

## Report

### Research and Development [1]

#### Task 1: Build Distributed Database

- The dataset once stored in the database, the data can be fragmented in a horizontal manner. The details of particular attribute can be derived with the help of base table. Example: If we consider our table “order\_payments” as base table from the remote database.

```
SQL> SELECT * FROM purchaseInfo.order_payments;
```

	A	B	C	D	E
1	order_id	payment_sequential	payment_type	payment_installments	payment_value
2	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
3	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
4	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
5	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
6	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45
7	298fcd1f73eb413e4d26d01b25bc1cd	1	credit_card	2	96.12
8	771ee386b001f06208a7419e4fc1bbd7	1	credit_card	1	81.16
9	3d7239c394a212faae122962df514ac7	1	credit_card	3	51.84
10	1f78449c87a54faf9e96e88ba1491fa9	1	credit_card	6	341.09
11	0573b5e23cbd798006520e1d5b4c6714	1	boleto	1	51.95
12	d88e0d5fa41661ce03cf6cf336527646	1	credit_card	8	188.73
13	2480f727e869fdeb397244a21b721b67	1	credit_card	1	141.9
14	616105c9352a9668c38303ad44e056cd	1	credit_card	1	75.78
15	cf95215a722f3ebf29e6bbab87a29e61	1	credit_card	5	102.66
16	769214176682788a92801d8907fa1b40	1	credit_card	4	105.28
17	12e5cfe0e4716b59afb0e0f4a3bd6570	1	credit_card	10	157.45
18	61059985a6fc0ad64e95d9944caacdad	1	credit_card	1	132.04
19	79da3f5fe31ad1e454f06f95dc032ad5	1	credit_card	1	98.94
20	8ac09207f415d55acff302df7d6a895c	1	credit_card	4	244.15

Here, when the user needs to access the specific payment\_type and list the order\_id accordingly or payment\_installments which corresponds to it specific payment\_value, it is difficult for the user to retrieve the required data quickly. However, we can fragment the table in a horizontal manner by considering a requirement based on payment\_type.

```
SQL> SELECT * FROM purchaseInfo.order_payments where  
payment_type="credit_card";
```

The data is now horizontally fragmented where the order\_payments table is pointing to a specific payment\_type “credit\_card”. The user could get the other attributes like order\_id associated with the “credit\_card”. Refer Image1.png

We can also fragment the data horizontally in other ways by adding more conditions to it. Example: if we consider user wants to access order\_payments table and he wants retrieve fields with payment\_type as “credit\_card” and associated payment\_installments are “8”. Refer Image2.png

```
SQL> SELECT * FROM purchaseInfo.order_payments where
payment_type="credit_card" and payment_installments=8;
```

Image1

	A	B	C	D	E	I
1	order_id	payment_sequential	payment_type	payment_installments	payment_value	
2	b81ef226f3fe1789b1e8b2acac839d17		1 credit_card	8	99.33	
3	a9810da82917af2d9aefd1278f1dcfa0		1 credit_card	1	24.39	
4	25e8ea4e93396b6fa0d3dd708e76c1bd		1 credit_card	1	65.71	
5	ba78997921bbcdc1373bb41e913ab953		1 credit_card	8	107.78	
6	42fdf880ba16b47b59251dd489d4441a		1 credit_card	2	128.45	
7	298fcd1f73eb413e4d26d01b25bc1cd		1 credit_card	2	96.12	
8	771ee386b001f06208a7419e4fc1bbd7		1 credit_card	1	81.16	
9	3d7239c394a212faae122962df514ac7		1 credit_card	3	51.84	
10	1f78449c87a54faf9e96e88ba1491fa9		1 credit_card	6	341.09	
11	d88e0d5fa41661ce03cf6cf336527646		1 credit_card	8	188.73	
12	2480f727e869fdeb397244a21b721b67		1 credit_card	1	141.9	
13	616105c9352a9668c38303ad44e056cd		1 credit_card	1	75.78	
14	cf95215a722f3ebf29e6bbab87a29e61		1 credit_card	5	102.66	
15	769214176682788a92801d8907fa1b40		1 credit_card	4	105.28	
16	12e5cfe0e4716b59afb0e0f4a3bd6570		1 credit_card	10	157.45	
17	61059985a6fc0ad64e95d9944caacdad		1 credit_card	1	132.04	
18	79da3f5fe31ad1e454f06f95dc032ad5		1 credit_card	1	98.94	
19	8ac09207f415d55acff302df7d6a895c		1 credit_card	4	244.15	
20	b2349a3f20dfbeef62e7b31baa22f84b		1 credit_card	3	136.71	

Image2

	A	B	C	D	E
1	order_id	payment_sequential	payment_type	payment_installments	payment_value
2	b81ef226f3fe1789b1e8b2acac839d17		1 credit_card	8	99.33
3	ba78997921bbcdc1373bb41e913ab953		1 credit_card	8	107.78
4	d88e0d5fa41661ce03cf6cf336527646		1 credit_card	8	188.73
5	d3e774a185c0b1b2286ffd6c70abe2e6		1 credit_card	8	117.74
6					
7					
8					

- Before importing the tables into the local database, data cleaning has been done in the table “olist\_geolocation\_dataset” by changing the values of the attribute “geolocation\_city” into human readable format.

D	D
geolocation_city	geolocation_city
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo
sao paulo	sao paulo

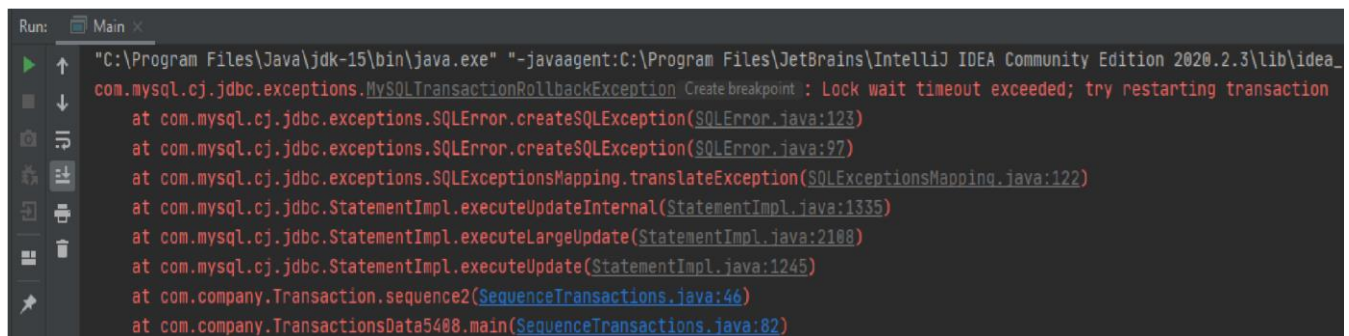
- Data dictionary of Data5408 defines the tables and fields from the dataset provided it can be created using the most simple and easy way that is Excel Spreadsheet document. The drawback

for this being not user friendly, difficulty to attain readability and high maintenance. This spreadsheet can be placed in the database folder where all the information regarding database is present. Alternative, Global Data Catalogue can be achieved by the process of reverse engineering of the schema then integrate it into a newly created model, further description of the tables can be given and finally extracting the data catalogue to a CSV file or a HTML page.

## Task 2: Perform Distributed Concurrent Transaction

- After importing the relevant tables in remote database and local database, two embedded transactions are performed in the given sequence.
- The observation is shown in Image3.png where we encounter `MYSQLErrorException` where lock wait timeout exceeded because a process is holding a record lock on some record for too long, thread is being timed out and the lock is still not obtainable. Refer Image3.png

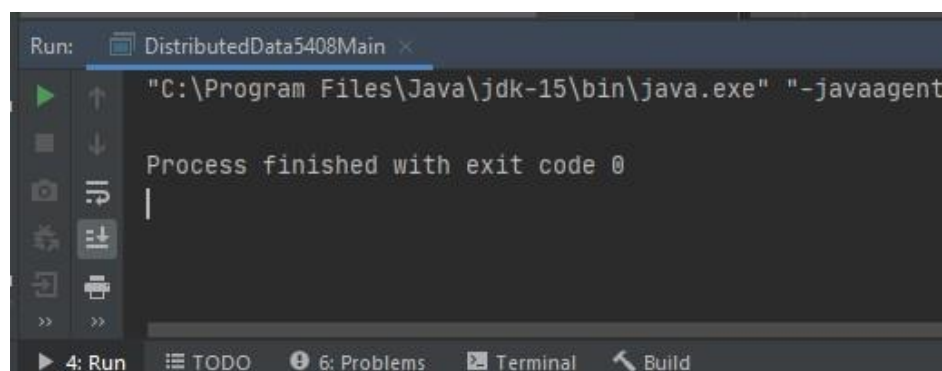
Image3



```

Run: Main x
"C:\Program Files\Java\jdk-15\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.3\lib\idea_
com.mysql.cj.jdbc.exceptions.MySQLTransactionRollbackException Create breakpoint : Lock wait timeout exceeded; try restarting transaction
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:123)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:97)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.StatementImpl.executeUpdateInternal(StatementImpl.java:1335)
    at com.mysql.cj.jdbc.StatementImpl.executeLargeUpdate(StatementImpl.java:2108)
    at com.mysql.cj.jdbc.StatementImpl.executeUpdate(StatementImpl.java:1245)
    at com.company.Transaction.sequence2(SequenceTransactions.java:46)
    at com.company.TransactionsData5408.main(SequenceTransactions.java:82)
  
```

- The controller engine is modified to perform distributed transactions and 4 update operations are performed at the local site and 4 operations (2 Update, 1 Insert, 1 Delete) are performed at the remote site and it happened successfully and updating the tables in remote database and local database.



```

Run: DistributedData5408Main x
"C:\Program Files\Java\jdk-15\bin\java.exe" "-javaagent:
Process finished with exit code 0
|
  
```

## References

- [1] [https://www.kaggle.com/olistbr/brazilianecommerce?select=olist\\_order\\_payments\\_dataset.csv](https://www.kaggle.com/olistbr/brazilianecommerce?select=olist_order_payments_dataset.csv)