

ADBMS

Name: Meet Patel

SAP-ID: 60004200104

Division/Batch: B/B1

Branch: Computer Engineering

Experiment 5

AIM: To implement Fragmentation using Range, Key, Hash and List.

ADBMS	
Meet Patel B 1	60004200104
Experiment - 5	
<u>Aim:</u> Perform fragmentation (Range, List, Hash and Key) in DBS design.	
<u>Theory:</u> MySQL partitioning is about altering - ideally optimizing - the way the database engine physically store data. It allows you to distribute portion of table data partitions across file system based on a set of user-defined rules the partitioning functions.	
<u>Types of partitioning:</u> <u>Horizontal Partitioning:</u> It means that all rows matching the partitioning function will	

rows matching the partitioning function will be assigned to different physical functions

key	room no		key	room no
1	66	→	1	66
2	89		2	89
3	77			
4	54			

Vertical Partitioning: It allows different table columns to be split into different physical partitions

id	fname	lname		id	fname
			→		

Currently MySQL supports horizontal partitioning but not vertical partitioning

Partition Types:

Range: This type of partition assigns rows to partitions based on column values that fall within a stated range

List: It is similar to Range except that the partitioning is selected based on columns matching one of set of discrete values.

Hash: In hash partitioning, a partition is selected based on values returned by a user defined expression.

Key: This is very similar to Hash partitioning but the hashing function is supplied by MySQL.

Conclusion: From personal experience, partitioning is the last part of optimization process. In general partitioning make the most sense when your dealing with million records.

OUTPUT:

RANGE:

```
1 • alter table film_year
2 PARTITION BY RANGE (year(film_years))(
3 PARTITION p0 VALUES LESS THAN (2016),
4 PARTITION p1 VALUES LESS THAN (2017),
5 PARTITION p2 VALUES LESS THAN (2018),
6 PARTITION p3 VALUES LESS THAN (2020));
7
8
9
10 • SELECT PARTITION_NAME, TABLE_ROWS
11 FROM INFORMATION_SCHEMA.PARTITIONS
12 WHERE TABLE_SCHEMA = 'sakila' AND TABLE_NAME = 'film_year';
13
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	2
	p2	2
	p3	0

LIST:

```
1 • CREATE TABLE film_genre (
2
3 category_id INT PRIMARY KEY NOT NULL
4
5 )
6
7 PARTITION BY LIST(category_id) (
8 PARTITION horror VALUES IN (101, 103, 105),
9 PARTITION comedy VALUES IN (102, 104, 106),
10 PARTITION actions VALUES IN (107, 109, 111),
11 PARTITION romance VALUES IN (108, 110, 112));
12
13 • SELECT PARTITION_NAME, TABLE_ROWS
14 FROM INFORMATION_SCHEMA.PARTITIONS
15 WHERE TABLE_SCHEMA = 'sakila' AND TABLE_NAME = 'film_genre';
16
17
18
```

Limit to 1000 rows

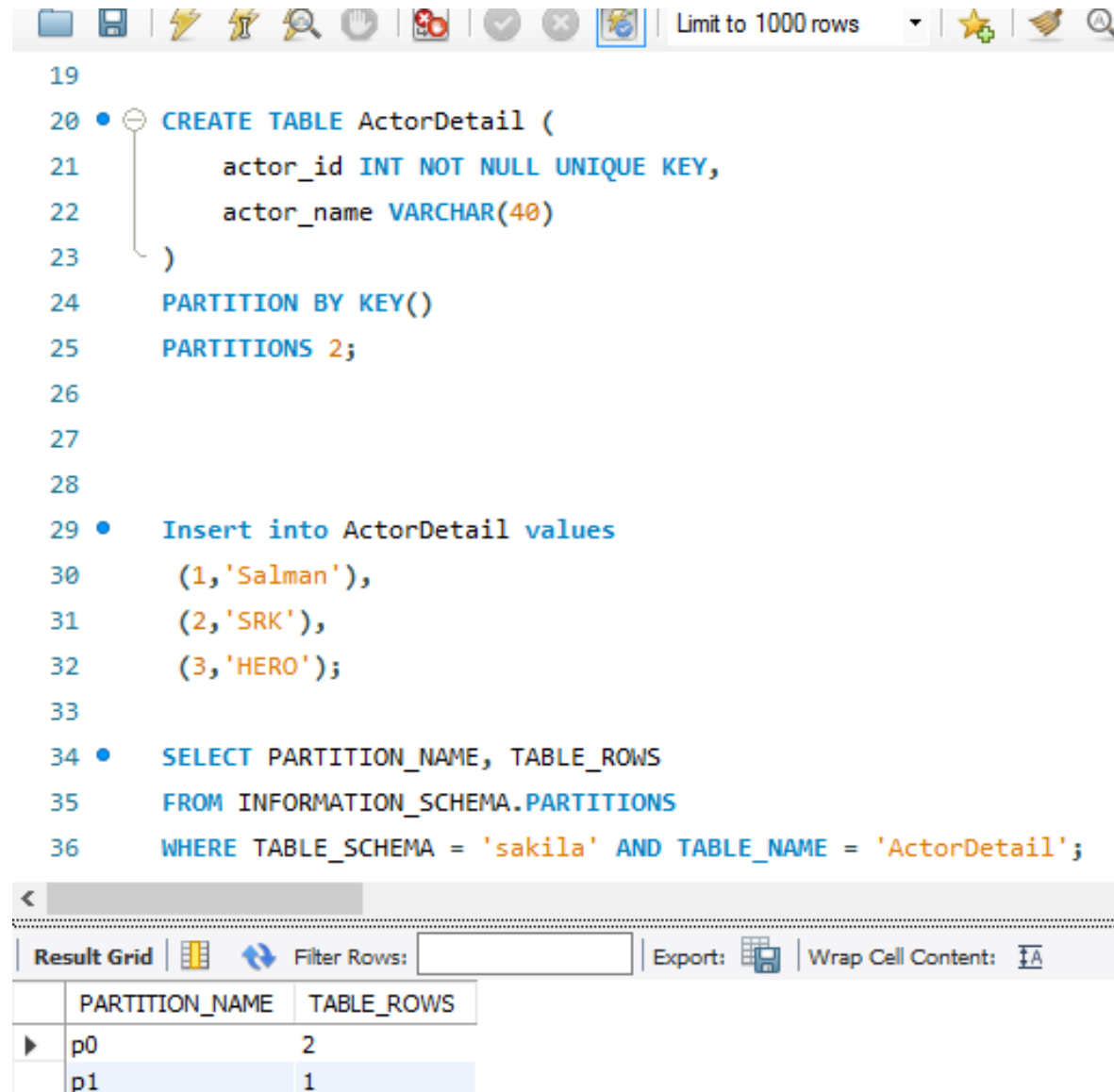
Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	PARTITION_NAME	TABLE_ROWS
▶	actions	0
	comedy	0
	horror	0
	romance	0

Autom
manua

Result Grid
Form Editor

HASH:



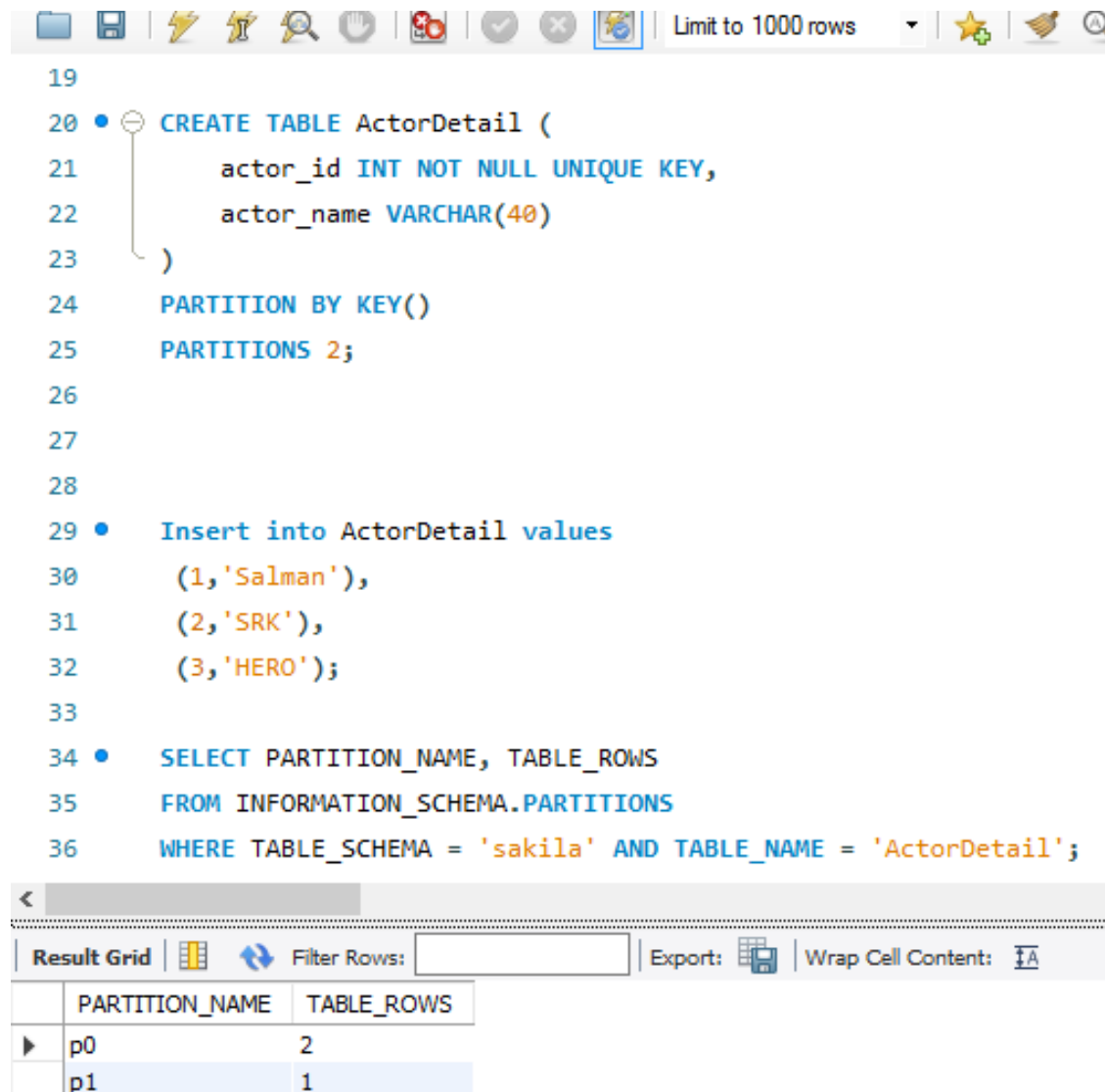
The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search, along with a "Limit to 1000 rows" dropdown. The main editor contains the following SQL script:

```
19
20 • CREATE TABLE ActorDetail (
21     actor_id INT NOT NULL UNIQUE KEY,
22     actor_name VARCHAR(40)
23 )
24 PARTITION BY KEY()
25 PARTITIONS 2;
26
27
28
29 • Insert into ActorDetail values
30     (1, 'Salman'),
31     (2, 'SRK'),
32     (3, 'HERO');
33
34 • SELECT PARTITION_NAME, TABLE_ROWS
35 FROM INFORMATION_SCHEMA.PARTITIONS
36 WHERE TABLE_SCHEMA = 'sakila' AND TABLE_NAME = 'ActorDetail';
```

Below the script, a "Result Grid" is displayed with the following data:

	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	1

KEY:



The screenshot shows a SQL IDE interface with a toolbar at the top. The main area displays SQL code for creating a table and inserting data. The code is as follows:

```
19
20 • CREATE TABLE ActorDetail (
21     actor_id INT NOT NULL UNIQUE KEY,
22     actor_name VARCHAR(40)
23 )
24 PARTITION BY KEY()
25 PARTITIONS 2;
26
27
28
29 • Insert into ActorDetail values
30     (1, 'Salman'),
31     (2, 'SRK'),
32     (3, 'HERO');
33
34 • SELECT PARTITION_NAME, TABLE_ROWS
35 FROM INFORMATION_SCHEMA.PARTITIONS
36 WHERE TABLE_SCHEMA = 'sakila' AND TABLE_NAME = 'ActorDetail';
```

Below the code editor, there is a "Result Grid" section. It includes a "Filter Rows" input field, an "Export" button, and a "Wrap Cell Content" checkbox. The result grid displays the following data:

	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	1

CONCLUSION:

Partitioning is powerful functionality that allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity.