"interestName","dataType":"String","cardinality":"SINGLE"}],"vertexLabels":[{"name":"person"},{"name":"interests"}],"edgeLabels":[{"name":"likes","directed":true,"multiplicity":"MULTI"},{"name":"friendsWith","directed":true,"multiplicity":"MULTI"}],"vertexIndexes":[{"name":"vByPersonName","composite":true,"unique":false,"propertyKeys":["personName"],"requiresReindex":false,"type":"vertex"},{"name":"vByInterestName","composite":true,"unique":false,"propertyKeys":["interestName"],"requiresReindex":false,"type":"vertex"}],"edgeIndexes":[]}],"meta":{}}}

```
amita@LAPTOP-E8TGACIL:~$ curl -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1
NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af
1adaee-895d-4d69-a513-195d00187aaf/schema --data @graph-schema.json
{"requestId":"215c3702-2d24-4548-9582-55f34a2baae8","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"propertyKeys":[{"name":"pers
onName","dataType":"String","cardinality":"SINGLE"},{"name":"interestName","dataType":"String","cardinality":"SINGLE"}],"vertexLabels":[{"name":"person
"},{"name":"interests"}],"edgeLabels":[{"name":"likes","directed":true,"multiplicity":"MULTI"},{"name":"friendsWith","directed":true,"multiplicity":"MU
LTI"}],"vertexIndexes":[{"name":"vByPersonName","composite":true,"unique":false,"propertyKeys":["personName"],"requiresReindex":false,"type":"vertex"},
{"name":"vByInterestName","composite":true,"unique":false,"propertyKeys":["interestName"],"requiresReindex":false,"type":"vertex"}],"edgeIndexes":[]}],
"meta":{}}}amita@LAPTOP-E8TGACIL:~$
```

3. Inspect the schema by GET request:

amita@LAPTOP-E8TGACIL:~$ curl -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -sS https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/schema

<u>Response:</u>

{"requestId":"0366bd76-3e97-40cf-8648-43945dee8fe8","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"propertyKeys":[{"name":"personName","dataType":"String","cardinality":"SINGLE"},{"name":"interestName","dataType":"String","cardinality":"SINGLE"}],"vertexLabels":[{"name":"person"},{"name":"interests"}],"edgeLabels":[{"name":"likes","directed":true,"multiplicity":"MULTI"},{"name":"friendsWith","directed":true,"multiplicity":"MULTI"}],"vertexIndexes":[{"name":"vByPersonName","composite":true,"unique":false,"propertyKeys":["personName"],"requiresReindex":false,"type":"vertex"},{"name":"vByInterestName","composite":true,"unique":false,"propertyKeys":["interestName"],"requiresReindex":false,"type":"vertex"}],"edgeIndexes":[]}],"meta":{}}}

- Creating vertices as defined in the schema:
    - Add ‘Person’ Vertices to the graph

amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/vertices -d '{"label": "person", "properties": {"personName": "Amita"}}'

<u>Response:</u>
{"requestId":"bd44eb20-776f-46fa-8c34-62f0b143e7be","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4184,"label":"person","type":"vertex","properties":{"personName":[{"id":"16z-388-sl","value":"Amita"}]}}],"meta":{}}}

```
amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6
TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8e
d6e52/af1adaee-895d-4d69-a513-195d00187aaf/vertices -d '{"label": "person", "properties": {"personName": "Amita"}}'
{"requestId":"bd44eb20-776f-46fa-8c34-62f0b143e7be","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4184,"label":"person","t
ype":"vertex","properties":{"personName":[{"id":"16z-388-sl","value":"Amita"}]}}],"meta":{}}}amita@LAPTOP-E8TGACIL:~$
```

Similarly, add some more Person vertices to the graph according to the diagram.

<u>NOTE: Note the ID of all vertices.</u>

- Add Interest Vertices to the graph

amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/vertices -d '{"label": "interest", "properties": {"interestName": "books"}}'

<u>Response:</u>
{"requestId":"69b9e25a-b45b-4d15-95c9-0941c34c9606","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4280,"label":"interest","type":"vertex","properties":{"interestName":[{"id":"17b-3aw-1l1","value":"books"}]}}],"meta":{}}}

Similarly, add some more Interest Vertices.

- Get vertices by ID

amita@LAPTOP-E8TGACIL:~$ curl -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -sS https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/vertices/4280

Response:
{"requestId":"ea71a242-ce45-44a0-8b41-76dc72862689","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4280,"label":"interest","type":"vertex","properties":{"interestName":[{"id":"17b-3aw-1l1","value":"books"}]}}],"meta":{}}}

```
amita@LAPTOP-E8TGACIL:~$ curl -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1
NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -sS https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187a
af/vertices/4280
{"requestId":"ea71a242-ce45-44a0-8b41-76dc72862689","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4280,"label":"interest",
"type":"vertex","properties":{"interestName":[{"id":"17b-3aw-1l1","value":"books"}]}}],"meta":{}}}amita@LAPTOP-E8TGACIL:~$
```

- Adding Edges to the graph: connecting the vertices to make the data more meaningful.

  1. Create an edge for Amita “friends with” Siddharth and vice versa to show both are friends with each other.

amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/edges -d '{ "outV": 4184, "label": "friendsWith", "inV": 4176 }'

Response:
{"requestId":"0be06b85-7546-47b7-872d-49ea0cc89e4d","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":"odxcb-388-4r9-380","label":"friendsWith","type":"edge","inVLabel":"person","outVLabel":"person","inV":4176,"outV":4184}],"meta":{}}}

```
amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6
TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8e
d6e52/af1adaee-895d-4d69-a513-195d00187aaf/edges -d '{ "outV": 4184, "label": "friendsWith", "inV": 4176 }'
{"requestId":"0be06b85-7546-47b7-872d-49ea0cc89e4d","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":"odxcb-388-4r9-380","lab
el":"friendsWith","type":"edge","inVLabel":"person","outVLabel":"person","inV":4176,"outV":4184}],"meta":{}}}amita@LAPTOP-E8TGACIL:~$
```

Likewise create the other edges for other vertices.

2. Create an edge Amita “likes” Badminton

amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/edges -d '{ "outV": 4184, "label": "likes", "inV": 4208 }'

<u>Response:</u>
{"requestId":"f8a5627f-b326-43cd-940d-230b079d079c","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":"215r7v-388-3yt-38w","label":"likes","type":"edge","inVLabel":"interest","outVLabel":"person","inV":4208,"outV":4184}],"meta":{}}}

Similarly, create edges for others according to the diagram.

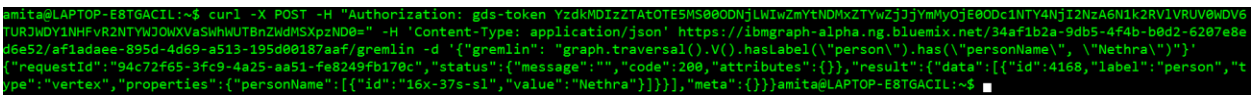
- Querying the graph data:

1. Get the properties of person whose name is “Nethra”

i. <u>Using curl:</u>

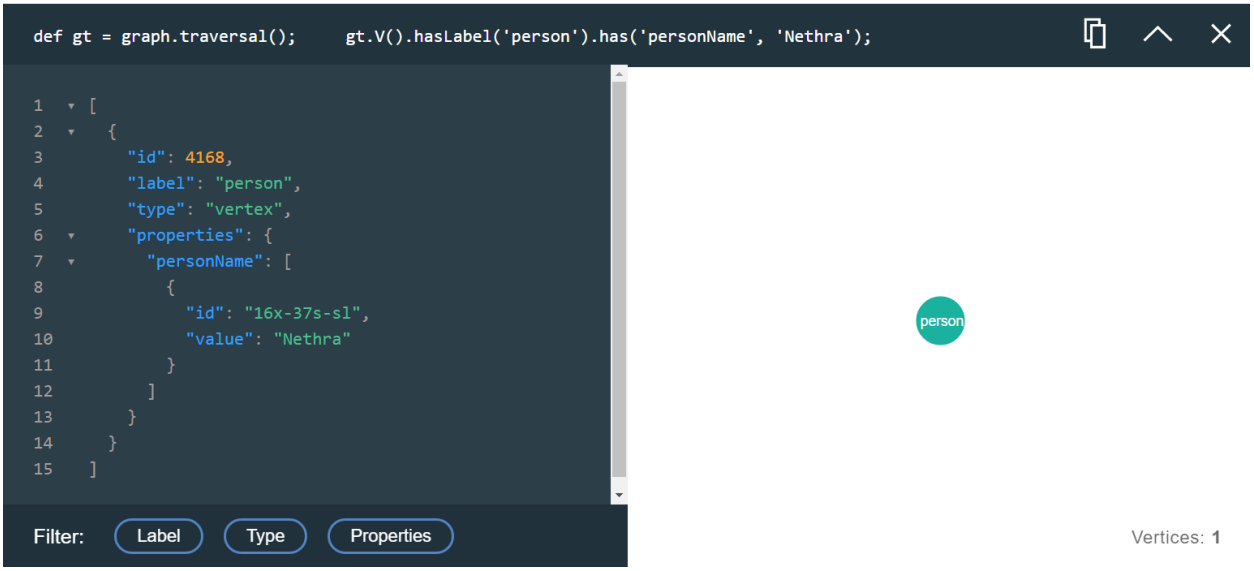
amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/gremlin -d '{"gremlin": "graph.traversal().V().hasLabel(\"person\").has(\"personName\", \"Nethra\")"}'

<u>Response:</u>
{"requestId":"94c72f65-3fc9-4a25-aa51-fe8249fb170c","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4168,"label":"person","type":"vertex","properties":{"personName":[{"id":"16x-37s-sl","value":"Nethra"}]}}],"meta":{}}}



ii. <u>Using Gremlin:</u>

def gt = graph.traversal();
gt.V().hasLabel('person').has('personName', 'Nethra');

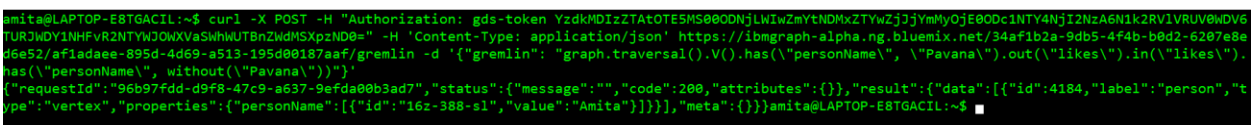


2. Find each Vertex that has a personName property set to 'Pavana', Find all connected vertices that are connected via an outward 'likes' edge (i.e., Pavana's interests), Find all vertices that are connected to these interests via an inward 'likes' edge (i.e., other people who also like Pavana's interests), Of these vertices, find all that have a personName (i.e., are a person), but where that name is not Pavana

   i. <u>Using curl:</u>

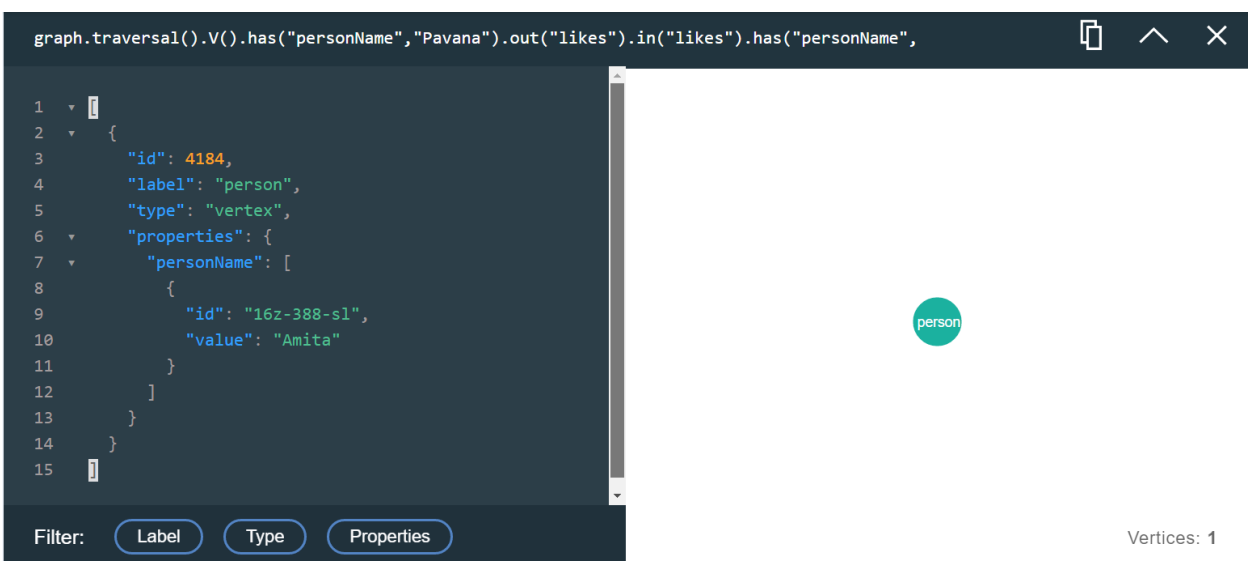
amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/gremlin -d '{"gremlin": "graph.traversal().V().has(\"personName\", \"Pavana\").out(\"likes\").in(\"likes\").has(\"personName\", without(\"Pavana\"))"}'

<u>Response:</u>
{"requestId":"96b97fdd-d9f8-47c9-a637-9efda00b3ad7","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4184,"label":"person","type":"vertex","properties":{"personName":[{"id":"16z-388-sl","value":"Amita"}]}}],"meta":{}}}



   ii. <u>Using Gremlin:</u>

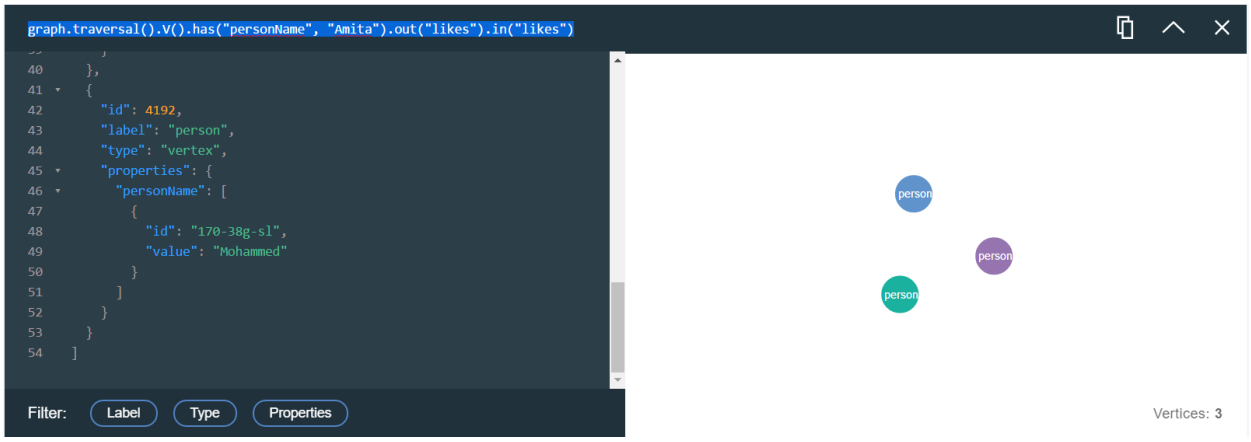
graph.traversal().V().has("personName","Pavana").out("likes").in("likes").has("personName", without("Pavana"))



3. Find persons who shares similar interests as Amita – (Amita, Pavana, Mohammed)

   i. <u>Using Gremlin:</u>

graph.traversal().V().has("personName", "Amita").out("likes").in("likes")

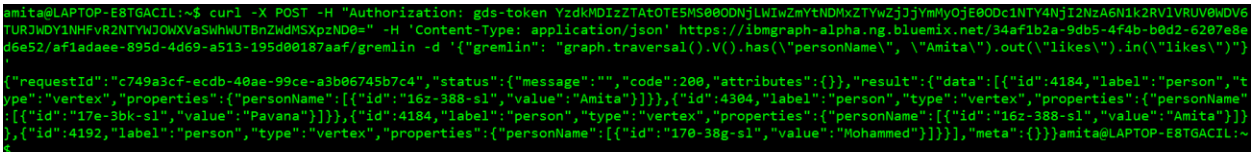


   ii. <u>Using curl:</u>

amita@LAPTOP-E8TGACIL:~$ curl -X POST -H "Authorization: gds-token YzdkMDIzZTAtOTE5MS00ODNjLWIwZmYtNDMxZTYwZjJjYmMyOjE0ODc1NTY4NjI2NzA6N1k2RVlVRUV0WDV6TURJWDY1NHFvR2NTYWJOWXVaSWhWUTBnZWdMSXpzND0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52/af1adaee-895d-4d69-a513-195d00187aaf/gremlin -d '{"gremlin": "graph.traversal().V().has(\"personName\", \"Amita\").out(\"likes\").in(\"likes\")"}'

<u>Response:</u>

{"requestId":"c749a3cf-ecdb-40ae-99ce-a3b06745b7c4","status":{"message":"","code":200,"attributes":{}},"result":{"data":[{"id":4184,"label":"person","type":"vertex","properties":{"personName":[{"id":"16z-388-sl","value":"Amita"}]}},{"id":4304,"label":"person","type":"vertex","properties":{"personName":[{"id":"17e-3bk-sl","value":"Pavana"}]}},{"id":4184,"label":"person","type":"vertex","properties":{"personName":[{"id":"16z-388-sl","value":"Amita"}]}},{"id":4192,"label":"person","type":"vertex","properties":{"personName":[{"id":"170-38g-sl","value":"Mohammed"}]}}],"meta":{}}}



- Deleting a graph

amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs/62ec6492-054c-4d85-8b97-d971188a1bbb" -X DELETE -u "$username:$password"

Response:
{"data":{}}

Verify using GET that the graph is deleted.

amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs" -X GET -u "$username:$password"

Response:
{"graphs":["14cc0f3d-0fa2-479c-a1ab-ccdb8a88c873","67d23268-e5e8-4e5d-8377-dd68de5c6360","g"]}

```
amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs/62ec6492-054c-4d85-8b97-d971188a1bbb" -X DELETE -u "$username:$password"
{"data":{}}amita@LAPTOP-E8TGACIL:~$
amita@LAPTOP-E8TGACIL:~$ curl "$BASE_URL/_graphs" -X GET -u "$username:$password"
{"graphs":["14cc0f3d-0fa2-479c-a1ab-ccdb8a88c873","67d23268-e5e8-4e5d-8377-dd68de5c6360","g"]}amita@LAPTOP-E8TGACIL:~$
```

#Content of explainPlan.txt

```
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool



******************** EXPLAIN INSTANCE ********************

DB2_VERSION:       11.01.1
FORMATTED ON DB:   HAROON
SOURCE_NAME:       SQLC2O26
SOURCE_SCHEMA:     NULLID
SOURCE_VERSION:
EXPLAIN_TIME:      2017-02-21-13.10.44.768000
EXPLAIN_REQUESTER: HAROON

Database Context:
----------------
        Parallelism:          None
        CPU Speed:            4.251098e-007
        Comm Speed:           0
        Buffer Pool size:     250
        Sort Heap size:       256
        Database Heap size:   600
        Lock List size:       4096
        Maximum Lock List:    22
        Average Applications: 1
        Locks Available:      28835
```

```
Package Context:
---------------
        SQL Type:          Dynamic
        Optimization Level: 5
        Blocking:          Block All Cursors
        Isolation Level:    Cursor Stability



---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:       2
        QUERYTAG:      CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     1

Original Statement:
------------------
select
  workdept,
  sum(salary) as GROUP_SALARY
from
  emp
where
  salary > 70000
group by
  workdept
order by
  GROUP_SALARY desc
```

```
Optimized Statement:
-------------------
SELECT
  Q3.WORKDEPT AS "WORKDEPT",
  Q3.$C1 AS "GROUP_SALARY"
FROM
  (SELECT
     Q2.WORKDEPT,
     SUM(Q2.SALARY)
   FROM
     (SELECT
        Q1.WORKDEPT,
        Q1.SALARY
      FROM
        HAROON.EMPLOYEE AS Q1
      WHERE
        (70000 < Q1.SALARY)
     ) AS Q2
   GROUP BY
     Q2.WORKDEPT
  ) AS Q3
ORDER BY
  Q3.$C1 DESC

Access Plan:
-----------
        Total Cost:          6.87992
        Query Degree:        1
```

Rows
RETURN
(1)
Cost
I/O
|
8
TBSCAN
(2)
6.87992
1
|
8
SORT
(3)
6.87842
1
|
8
GRPBY
(4)
6.87288
1
|
8.91871
TBSCAN
(5)
6.87176
1
|
8.91871

```
    SORT
    (   6)
    6.87016
      1
     |
    8.91871
    TBSCAN
    (   7)
    6.86593
      1
     |
     42
 TABLE: HAROON
   EMPLOYEE
     Q1
```

Extended Diagnostic Information:

Diagnostic Identifier: 1
Diagnostic Details: EXP0073W The following MQT or statistical view was not eligible because one or more data filtering predicates from the query could not be matched with the MQT: "HAROON "."ADEFUSR".
Diagnostic Identifier: 2
Diagnostic Details: EXP0148W The following MQT or statistical view was considered in query matching: "HAROON "."ADEFUSR".

Plan Details:

```
-------------

    1) RETURN: (Return Result)
        Cumulative Total Cost:          6.87992
        Cumulative CPU Cost:            176236
        Cumulative I/O Cost:            1
        Cumulative Re-Total Cost:       0.0453515
        Cumulative Re-CPU Cost:         106682
        Cumulative Re-I/O Cost:         0
        Cumulative First Row Cost:      6.87907
        Estimated Bufferpool Buffers:   0

        Arguments:
        ---------
        BLDLEVEL: (Build level)
            DB2 v11.1.1010.160 : s1612051900
        HEAPUSE : (Maximum Statement Heap Usage)
            112 Pages
        PLANID  : (Access plan identifier)
            a327d46dee0d7ba6
        PREPTIME: (Statement prepare time)
                78 milliseconds
        SEMEVID : (Semantic environment identifier)
            8bb60f2a8460e3e1
        STMTHEAP: (Statement heap size)
            8192
        STMTID  : (Normalized statement identifier)
            d39f4bf6af7c3325

        Input Streams:
```

```
-------------
        7) From Operator #2

                Estimated number of rows:    8
                Number of columns:           2
                Subquery predicate ID:                Not Applicable

                Column Names:
                ------------
                +Q4.GROUP_SALARY(D)+Q4.WORKDEPT


2) TBSCAN: (Table Scan)
        Cumulative Total Cost:              6.87992
        Cumulative CPU Cost:                176236
        Cumulative I/O Cost:          1
        Cumulative Re-Total Cost:     0.0453515
        Cumulative Re-CPU Cost:       106682
        Cumulative Re-I/O Cost:       0
        Cumulative First Row Cost:    6.87907
        Estimated Bufferpool Buffers:       0

        Arguments:
        ---------
        MAXPAGES: (Maximum pages for prefetch)
                ALL
        PREFETCH: (Type of Prefetch)
                NONE
        SCANDIR : (Scan Direction)
                FORWARD
        SPEED   : (Assumed speed of scan, in sharing structures)
```

```
        SLOW
THROTTLE: (Scan may be throttled, for scan sharing)
        FALSE
VISIBLE : (May be included in scan sharing structures)
        FALSE
WRAPPING: (Scan may start anywhere and wrap)
        FALSE

Input Streams:
-------------
        6) From Operator #3

                Estimated number of rows:    8
                Number of columns:           2
                Subquery predicate ID:               Not Applicable

                Column Names:
                ------------
                +Q3.$C1(D)+Q3.WORKDEPT


Output Streams:
--------------
        7) To Operator #1

                Estimated number of rows:    8
                Number of columns:           2
                Subquery predicate ID:               Not Applicable

                Column Names:
                ------------
```

+Q4.GROUP_SALARY(D)+Q4.WORKDEPT

3) SORT : (Sort)

Cumulative Total Cost: 6.87842

Cumulative CPU Cost: 172699

Cumulative I/O Cost: 1

Cumulative Re-Total Cost: 0.0438478

Cumulative Re-CPU Cost: 103145

Cumulative Re-I/O Cost: 0

Cumulative First Row Cost: 6.87842

Estimated Bufferpool Buffers: 0

Arguments:

DUPLWARN: (Duplicates Warning flag)

FALSE

KEYS : (Key cardinality)

8

NUMROWS : (Estimated number of rows)

8

ROWWIDTH: (Estimated width of rows)

20.000000

SORTKEY : (Sort Key column)

1: Q3.$C1(D)

TEMPSIZE: (Temporary Table Page Size)

8192

UNIQUE : (Uniqueness required flag)

FALSE

Input Streams:

---

5) From Operator #4

Estimated number of rows: 8
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:

+Q3.$C1+Q3.WORKDEPT

Output Streams:
---

6) To Operator #2

Estimated number of rows: 8
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:

+Q3.$C1(D)+Q3.WORKDEPT

4) GRPBY : (Group By)

Cumulative Total Cost: 6.87288
Cumulative CPU Cost: 159682
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0421453
Cumulative Re-CPU Cost: 99139.8

```
Cumulative Re-I/O Cost:        0
Cumulative First Row Cost:   6.87095
Estimated Bufferpool Buffers:          0

Arguments:
---------
AGGMODE : (Aggregation Mode)
        COMPLETE
GROUPBYC: (Group By columns)
        TRUE
GROUPBYN: (Number of Group By columns)
        1
GROUPBYR: (Group By requirement)
        1: Q2.WORKDEPT
ONEFETCH: (One Fetch flag)
        FALSE

Input Streams:
-------------
        4) From Operator #5

                Estimated number of rows:    8.91871
                Number of columns:           2
                Subquery predicate ID:                Not Applicable

                Column Names:
                ------------
                +Q2.WORKDEPT(A)+Q2.SALARY


Output Streams:
```

```
--------------
        5) To Operator #3

                Estimated number of rows:    8
                Number of columns:           2
                Subquery predicate ID:                Not Applicable

                Column Names:
                ------------
                +Q3.$C1+Q3.WORKDEPT


5) TBSCAN: (Table Scan)
        Cumulative Total Cost:              6.87176
        Cumulative CPU Cost:                157045
        Cumulative I/O Cost:          1
        Cumulative Re-Total Cost:     0.0410242
        Cumulative Re-CPU Cost:       96502.7
        Cumulative Re-I/O Cost:       0
        Cumulative First Row Cost:    6.87081
        Estimated Bufferpool Buffers:       0

        Arguments:
        ---------
        MAXPAGES: (Maximum pages for prefetch)
                ALL
        PREFETCH: (Type of Prefetch)
                NONE
        SCANDIR : (Scan Direction)
                FORWARD
        SPEED   : (Assumed speed of scan, in sharing structures)
```

```
        SLOW
THROTTLE: (Scan may be throttled, for scan sharing)
        FALSE
VISIBLE : (May be included in scan sharing structures)
        FALSE
WRAPPING: (Scan may start anywhere and wrap)
        FALSE

Input Streams:
-------------
        3) From Operator #6

                Estimated number of rows:    8.91871
                Number of columns:           2
                Subquery predicate ID:               Not Applicable

                Column Names:
                ------------
                +Q2.WORKDEPT(A)+Q2.SALARY


Output Streams:
--------------
        4) To Operator #4

                Estimated number of rows:    8.91871
                Number of columns:           2
                Subquery predicate ID:               Not Applicable

                Column Names:
                ------------
```

+Q2.WORKDEPT(A)+Q2.SALARY

6) SORT : (Sort)

Cumulative Total Cost: 6.87016

Cumulative CPU Cost: 153278

Cumulative I/O Cost: 1

Cumulative Re-Total Cost: 0.039423

Cumulative Re-CPU Cost: 92736

Cumulative Re-I/O Cost: 0

Cumulative First Row Cost: 6.87016

Estimated Bufferpool Buffers: 1

Arguments:

DUPLWARN: (Duplicates Warning flag)

FALSE

KEYS : (Key cardinality)

8

NUMROWS : (Estimated number of rows)

9

ROWWIDTH: (Estimated width of rows)

16.000000

SORTKEY : (Sort Key column)

1: Q2.WORKDEPT(A)

TEMPSIZE: (Temporary Table Page Size)

8192

UNIQUE : (Uniqueness required flag)

FALSE

Input Streams:

---

2) From Operator #7

Estimated number of rows: 8.91871
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:

+Q2.SALARY+Q2.WORKDEPT

Output Streams:
---

3) To Operator #5

Estimated number of rows: 8.91871
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:

+Q2.WORKDEPT(A)+Q2.SALARY

7) TBSCAN: (Table Scan)

Cumulative Total Cost: 6.86593
Cumulative CPU Cost: 143329
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.039423
Cumulative Re-CPU Cost: 92736

```
Cumulative Re-I/O Cost:        0
Cumulative First Row Cost:   6.83083
Estimated Bufferpool Buffers:          1

Arguments:
---------
CUR_COMM: (Currently Committed)
        TRUE
LCKAVOID: (Lock Avoidance)
        TRUE
MAXPAGES: (Maximum pages for prefetch)
        ALL
PREFETCH: (Type of Prefetch)
        NONE
ROWLOCK : (Row Lock intent)
        SHARE (CS/RS)
SCANDIR : (Scan Direction)
        FORWARD
SKIP_INS: (Skip Inserted Rows)
        TRUE
SPEED   : (Assumed speed of scan, in sharing structures)
        FAST
TABLOCK : (Table Lock intent)
        INTENT SHARE
TBISOLVL: (Table access Isolation Level)
        CURSOR STABILITY
THROTTLE: (Scan may be throttled, for scan sharing)
        TRUE
VISIBLE : (May be included in scan sharing structures)
        TRUE
WRAPPING: (Scan may start anywhere and wrap)
```

```
        TRUE

Predicates:
----------
3) Sargable Predicate,
        Comparison Operator:                Less Than (<)
        Subquery Input Required:    No
        Filter Factor:              0.21235

        Predicate Text:
        --------------
        (70000 < Q1.SALARY)



Input Streams:
-------------
        1) From Object HAROON.EMPLOYEE

                Estimated number of rows:   42
                Number of columns:          3
                Subquery predicate ID:              Not Applicable

                Column Names:
                ------------
                +Q1.$RID$+Q1.SALARY+Q1.WORKDEPT


Output Streams:
--------------
        2) To Operator #6
```

```
Estimated number of rows:    8.91871
Number of columns:           2
Subquery predicate ID:               Not Applicable

Column Names:
------------
+Q2.SALARY+Q2.WORKDEPT


Objects Used in Access Plan:
---------------------------

    Schema: HAROON
    Name:  EMP
    Type:   Alias (reference only)

    Schema: HAROON
    Name:  ADEFUSR
    Type:   Materialized View (reference only)

    Schema: HAROON
    Name:  EMPLOYEE
    Type:   Table
                Time of creation:              2017-02-20-18.30.02.482001
                Last statistics update:  2017-02-20-18.38.09.092000
                Number of columns:             14
                Number of rows:                42
                Width of rows:                         99
                Number of buffer pool pages:           1
                Number of data partitions:     1
```

Distinct row values:	No
Tablespace name:	USERSPACE1
Tablespace overhead:	6.725000
Tablespace transfer rate:	0.080000
Source for statistics:	Single Node
Prefetch page count:	32
Container extent page count:	32
Table overflow record count:	0
Table Active Blocks:	-1
Average Row Compression Ratio:	0
Percentage Rows Compressed:	0
Average Compressed Row Size:	0

IBM Bluemix JAVA Application

```
package ibmbluemix;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.ParseException;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.entity.mime.content.FileBody;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import org.apache.commons.codec.binary.Base64;
import org.apache.http.HttpHost;
import org.apache.http.HttpRequest;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpUriRequest;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HttpContext;
import org.apache.wink.json4j.JSON;
import org.apache.wink.json4j.JSONArray;
import org.apache.wink.json4j.JSONException;
import org.apache.wink.json4j.JSONObject;
import org.apache.wink.json4j.JSONArtifact;
```

```
/**
 *
 * @author Amita Kamat
 */


public class IBMBluemix {


  public static void main(String[] args) {
      String apiURL = null;
      String username = null;
      String password = null;
      String baseURL = null;
      String basicAuth = null;

      try{
         // NOTE : Enter your credentials below before running the program
         apiURL = "Enter your API URL here";
         username = "Enter your username here";
         password = "Enter your password here";
         baseURL = apiURL.substring(0, apiURL.length() - 2);
         byte[] userpass = (username + ":" + password).getBytes();
         byte[] encoding = Base64.encodeBase64(userpass);
         basicAuth = "Basic " + new String(encoding);

         ArrayList<String> token = getToken(baseURL, basicAuth);
         String dbURL = createGraph(baseURL, token.get(1));
         createSchema(dbURL, token.get(1));
         HashMap vertices = createVertices(dbURL, token.get(1));
         createEdges(vertices, dbURL, token.get(1));
      }
      catch(Exception ex)
      {
         System.out.println(ex.getMessage());
      }
   }

   /**
    * @param baseURL BASE URL of the graph service
    * @param basicAuth basic authorization
    * @return Session token
    */
```

```
private static ArrayList<String> getToken(String baseURL, String basicAuth){
    ArrayList<String> token = new ArrayList<String>();
    try{
        String gdsToken;
        String gdsTokenAuth = null;
        HttpClient client = HttpClients.createDefault();
        HttpGet httpGet = new HttpGet(baseURL + "/_session");
        httpGet.setHeader("Authorization", basicAuth);
        HttpResponse httpResponse = client.execute(httpGet);
        HttpEntity httpEntity = httpResponse.getEntity();
        String content = EntityUtils.toString(httpEntity);
        EntityUtils.consume(httpEntity);
        JSONObject jsonContent = new JSONObject(content);
        gdsToken = jsonContent.getString("gds-token");
        token.add(gdsToken);
        token.add("gds-token " + gdsToken);
    }
    catch(Exception ex){
        System.out.println(ex.getMessage());
    }

    return token;
}

/**
 * Creates graph in IBM bluemix
 * @param baseURL Base url of graph service
 * @param gdsTokenAuth session token
 * @return dbURL from response
 */
private static String createGraph(String baseURL, String gdsTokenAuth){

    String apiURL = "";
    HttpClient client = HttpClients.createDefault();
    String graphID = UUID.randomUUID().toString().replaceAll("-", "");
    String postURLGraph = baseURL + "/_graphs/" + graphID;
    HttpPost httpPostGraph = new HttpPost(postURLGraph);
    httpPostGraph.setHeader("Authorization", gdsTokenAuth);

    try{
        HttpResponse httpResponseGraph = client.execute(httpPostGraph);
        HttpEntity httpEntityGraph = httpResponseGraph.getEntity();
        String contentGraph = EntityUtils.toString(httpEntityGraph);
        EntityUtils.consume(httpEntityGraph);
```

```
            JSONObject jsonContentGraph = new JSONObject(contentGraph);
            System.out.println("response from creating graph" + jsonContentGraph.toString());
            // Update apiURL
            apiURL = jsonContentGraph.getString("dbUrl");
            System.out.println("Graph created with id:" + jsonContentGraph.getString("graphId"));
        }
        catch(Exception ex)
        {
            System.out.println("Graph cannot be created . Error : " + ex.getMessage());
        }

        return apiURL;
    }

    /**
     * Get graph schema from a file
     * @return graph schema
     */
    private static JSONArtifact getGraphSchema(){
        JSONArray jsonArray = new JSONArray();
        JSON json = new JSON();
        JSONArtifact schema = null;
        try{
            schema = json.parse(new FileReader("C:/Users/Amita
Kamat/Documents/NetBeansProjects/IBMBluemix/src/ibmbluemix/graph-schema.json"));
        }
        catch(Exception ex){
            System.out.println("Cannot fetch graph schema. Error: " + ex.getMessage());
        }

        return schema;
    }

    /**
     * Create graph schema
     * @param apiURL API URL of IBM BLUEMIX graph service
     * @param gdsTokenAuth session token
     */
    private static void createSchema(String apiURL, String gdsTokenAuth){
        JSONArtifact postData = getGraphSchema();
        HttpClient client = HttpClients.createDefault();
        HttpPost httpPost = new HttpPost(apiURL + "/schema");
        httpPost.setHeader("Authorization", gdsTokenAuth);
        httpPost.setHeader("Content-Type", "application/json");
```

```
        httpPost.setHeader("Accept", "application/json");
        StringEntity strEnt = new StringEntity(postData.toString(), ContentType.APPLICATION_JSON);
        httpPost.setEntity(strEnt);

        try{
            HttpResponse httpResponse = client.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            String content = EntityUtils.toString(httpEntity);
            EntityUtils.consume(httpEntity);
            JSONObject jsonContent = new JSONObject(content);
            JSONObject result = jsonContent.getJSONObject("result");
            JSONArray data = (result.getJSONArray("data"));
            if (data.length() > 0) {
                JSONObject response = data.getJSONObject(0);
                System.out.println("Response from creating schema" + response);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Could not create schema. Error: " + ex.getMessage());
        }
    }

    /**
     * Create vertices/nodes for the graph
     * @param apiURL API URL of IBM BLUEMIX graph
     * @param gdsTokenAuth session token
     * @return
     */
    private static HashMap createVertices(String apiURL, String gdsTokenAuth)
    {
        String v1 = null;
        HttpClient client = HttpClients.createDefault();
        String postURL = apiURL + "/vertices";
        JSONObject postData = new JSONObject();
        String[] personInterestNames = {"Amita", "Nethra", "Pavana", "Mohammed", "Siddharth", "Books",
"Films", "Badminton", "Music"};
        HashMap personID = new HashMap();

        try{
        // We must nest our vertex/edge indexes inside a properties {} object
        JSONObject vertexIndices = new JSONObject();

            for(int i=0; i< personInterestNames.length; i++){
```

```
            if(i<5){
                vertexIndices.put("personName", personInterestNames[i]);
                postData.put("properties", vertexIndices);
                postData.put("label", "person");
            }
            else{
                vertexIndices.put("interestName", personInterestNames[i]);
                postData.put("properties", vertexIndices);
                postData.put("label", "interests");
            }
            HttpPost httpPost = new HttpPost(postURL);
            StringEntity strEnt = new StringEntity(postData.toString(),
ContentType.APPLICATION_JSON);
            httpPost.setEntity(strEnt);
            httpPost.setHeader("Authorization", gdsTokenAuth);
            HttpResponse httpResponse = client.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            String content = EntityUtils.toString(httpEntity);
            EntityUtils.consume(httpEntity);
            JSONObject jsonContent = new JSONObject(content);
            JSONObject result = jsonContent.getJSONObject("result");
            JSONArray data = result.getJSONArray("data");
            if (data.length() > 0) {
                JSONObject response = data.getJSONObject(0);
                v1 = response.getString("id");
                personID.put(personInterestNames[i], Integer.parseInt(v1));
                System.out.println("Vertices " + i + ": " + personInterestNames[i] + " with ID " + v1);
            }
        }
    }
    catch(Exception ex){
        System.out.println("Vertices could not be created. Error : " + ex.getMessage());
    }

    return personID;
}

/**
 * Create edges for the graph
 * @param vertices List of vertices with IDs
 * @param apiURL API URL of graph service
 * @param gdsTokenAuth session token
 */
```

```
    private static void createEdges(HashMap vertices, String apiURL, String gdsTokenAuth){
        String e1 = null;
        JSONObject postData = new JSONObject();
        HttpClient client = HttpClients.createDefault();
        int[] out = new int[]{ (Integer)vertices.get("Amita"), (Integer)vertices.get("Siddharth"),
(Integer)vertices.get("Amita"), (Integer)vertices.get("Nethra"), (Integer)vertices.get("Nethra"),
(Integer)vertices.get("Mohammed"), (Integer)vertices.get("Mohammed"), (Integer)vertices.get("Pavana"),
(Integer)vertices.get("Amita"), (Integer)vertices.get("Amita"), (Integer)vertices.get("Nethra"),
(Integer)vertices.get("Pavana"), (Integer)vertices.get("Pavana"), (Integer)vertices.get("Mohammed")};
        int[] in = new int[]{ (Integer)vertices.get("Siddharth"), (Integer)vertices.get("Amita"),
(Integer)vertices.get("Nethra"), (Integer)vertices.get("Amita"), (Integer)vertices.get("Mohammed"),
(Integer)vertices.get("Nethra"), (Integer)vertices.get("Pavana"), (Integer)vertices.get("Mohammed"),
(Integer)vertices.get("Badminton"), (Integer)vertices.get("Films"), (Integer)vertices.get("Books"),
(Integer)vertices.get("Badminton"), (Integer)vertices.get("Music"), (Integer)vertices.get("Films")};
        String label = null;
        try{
            for(int i=0; i< out.length; i++){
                postData.put("outV", out[i]);
                postData.put("inV", in[i]);

                if(i<8){
                    postData.put("label", "friendsWith");
                    label = "friendsWith";
                }
                else{
                    postData.put("label", "likes");
                    label = "likes";
                }

                HttpPost httpPost = new HttpPost(apiURL + "/edges");
                httpPost.setHeader("Authorization", gdsTokenAuth);
                StringEntity strEnt = new StringEntity(postData.toString(),
ContentType.APPLICATION_JSON);
                httpPost.setEntity(strEnt);
                HttpResponse httpResponse = client.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                String content = EntityUtils.toString(httpEntity);
                EntityUtils.consume(httpEntity);
                JSONObject jsonContent = new JSONObject(content);
                JSONObject result = jsonContent.getJSONObject("result");
                JSONArray data = result.getJSONArray("data");

                if (data.length() > 0) {
                    JSONObject response = data.getJSONObject(0);
```

```
                e1 = response.getString("id");
                System.out.println("Edge created from " + out[i] + " to " + in[i] + " with label " + label + "
having ID " + e1 );
            }
        }
    }
    catch(Exception ex){
        System.out.println("Could not craete edges for graph. Error : " + ex.getMessage());
    }
  }

}
```

Program Output:

```
Output - IBMBluemix (run)
run:
response from creating graph{"graphId":"5426d3ba561b4d3c81105a014b577b6c","dbUrl":"https:\/\/ibmgraph-alpha.ng.bluemix.net\/34af1b2a-9db5-4f4b-b0d2-6207e8ed6e52\/5426d3ba561b4d3c81105a014b577b6c"}
Graph created with id:5426d3ba561b4d3c81105a014b577b6c
Response from creating schema{"edgeLabels":[{"multiplicity":"MULTI","directed":true,"name":"likes"},{"multiplicity":"MULTI","directed":true,"name":"friendsWith"}],"edgeIndexes":[],"vertexLabels":[{"name":"person"},{"name":"i
nterests"}],"propertyKeys":[{"dataType":"String","name":"personName","cardinality":"SINGLE"},{"dataType":"String","name":"interestName","cardinality":"SINGLE"}],"vertexIndexes":[{"composite":true,"unique":false,"name":"vByPe
rsonName","requiresReindex":false,"type":"vertex","propertyKeys":["personName"]},{"composite":true,"unique":false,"name":"vByInterestName","requiresReindex":false,"type":"vertex","propertyKeys":["interestName"]}]}
Vertices 0: Amita with ID 4232
Vertices 1: Nethra with ID 4264
Vertices 2: Pavana with ID 4224
Vertices 3: Mohammed with ID 40964232
Vertices 4: Siddharth with ID 4272
Vertices 5: Books with ID 40968328
Vertices 6: Films with ID 8360
Vertices 7: Badminton with ID 4168
Vertices 8: Music with ID 4144
Edge created from 4232 to 4272 with label friendsWith having ID oe041-39k-4r9-3ao
Edge created from 4272 to 4232 with label friendsWith having ID odxcm-3ao-4r9-39k
Edge created from 4232 to 4264 with label friendsWith having ID oe0i9-39k-4r9-3ag
Edge created from 4264 to 4232 with label friendsWith having ID 36d-3ag-4r9-39k
Edge created from 4264 to 40964232 with label friendsWith having ID odxcl-3ag-4r9-oe07c
Edge created from 40964232 to 4264 with label friendsWith having ID 11d-oe07c-4r9-3ag
Edge created from 40964232 to 4224 with label friendsWith having ID oe0wh-oe07c-4r9-39c
Edge created from 4224 to 40964232 with label friendsWith having ID odxcg-39c-4r9-oe07c
Edge created from 4232 to 4168 with label likes having ID 1zl-39k-3yt-37s
Edge created from 4232 to 8360 with label likes having ID 1crua9-39k-3yt-6g8
Edge created from 4264 to 40968328 with label likes having ID odxqt-3ag-3yt-oe3d4
Edge created from 4224 to 4168 with label likes having ID 1crua8-39c-3yt-37s
Edge created from 4224 to 4144 with label likes having ID 1cruog-39c-3yt-374
Edge created from 40964232 to 8360 with label likes having ID 1cruoh-oe07c-3yt-6g8
BUILD SUCCESSFUL (total time: 17 seconds)
```

graph-schema.json

```
{
  "propertyKeys": [
    {"name": "personName", "dataType": "String", "cardinality":
"SINGLE"},
    {"name": "interestName", "dataType": "String", "cardinality":
"SINGLE"}
  ],
  "vertexLabels": [
    {"name": "person"},
    {"name": "interests"}
  ],
  "edgeLabels": [
    { "name": "likes" },
    { "name": "friendsWith" }
  ],
  "vertexIndexes": [
    {"name": "vByPersonName", "propertyKeys": ["personName"],
"composite": true, "unique": false},
    {"name": "vByInterestName", "propertyKeys": ["interestName"],
"composite": true, "unique": false}
  ]
}
```