

Kalitik Jolapoma
60004200107
B.T. CompS

Experiment 1

Date of Practical: 1/10/22

Aim: Case study of professional & commercial database

~~22~~
~~25~~

Database: CassandraDB

CassandraDB is an open-source distributed NOSQL wide-column data store. It was designed to be compatible with Apache Cassandra while achieving significantly higher throughputs and lower latencies.

CassandraDB uses a sharded design on each node, meaning that each CPU core handles a different subset of data. cores do not share data but rather communicate explicitly when they need to.

why is CassandraDB preferred?

1) Performance

Building on the NOSQL DBaaS achieves millions of OPS throughput from single node resulting in huge cost savings.

2) Low latencies

consistent single-digit millisecond P99 latencies ensure dependable performance

3) No vendor lock-in

Built in full portability, you can easily migrate data to other clouds or on-premises deployments

4) Cassandra/DynamoDB compatible

Migrate to CassandraDB cloud from Apache Cassandra or Amazon DynamoDB without changing your application code

Company: Hotstar (Disney+ Hotstar), India's most popular streaming service, accounts for over 40% of global Disney+ subscribers, offers 100,000 hours of content on-demand and most-watched sporting event (IPL with over 25 million concurrent viewers)

The Hotstar database previously consisted of a new data model, then they moved to a high-performance low-latency database-as-a-service called as Sylla Cloud. It lowered the latency for both read and write to ensure snappy user experience that today's streaming users expect - even with rapidly growing content library and skyrocketing subscriber base.

A main-engaging feature that they used was of "continue-watching" which majority of the user based had been keen to use as it would promote the watching again from that particular time-stamp. So they use SyllaDB to process 100 to 200GB of data daily for this feature to be in use.

Comparison:

DynamoDB

PROS:- It is a fully managed proprietary NoSQL database that supports the column data store. It was designed to be compatible as well as key-value pairs.

SyllaDB

It is an open-source distributed NoSQL database with wide-service that supports the column data store. It was designed to be compatible with Apache Cassandra to lower latencies.

ANS:-

Sent OR ?

Compatible

	Dyndrone	Spillabe
Usage	Commercial	Open-Source
Cloud	✗	✗
baked only	✗	✗
Built-in security	Yes, with encryption at rest	-
Continuous backup	✗	✗
Cost	High	Very Low
Migration	Possible	not competitive
Expiration	-	-
Scalability	Tracks how close usage highly scalable so not performance is to the upper bounds doesn't adjust itself according to scaling the work to the database to auto-scale	-
ACID	permits fine-grained ACID might also demand controls to access controls conflicts per object, no granular data by table owner distinguishing available smoothing the workload to avoid creating bottlenecks	-
Planning	Only factor where Dyndrone falls apart, limited planning cost	-
	\$ 525,000	\$ 112,00
Regions	✓	✗
Transaction	ACID	✗ AFI for single operations
Encryption	✗	self-tunable replication factor
Throughput	Overall throughput: 8.57 2.3M	202K (ops/sec)
Workload	READ operations: 50M 2.3M	2.3M (Avg)
	UPDATE operations: 50M 2.3M	2.3M (Avg)

Recommendation: As the previously used database which is SyllaDB has some drawbacks of its own and which DynamoDB shines in. Some of the features are like the persistence of event-stream data, which has time-ordered streams within the AWS database that enables the developers to receive and update item-level data both before and after DynamoDB updates. In case of the system failure the SyllaDB also doesn't hold the backup database in which there can be stored the data; it has some backup of the data but not comparable to the AWS services provided by DynamoDB. There are many AWS options to use as a service for DynamoDB and its data analysis and maintenance of which DynamoDB has a few. As Hotsail needs high performance and low latency database ^{SyllaDB} ~~DynamoDB~~ satisfies its job but DynamoDB is a very good option.

Conclusion: SyllaDB has both its advantages and disadvantages but overall it is working perfectly for the intended work of Hotstar in serving the customers to their requirements. The only major deciding factor of the usage of SyllaDB is because of its cost and the vast open source community. Apart from those factors DynamoDB is a far better alternative to it and can be used extensively instead of SyllaDB to manifest all the AWS cloud services as well.

Name: Kartik Jolapara

SAPID: 60004200107

DIV: B/B1

ADBMS

Exp 2

Kartik Jolapara

60004200107

8/10/22

81

ADBMS

Exp 2

21
25
f

Page No. 1
Date 8/10/22

Aim: Perform operation like searching, insertion, deletion on B-Tree and B+Tree

Theory:

@B-Tree

B-Tree is a self-balancing tree. B-trees are useful when we are dealing with huge amounts of data that can't be fitted in main memory. When the number of keys is high, the data is read from the disk in form of blocks. Disk access time is very high compared to the main memory access time.

The main idea of using B-Trees is to reduce the number of disk access. Most of the tree operations require $O(h)$ disk access where h is the height of the tree. B-Tree is a fat tree meaning, the height of BTree is kept low by putting maximum possible keys in a B-tree node. Generally, the B-Tree node size is kept equal to disk block size since the height of the B-Tree is kept low so total disk access for most of the operations are reduced significantly compared to balanced Binary search tree like AVL Tree.

Properties of B-Tree

→ All leaves are at the same level

→ B-Tree is defined by term minimum degree 't'

The value of 't' depends upon disk block size

→ Every node except the root must contain atleast $(t-1)$ keys. The root may contain minimum of 1 key.

Page No. 2
Date 8/10/22

- All nodes may contain almost $(2^k - 1)$ keys.
- No. of children of a node is equal to no. of keys in it plus 1
- All keys of a node are sorted in increasing order.
- It grows & shrinks from root.
- Insertion happens only at leaf node.

Time complexity of B-Tree:

Search - $O(\log n)$

Insert - $O(\log n)$

Delete - $O(\log n)$

* B+ Tree

B+ Tree is an extension of B-Tree which allows efficient insertion, deletion and search operations. In B-tree keys and records both can be stored in internal as well as leaf nodes whereas, in B+ trees records can only be stored on leaf nodes and internal nodes can only store key values.

The leaf nodes of a B+ tree are linked together in the form of singly linked list to make search queries more efficient. B+ trees are used to store the large amount of data which can't be stored in main memory. Due to the fact that size of main memory is always limited the internal nodes of B+ tree are stored in main memory whereas, leaf nodes are stored in secondary memory.

Properties of B+ tree

- All leafs are at some level.
- The root has atleast two children
- Each node except root can have a maximum of m children and atleast $m/2$ children
- Each node can contain a maximum of $(m-1)$ keys and a maximum of $\lceil m/2 \rceil - 1$ keys
- Keys are used for indexing
- Data can be stored sequentially or directly

Time complexity

Search - $O(\log n)$

Insert - $O(\log n)$

Delete - $O(\log n)$

B-tree v/s B+ tree

B-tree

- ① Search keys cannot be repeatedly stored
- ② Data is stored in the leaf node as well as internal nodes
- ③ Searching for some data is slower process as some data can be found on internal nodes as well as leaf nodes
- ④ Deletion of internal nodes are complicated & time consuming

B+ tree

- ① Redundant search keys can be present.
- ② Data can only be stored on the leaf nodes
- ③ Search is comparatively faster as data can only be found on the leaf nodes.
- ④ Deletion will never be a complexed process since some elements will be deleted from leaf

Page No. 4
Date 8/10/22

B TREE

④ Leaf nodes can't be linked together.

B+ TREE

④ Leaf nodes are linked together to make search operations more efficient.

Conclusion: Thus, we successfully studied & compared various operations in B tree & B+ tree.

B Tree Code:

```
class BTreeNode:  
    def __init__(self, leaf=False):  
        self.leaf = leaf  
        self.keys = []    self.child  
        = []  
  
class BTree:  
    def __init__(self, t):  
        self.root = BTreeNode(True)  
        self.t = t  
  
    def insert(self, k):    root = self.root  
        if len(root.keys) == (2 * self.t) - 1:  
            temp = BTreeNode()  
            self.root = temp  
            temp.child.insert(0, root)  
            self.split_child(temp, 0)  
            self.insert_non_full(temp, k)  
        else:  
            self.insert_non_full(root, k)
```

```

def insert_non_full(self, x, k):
    i = len(x.keys) - 1
    if x.leaf:
        x.keys.append((None, None))
    while i >= 0 and k[0] < x.keys[i][0]:
        x.keys[i + 1] = x.keys[i]
        i -= 1
        x.keys[i + 1] = k
    else:
        while i >= 0 and k[0] < x.keys[i][0]:
            i -= 1      i += 1
            if len(x.child[i].keys) == (2 * self.t) - 1:
                self.split_child(x, i)
        if k[0] > x.keys[i][0]:
            i += 1
            self.insert_non_full(x.child[i], k)

def split_child(self, x, i):
    t = self.t      y =
    x.child[i]      z =
    BTreeNode(y.leaf)
    x.child.insert(i + 1, z)
    x.keys.insert(i, y.keys[t - 1])
    z.keys = y.keys[t: (2 * t) - 1]
    y.keys = y.keys[0: t - 1]
    if not y.leaf:
        z.child = y.child[t: 2 * t]

```

```

y.child = y.child[0: t - 1]

def print_tree(self, x, l=0):
    print("Level ", l, " ", len(x.keys), end=":")
    for i in x.keys:    print(i, end=" ")    print()
    l += 1    if len(x.child) > 0:    for i in
    x.child:    self.print_tree(i, l)

def search_key(self, k, x=None):
    if x is not None:
        i = 0    while i < len(x.keys) and k >
        x.keys[i][0]:    i += 1    if i < len(x.keys)
        and k == x.keys[i][0]:
            return (x, i)
    elif x.leaf:
        return None
    else:
        return self.search_key(k, x.child[i])

else:
    return self.search_key(k, self.root)

def main():  B
= BTree(3)

```

```

for i in range(10):
    B.insert((i, 2 * i))

B.print_tree(B.root)

if B.search_key(8) is not None:
    print("\nFound")
else:
    print("\nNot Found")

if __name__ == '__main__':
    main()

```

Output:

```

Level 1 2: (3, 6) (4, 8)
Level 1 4: (6, 12) (7, 14) (8, 16) (9, 18)

```

```
Found
```

B+ Tree

Code:

```
import math
```

```

class Node:
    def
        __init__(self, order):

```

```
    self.order = order  
    self.values = []      self.keys  
    = []      self.nextKey =  
    None      self.parent =  
    None      self.check_leaf =  
    False
```

```
def insert_at_leaf(self, leaf, value, key):  
    if (self.values):  
        temp1 = self.values  
        for i in range(len(temp1)):  
            if (value == temp1[i]):  
                self.keys[i].append(key)  
                break  
            elif (value <  
                  temp1[i]):  
                self.values = self.values[:i] + [value] + self.values[i:]  
                self.keys = self.keys[:i] + [[key]] + self.keys[i:]  
                break  
            elif (i + 1 == len(temp1)):  
                self.values.append(value)          self.keys.append([key])  
                break  
            else:  
                self.values = [value]  
                self.keys = [[key]]
```

```
class BplusTree:
```

```

def __init__(self, order):
    self.root = Node(order)
    self.root.check_leaf = True

def insert(self, value, key):
    value = str(value)
    old_node = self.search(value)
    old_node.insert_at_leaf(old_node,
                           value, key)

    if (len(old_node.values) == old_node.order):
        node1 = Node(old_node.order)
        node1.check_leaf = True          node1.parent =
        old_node.parent      mid = int(math.ceil(old_node.order /
2)) - 1          node1.values = old_node.values[mid + 1:]
        node1.keys = old_node.keys[mid + 1:]      node1.nextKey
        = old_node.nextKey      old_node.values =
        old_node.values[:mid + 1]      old_node.keys =
        old_node.keys[:mid + 1]      old_node.nextKey = node1
        self.insert_in_parent(old_node, node1.values[0], node1)

def search(self, value):      current_node =
    self.root      while(current_node.check_leaf
    == False):
        temp2 = current_node.values
        for i in range(len(temp2)):
            if (value == temp2[i]):

```

```
        current_node = current_node.keys[i + 1]
break      elif (value < temp2[i]):
        current_node = current_node.keys[i]
break      elif (i + 1 ==
len(current_node.values)):
current_node = current_node.keys[i + 1]
break
return current_node
```

```
def find(self, value, key):    l =
self.search(value)    for i, item in
enumerate(l.values):    if item ==
value:    if key in l.keys[i]:
return True    else:
return False    return False
```

```
def insert_in_parent(self, n, value, ndash):
if (self.root == n):
    rootNode = Node(n.order)
    rootNode.values = [value]
    rootNode.keys = [n, ndash]
    self.root = rootNode    n.parent
    = rootNode    ndash.parent =
    rootNode    return
parentNode = n.parent
temp3 = parentNode.keys    for
```

```

i in range(len(temp3)):           if
(temp3[i] == n):
    parentNode.values = parentNode.values[:i] + \
        [value] + parentNode.values[i:]
parentNode.keys = parentNode.keys[:i +
    1] + [ndash] + parentNode.keys[i + 1:]
if (len(parentNode.keys) > parentNode.order):
    parentdash = Node(parentNode.order)
    parentdash.parent = parentNode.parent           mid =
    int(math.ceil(parentNode.order / 2)) - 1
    parentdash.values = parentNode.values[mid + 1:]
    parentdash.keys = parentNode.keys[mid + 1:]
    value_ = parentNode.values[mid]           if (mid == 0):
        parentNode.values = parentNode.values[:mid + 1]
else:
    parentNode.values = parentNode.values[:mid]
    parentNode.keys = parentNode.keys[:mid + 1]           for j
    in parentNode.keys:           j.parent = parentNode
    for j in parentdash.keys:           j.parent = parentdash
    self.insert_in_parent(parentNode, value_, parentdash)

def delete(self, value, key):
    node_ = self.search(value)

    temp = 0
    for i, item in enumerate(node_.values):
        if item == value:

```

```
temp = 1

if key in node_.keys[i]:
    if len(node_.keys[i]) > 1:
        node_.keys[i].pop(node_.keys[i].index(key))
        elif node_ == self.root:
            node_.values.pop(i)
        else:
            node_.keys[i].pop(node_.keys[i].index(key))
        del node_.keys[i]
        node_.values.pop(node_.values.index(value))
        self.deleteEntry(node_, value, key)
        else:
            print("Value not in Key")
    return      if temp == 0:
                print("Value not in Tree")
return
```

```
def deleteEntry(self, node_, value, key):  
    if not node_.check_leaf:  
        for i, item in enumerate(node_.keys):  
            if item == key:  
                node_.keys.pop(i)  
                break  
            for i, item in  
                enumerate(node_.values):  
                if item  
                    == value:  
                    del node_.values[i]
```

```

    node_.values.pop(i)
break

if self.root == node_ and len(node_.keys) == 1:
    self.root = node_.keys[0]
node_.keys[0].parent = None
del node_
return

elif (len(node_.keys) < int(math.ceil(node_.order / 2)) and node_.check_leaf
== False) or (len(node_.values) < int(math.ceil((node_.order - 1) / 2)) and
node_.check_leaf == True):

    is_predecessor = 0
parentNode = node_.parent
PrevNode = -1
NextNode = -1
PrevK = -1          PostK = -1          for
i, item in enumerate(parentNode.keys):

    if item == node_:

if i > 0:
    PrevNode = parentNode.keys[i - 1]
    PrevK = parentNode.values[i - 1]

    if i < len(parentNode.keys) - 1:
        NextNode = parentNode.keys[i + 1]
        PostK = parentNode.values[i]

```

```

    if PrevNode == -1:           ndash = NextNode
    value_ = PostK      elif NextNode == -1:      is_predecessor
    = 1                  ndash = PrevNode       value_ = PrevK
    else:                if len(node_.values) + len(NextNode.values) <
    node_.order:
        ndash = NextNode
    value_ = PostK      else:
        is_predecessor = 1
    ndash = PrevNode
    value_ = PrevK

    if len(node_.values) + len(ndash.values) < node_.order:
        if is_predecessor == 0:          node_, ndash = ndash,
        node_           ndash.keys += node_.keys      if not
        node_.check_leaf:
            ndash.values.append(value_)
        else:
            ndash.nextKey = node_.nextKey
    ndash.values += node_.values

    if not ndash.check_leaf:
        for j in ndash.keys:
            j.parent = ndash
            self.deleteEntry(node_.parent, value_, node_)
        del node_      else:      if is_predecessor ==
        1:          if not node_.check_leaf:

```

```

ndashpm = ndash.keys.pop(-1)

ndashkm_1 = ndash.values.pop(-1)

node_.keys = [ndashpm] + node_.keys

node_.values = [value_] + node_.values

parentNode = node_.parent           for i, item in

enumerate(parentNode.values):

    if item == value_:

        p.values[i] = ndashkm_1

break      else:

    ndashpm = ndash.keys.pop(-1)

ndashkm = ndash.values.pop(-1)

node_.keys = [ndashpm] + node_.keys

node_.values = [ndashkm] + node_.values

parentNode = node_.parent           for i, item in

enumerate(p.values):               if item == value_:

    parentNode.values[i] = ndashkm

break      else:                  if not

node_.check_leaf:

    ndashp0 = ndash.keys.pop(0)

ndashk0 = ndash.values.pop(0)           node_.keys

= node_.keys + [ndashp0]                 node_.values =

node_.values + [value_]                 parentNode =

node_.parent                   for i, item in

enumerate(parentNode.values):         if item ==

value_:

    parentNode.values[i] = ndashk0

break      else:

```

```
ndashp0 = ndash.keys.pop(0)
ndashk0 = ndash.values.pop(0)           node_.keys
= node_.keys + [ndashp0]                node_.values =
node_.values + [ndashk0]               parentNode =
node_.parent             for i, item in
enumerate(parentNode.values):          if item ==
value_:
                                         parentNode.values[i] = ndash.values[0]
break
```

```
if not ndash.check_leaf:
for j in ndash.keys:
j.parent = ndash           if not
node_.check_leaf:           for j in
node_.keys:                 j.parent =
node_                   if not
parentNode.check_leaf:
for j in parentNode.keys:
j.parent = parentNode
```

```
def printTree(tree):
lst = [tree.root]
level = [0]   leaf
= None   flag = 0
lev_leaf = 0
```

```
node1 = Node(str(level[0]) + str(tree.root.values))

while (len(lst) != 0):      x =
    lst.pop(0)      lev = level.pop(0)
    if (x.check_leaf == False):      for i,
        item in enumerate(x.keys):
            print(item.values)
    else:
        for i, item in enumerate(x.keys):
            print(item.values)
        if (flag == 0):
            lev_leaf = lev
            leaf = x      flag = 1

record_len = 3 bplustree =
BplusTree(record_len)
bplustree.insert('5', '33')
bplustree.insert('15', '21')
bplustree.insert('25', '31')
bplustree.insert('35', '41')
bplustree.insert('45', '10')

printTree(bplustree)

if(bplustree.find('45', '10')):
    print("Found") else:
```

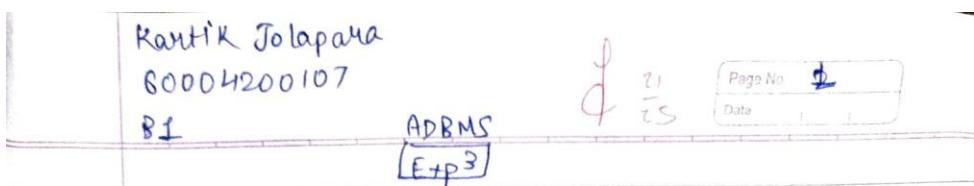
```
print("Not found") Output:
```

```
['15', '25']  
['35', '45']  
['5']  
Found
```

Name: Kartik Jolapara
Batch: B1

SapID: 60004200107

ADBMS Experiment-3



Aim: To simulate query optimisation by performing SEL query on database

Theory: Query optimization is of great importance for performance of a relational database, especially for execution of complexed SQL statements.

There are 2 ways:-

- 1) Heuristic Based
- 2) Cost Based

Heuristic Based:-

A query tree is a data structure that corresponds to a relational algebra expression. The same query could be corresponding to many different relational expressions & hence many different query trees. The task of heuristic optimization of query trees is to find a final query tree that is efficient to execute. The main heuristic is to apply first the operations that reduce the size of intermediate results.

Cost Based:-

Estimate and compare the costs of executing a query using different execution strategies and chose the strategy with lowest cost estimate. The cost of any query includes access wrt to secondary storage, storage cost, memory usage cost, no. of memory buffer at time of execution, communication etc.

Conclusion: In this experiment we have performed table level and index level optimization and compared it to the results of no optimization.

1) SELECT QUERY

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "%GEN%";
```

The results pane shows the output of the query. The SQL Additions panel on the right contains the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the same query in the SQL Editor. The results pane now displays the output of the query. The SQL Additions panel on the right contains the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

AFTER OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
 - Tables
 - actor
 - address
 - category
 - city
 - country
 - customer
 - film
 - film_actor
 - film_category
 - film_text
 - inventory
 - language
 - payment

SQL File 4*

```

1 USE sakila;
2 select * from actor where upper(last_name) like "%GEN%";
3 optimize table actor;

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.06200000

Timing (as measured by the server):
Execution time: 0:00:0.05738530
Table lock wait time: 0:00:0.00000390

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scan (Select_scan): 0
Range block scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Other Info:
Event Id: 106
Thread Id: 60

actor 3 Result 4 Result 5 x

Object Info Session Output

Apply Revert Context Help Snippets

2) NESTED

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
 - Tables
 - actor
 - address
 - category
 - city
 - country
 - customer
 - film
 - film_actor
 - film_category
 - film_text
 - inventory
 - language
 - payment

SQL File 4*

```

1 USE sakila;
2 select * from actor where upper(last_name) like "%GEN%";
3 SELECT concat(first_name, " ", last_name) AS name, email
4   from customer where customer_id IN
5     (select customer_id from rental where inventory_id in
6       (select inventory_id from inventory where film_id in
7         (select film_id from film_category join category using (category_id) where category.name = "Action")));

```

Visual Explain

Display Info: Read + Eval cost | Overview: | View Source: |

```

graph LR
    A[Full Table Scan Category] --> B[Non-Unique Key Lookup film_category]
    B --> C[Non-Unique Key Lookup inventory_idx_fk_film_id]
    C --> D[Non-Unique Key Lookup rental_idx_m_inventory_id]
    D --> E[Unique Key Lookup customer PRIMARY]
    E --> F[DISTINCT]
    F --> G[query_block #1]

```

Query cost: 1483.11

1.5499999999999999 Full Table Scan Category

10.41 Non-Unique Key Lookup film_category

72.88 Non-Unique Key Lookup inventory_idx_fk_film_id

584.97 Non-Unique Key Lookup rental_idx_m_inventory_id

477.7399999999995 Unique Key Lookup customer PRIMARY

12.20 nested loop 100 rows

85.14 nested loop 478 rows

670.11 nested loop 1.57K rows

1483.11 nested loop 1599 rows

DISTINCT

tmp table

Query Block #1

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Form Editor Field Types Query Stats Execution Plan

Object Info Session Output

Read Only Context Help Snippets

BEFORE OPTIMIZING

MySQL Workbench - Local instance MySQL80

SQL File 4*

```

1 • USE sakila;
2 • select * from actor where upper(last_name) like "%GEN%" ;
3 • SELECT concat(first_name, " ",last_name) AS name, email
4   from customer where customer_id IN
5     (select customer_id from rental where inventory_id in
6       (select inventory_id from inventory where film_id IN
7         (select film_id from film_category join category using (category_id) where category.name = "Action")));

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.000000000

Timing (as measured by the server):
Execution time: 0:00:0.00514680
Table lock wait time: 0:00:0.000000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 510
Rows examined: 2616

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Join using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 130
Thread Id: 60

actor 6 Result 7 Result 8 x

Object Info Session Output

Apply Revert Context Help Snippets

AFTER OPTIMIZING

MySQL Workbench - Local instance MySQL80

SQL File 4*

```

3 • SELECT concat(first_name, " ",last_name) AS name, email
4   from customer where customer_id IN
5     (select customer_id from rental where inventory_id in
6       (select inventory_id from inventory where film_id IN
7         (select film_id from film_category join category using (category_id) where category.name = "Action")));
8 • optimize table customer , rental , inventory , category ;

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.623000000

Timing (as measured by the server):
Execution time: 0:00:0.61268200
Table lock wait time: 0:00:0.00011400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Other Info:
Event Id: 135
Thread Id: 60

actor 6 Result 7 Result 8 Result 9 x

Object Info Session Output

Apply Revert Context Help Snippets

3) LEFT JOIN

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `sakila` with tables `actor`, `address`, `category`, `city`, `country`, `customer`, `film`, `film_actor`, `film_category`, `film_text`, `inventory`, `language`, and `payment`.
- SQL Editor:** Contains the following SQL query:


```

6   (select inventory_id from inventory where film_id in
7     (select film_id from film_category join category using (category_id) where category.name = "Action")));
8
9 *  select stf.first_name , stf.last_name , ad.address , ad.district , ad.postal_code , ad.city_id
10  from staff stf
11  left join address ad
12  on stf.address_id = ad.address_id;
      
```
- Visual Explain:** Displays the execution plan with the following details:
 - query_block #1:** Query cost: 3.90
 - stf:** Full Table Scan, 2 rows
 - ad:** PRIMARY, Unique Key Lookup, 1 row
 - nested loop:** 3.9, 2 rows (from stf) x 1 row (from ad)
- Output:** Shows the results of the query.

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `sakila` with the same set of tables as the first screenshot.
- SQL Editor:** Contains the same SQL query as the first screenshot.
- Query Statistics:** Displays the following information:
 - Timing (as measured at client side):** Execution time: 0:00:0.0000000
 - Timing (as measured by the server):** Execution time: 0:00:0.0004150, Table lock wait time: 0:00:0.000000400
 - Errors:** Had Errors: NO, Warnings: 0
 - Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 4
 - Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Execution Plan:** Shows the same execution plan as the first screenshot, indicating a nested loop join.
- Output:** Shows the results of the query.

AFTER OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- address
- category
- city
- country
- customer
- film
- film_actor
- film_category
- film_text
- inventory
- language
- payment

SQL File 4*

```

8
9 • select stf.first_name , stf.last_name , ad.address , ad.district , ad.postal_code , ad.city_id
10 from staff stf
11 left join address ad
12 on stf.address_id = ad.address_id;
13
14 • optimize table staff , address

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.18700000

Timing (as measured by the server):
Execution time: 0:00:0.17137930
Table lock wait time: 0:00:0.00011100

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:

- Full table scans (Select_full_scan): 0
- Index table scans (Select_full_join): 0
- Joins using range search (Select_full_range_join): 0
- Joins with range checks (Select_range_check): 0
- Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Other Info:
Event Id: 191
Thread Id: 60

actor 14 Result 15 Result 16 Result 17 Result 18 x

Object Info Session Output

Apply Revert Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

4) RIGHT JOIN

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- address
- category
- city
- country
- customer
- film
- film_actor
- film_category
- film_text
- inventory
- language
- payment

SQL File 4*

```

15
16 right join film_actor fin_act
17 on fin_act.film_id = fim_act.film_id
18 group by fin.title
19 order by number_of_actors desc;
20
21

```

Visual Explain

Display Info: Read + Eval cost

Query cost: 6164.92

query_block #1

GROUP → ORDER

tmp table

6154.15 5.46K rows

556.38 5.46K rows

5598.55 1 row

fin PRIMARY

Read Only Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

BEFORE OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- address
- category
- city
- customer
- film
- film_category
- film_actor
- film_text
- inventory
- language
- payment

Administration Schemas

Information

Table: actor

Columns:

actor_id	smallint UN
first_name	AI PK
last_name	varchar(45)
last_update	timestamp

Timing (as measured at the client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.0053390
Table lock wait time: 0:00:0.00000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 997
Rows examined: 11921

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Join using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1

Index Usage:
At least one Index was used

Other Info:
Event Id: 325
Thread Id: 60

actor 36 Result 37 Result 38 Result 39 Result 40 ×

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

AFTER OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- address
- category
- city
- customer
- film
- film_actor
- film_category
- film_text
- inventory
- language
- payment

Administration Schemas

Information

Table: actor

Columns:

actor_id	smallint UN
first_name	AI PK
last_name	varchar(45)
last_update	timestamp

Timing (as measured at the client side):
Execution time: 0:00:0.2340000

Timing (as measured by the server):
Execution time: 0:00:0.2194260
Table lock wait time: 0:00:0.0000560

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Other Info:
Event Id: 330
Thread Id: 60

actor 36 Result 37 Result 38 Result 39 Result 40 Result 41 ×

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

5) INNER JOIN

```

15   from film fm
16   inner join film_actor fm_act
17   on fm.film_id = fm_act.film_id
18   group by fm.title
19   order by number_of_actors desc;
20
21
--
```

Visual Explain

Query cost: 1473.18

query_block #1

GROUP → ORDER tmp_table,filesort

110.1K rows from film idx_title

1000 rows to nested loop

1473.1K rows from fm_act idx_fk_film_id

5 rows to ORDER

BEFORE OPTIMIZING

Timing (as measured at client side):
Execution time: 0:00:0.0000000

Timing (as measured by the server):
Execution time: 0:00:0.00371900
Table lock wait time: 0:00:0.0000000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 997
Rows examined: 7459

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Join per Type:
Full table scans (Select_scan): 1
Join using range scan (Select_range): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_checked): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1

Index Usage:
At least one index was used

Other Info:
Event Id: 288
Thread Id: 60

AFTER OPTIMIZING

The screenshot shows the MySQL Workbench interface. The main window displays a SQL editor with the following script:

```
15 from film fml
16 inner join film_actor fim_act
17 on fml.film_id = fim_act.film_id
18 group by fml.title
19 order by number_of_actors desc;
20
21 • optimize table film , film_actor
```

Below the script, the "Query Statistics" section provides performance metrics:

- Timing (as measured at the client side):**
Execution time: 0:00:0.18800000
- Timing (as measured by the server):**
Execution time: 0:00:0.17838000
Table lock wait time: 0:00:0.00000600

The "Errors" section indicates no errors or warnings.

The "Rows Processed" section shows 0 rows affected, sent to client, and examined.

The "Temporary Tables" section shows 0 temporary disk tables and 0 temporary tables created.

The "Indexes" section lists the following indexes for the actor table:

- b'akila'
- newschema
- sala'
- actor
- address
- category
- city
- country
- customer
- film
- film_actor
- film_category
- film_text
- inventory
- language
- payment

The "Table: actor" section provides detailed information about the table's columns:

Column	Type	Properties
actor_id	smallint(6)	UNI PK
first_name	varchar(45)	
last_name	varchar(45)	
last_update	timestamp	

The "Columns" section also lists the columns and their properties.

On the right side, there are several panels: "SQLAdditions", "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan". The "Query Stats" panel is currently active, showing the execution plan and other statistics.

6) CROSS JOIN

BEFORE OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- columns
- indexes
- foreign keys
- triggers
- address
- category
- city
- country
- customer
- film
- film_actor
- film_category
- film_text
- language
- inventory
- payment

Administration Schemas

Information

Table: actor

Columns:

actor_id	smallint UN
first_name	AI PK
last_name	varchar(45)
last_update	timestamp

Temporary Tables:

- Temporary disk tables created: 0
- Temporary tables created: 0

Timing (as measured at client side):

- Execution time: 0:00:0.01500000

Timing (as measured by the server):

- Execution time: 0:00:0.00581580

Table lock wait time: 0:00:0.00000000

Errors:

- Had Errors: NO
- Warnings: 0

Rows Processed:

- Rows affected: 0
- Rows sent to client: 997
- Rows examined: 749

Temporary Tables:

- Temporary disk tables created: 0
- Temporary tables created: 0

Joins per Type:

- Full table scans (Select_scan): 1
- Index table scans (Select_full_join): 0
- Joins using range search (Select_full_range_join): 0
- Joins with range checks (Select_range_check): 0
- Joins using range (Select_range): 0

Sorting:

- Sorted rows (Sort_rows): 997
- Sort merge passes (Sort_merge_passes): 0
- Sorts with ranges (Sort_range): 0
- Sorts with table scans (Sort_scan): 1

Index Usage:

- At least one Index was used

Other Info:

- Event Id: 222
- Thread Id: 60

actor 19 Result 20 Result 21 Result 22 Result 23 x

Object Info Session Output

Apply Revert Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

AFTER OPTIMIZING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila
- Tables
- actor
- columns
- indexes
- foreign keys
- triggers
- address
- category
- city
- country
- customer
- film
- film_actor
- film_category
- film_text
- language
- inventory
- payment

Administration Schemas

Information

Table: actor

Columns:

actor_id	smallint UN
first_name	AI PK
last_name	varchar(45)
last_update	timestamp

Temporary Tables:

- Temporary disk tables created: 0
- Temporary tables created: 0

Timing (as measured at client side):

- Execution time: 0:00:0.18800000

Timing (as measured by the server):

- Execution time: 0:00:0.19220070

Table lock wait time: 0:00:0.00007700

Errors:

- Had Errors: NO
- Warnings: 0

Rows Processed:

- Rows affected: 0
- Rows sent to client: 0
- Rows examined: 0

Temporary Tables:

- Temporary disk tables created: 0
- Temporary tables created: 0

Joins per Type:

- Full table scans (Select_scan): 0
- Index table scans (Select_full_join): 0
- Joins using range search (Select_full_range_join): 0
- Joins with range checks (Select_range_check): 0
- Joins using range (Select_range): 0

Sorting:

- Sorted rows (Sort_rows): 0
- Sort merge passes (Sort_merge_passes): 0
- Sorts with ranges (Sort_range): 0
- Sorts with table scans (Sort_scan): 0

Index Usage:

- At least one Index was used

Other Info:

- Event Id: 245
- Thread Id: 60

actor 24 Result 25 Result 26 Result 27 Result 28 Result 29 x

Object Info Session Output

Apply Revert Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

USING INDEXING

1) SIMPLE QUERY

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
USE sakila;
select * from film where length = 120;
```

The "Execution Plan" tab displays the following execution plan diagram:

```
graph TD
    A[query_block #1] --> B[Non-Unique Key Lookup  
film  
length]
    B --> C[9 rows]
    C --> D[9.22]
    D --> E[Query cost: 9.22]
```

The "Result Grid" tab shows the results of the query.

BEFORE INDEXING

The screenshot shows the MySQL Workbench interface with a query editor containing the same SQL code as above. The "Query Statistics" tab displays the following information:

Timing (as measured at client side):
Execution time: 0:00:0.0000000

Timing (as measured by the server):
Execution time: 0:00:0.00041420
Table lock wait time: 0:00:0.00000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 9

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 0
Join using index scan (Select_index_scan): 0
Join using range search (Select_full_range_join): 0
Table lock wait time: 0:00:0.00000300
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one index was used

Other Info:
Event Id: 104
Thread Id: 131

AFTER INDEXING

The screenshot shows the MySQL Workbench interface. The main window displays a SQL editor with the following query:

```
1 • USE sakila;
2 • create index length on film(length);
3 • select * from film where length = 120;
```

The left sidebar shows the Navigator and Schemas panes. The Schemas pane lists databases (b'akila', newschemas), and the sakila database, which contains tables such as actor, address, category, city, country, customer, film, film_actor, film_category, film_text, inventory, language, payment, rental, staff, and store.

A context help panel on the right is titled "Automatic context help is disabled. Use the toolbar manually get help for the current caret position or toggle automatic help." It includes tabs for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

At the bottom, there are tabs for Explain, film 5, Object Info, Session, Output, Apply, Revert, Context Help, and Snippets.

2) NESTED

The screenshot shows the MySQL Workbench interface with a complex SQL query and its execution plan.

SQL File 4*

```
1 USE sakila;
2 select * from actor where upper(last_name) like "%GEN%";
3 SELECT concat(first_name, " ", last_name) AS name, email
4 from customer where customer_id IN
5 (select customer_id from rental where inventory_id in
6 (select inventory_id from inventory where film_id in
7 (select film_id from film_category join category using (category_id) where category.name = "Action")));
```

Visual Explain

The Visual Explain plan illustrates the execution flow:

- Initial table: **actor** (Full Table Scan).
- Join 1: **actor** (100 rows) joins **film_category** (62 rows) via **actor_id** (Nested Loop).
- Join 2: The result of Join 1 joins **category** (4 rows) via **category_id** (Nested Loop).
- Join 3: The result of Join 2 joins **inventory** (478 rows) via **inventory_id** (Nested Loop).
- Join 4: The result of Join 3 joins **rental** (3 rows) via **rental.inventory_id** (Nested Loop).
- Final step: **rental** (3 rows) joins **customer** (1483.1599 rows) via **customer.customer_id** (Unique Key Lookup).

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

Object Info **Session** **Output** **actor 6** **Result 7** **Read Only** **Context Help** **Snippets**

BEFORE INDEXING

MySQL Workbench - Local instance MySQL80

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila**
 - Tables**
 - actor**
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - address
 - category
 - city
 - country
 - customer
 - film
 - film_actor
 - film_category
 - language
 - payment

Administration Schemas Information

Table: actor

Columns:

actor_id	smallint UN
first_name	AI PK varchar(45)
last_name	varchar(45)
last_update	timestamp

Temporary Tables: Temporary disk tables created: 0
Temporary tables created: 1

Timing (as measured at client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.00514680
Table lock wait time: 0:00:0.000000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 510
Rows examined: 2616

Join Statistics

Joins per Type:
Full table scans (Select_scan): 1
Join using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No index used

Other Info:
Event Id: 130
Thread Id: 60

actor 6 Result 7 Result 8

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

AFTER INDEXING

MySQL Workbench - Local instance MySQL80

Navigator

SCHEMAS

- b'sakila'
- newschema
- sakila**
 - Tables**
 - actor**
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - address
 - category
 - city
 - country

Administration Schemas Information

Table: category

Columns:

category_id	tinyint UN
name	AI PK varchar(25)
last_update	timestamp

Temporary Tables: Temporary disk tables created: 0
Temporary tables created: 1

Timing (as measured at client side):
Execution time: 0:00:0.01600000

Timing (as measured by the server):
Execution time: 0:00:0.00457000
Table lock wait time: 0:00:0.000000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 510
Rows examined: 2616

Join Statistics

Joins per Type:
Full table scans (Select_scan): 1
Join using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No index used

Other Info:
Event Id: 329
Thread Id: 134

Result 18 Result 19

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

3) LEFT JOIN

```

1 • select film.title , count(*) number_of_actors
2   from film fm
3   left join film_actor fm_act
4     on fm.film_id = fm_act.film_id
5   group by fm.rental_duration
6   order by number_of_actors desc;

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

BEFORE INDEXING

```

1 • select film.title , count(*) number_of_actors
2   from film fm
3   left join film_actor fm_act
4     on fm.film_id = fm_act.film_id
5   group by fm.rental_duration
6   order by number_of_actors desc;

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

AFTER INDEXING

The screenshot shows the MySQL Workbench interface. The main area displays a query editor with the following SQL script:

```
1 • create index rental_duration on film(rental_duration);
2 • select file.title , count(*) number_of_actors
3   from film
4   left join film_actor film_act
5     on film.film_id = film_act.film_id
6   group by film.rental_duration
7   order by number_of_actors desc;
```

Below the query editor, the "Query Statistics" section provides performance metrics:

- Timing (as measured at client side):** Execution time: 0:00:00.01500000
- Timing (as measured by the server):** Execution time: 0:00:00.00662890; Table lock wait time: 0:00:00.000000200

The "Errors" section shows 0 errors and 0 warnings.

The "Columns" section lists columns for the "film" table, including:

- film_id
- title
- description
- release_year
- language_id
- original_language_id
- rental_duration
- rental_rate
- length
- replacement_cost
- rating
- special_features

The "Temporary Tables" section indicates 0 temporary disk tables and 1 temporary table created.

The "Result 15" and "Result 16" tabs are visible at the bottom.

On the right side, there are several toolbars and panels:

- SQLAdditions toolbar: Includes icons for back, forward, search, and jump.
- Automatic context help panel: A tooltip explaining how to toggle automatic help.
- Result Grid, Form Editor, Field Types, and Query Stats panels.
- Execution Plan panel.
- Context Help and Snippets buttons.

4) RIGHT JOIN

The screenshot shows the MySQL Workbench interface with a query editor and a visual explain plan.

Query Editor:

```
from film f
right join film_actor fim_act
on fim.film_id = fim_act.film_id
group by fim.title
order by number_of_actors desc;
```

Visual Explain Plan:

The explain plan illustrates the execution flow:

- Full Index Scan** on the **fim_act** table (566.38 rows) with index **idx_fk_film_id**.
- Unique Key Lookup** on the **fim** table (5888.55 rows) with primary key.
- nested loop**: The **fim_act** table feeds into the **GROUP** stage.
- GROUP**: The **fim** table is joined via **filsort** into the **query_block #1**.
- ORDER**: The results are sorted before being returned.

Schemas: b'akila', newschema, sakila

Tables: actor, address, category, city, country, customer, film, film_actor, film_category, film_text, inventory, language, payment

Administration Schemas

Table: actor

Columns:

actor_id	smallint UN
1	AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

Object Info Session Output

Result 36 Result 37 Result 38 Result 39

SQL Additions

Automatic context help is disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

BEFORE INDEXING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

customer

- Columns
- Indexes
- Foreign Keys
- Triggers
- film
- film_actor

Administration Schemas

Information

Columns:

- film_id
- title
- description
- release_year
- language_id
- original_language_id
- rental_duration
- rental_rate
- length
- replacement_cost
- rating
- special_features

Result 14 x

Object Info Session Output

SQL File 4* SQL File 4* SQL File 5* x

```

1 • select film.title , count(*) number_of_actors
2   from film fml
3   right join film_actor fim_act
4   on fml.film_id = fim_act.film_id
5   group by fml.title
6   order by number_of_actors desc;

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.0000000

Timing (as measured by the server):
Execution time: 0:00:0.0056200
Table lock wait time: 0:00:0.000000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 997
Rows examined: 11921

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1

Index Usage:
At least one Index was used

Other Info:
Event Id: 244
Thread Id: 134

Result 14 x

Read Only Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

AFTER INDEXING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

customer

- Columns
- Indexes
- Foreign Keys
- Triggers
- film
- film_actor

Administration Schemas

Information

Columns:

- film_id
- title
- description
- release_year
- language_id
- original_language_id
- rental_duration
- rental_rate
- length
- replacement_cost
- rating
- special_features

Result 13 x

Object Info Session Output

SQL File 4* SQL File 4* SQL File 5* x

```

1 • create index rental_rate on film(rental_rate);
2 • select film.title , count(*) number_of_actors
3   from film fml
4   right join film_actor fim_act
5   on fml.film_id = fim_act.film_id
6   group by fml.title
7   order by number_of_actors desc;

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.0150000

Timing (as measured by the server):
Execution time: 0:00:0.0050320
Table lock wait time: 0:00:0.000000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 9
Rows sent to client: 997
Rows examined: 11921

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1

Index Usage:
At least one Index was used

Other Info:
Event Id: 241
Thread Id: 134

Result 13 x

Apply Review Context Help Snippets

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Name: Kartik Jolapara

SAPID: 60004200107

DIV: B/B1

ADBMS

Exp4

Aim: To implement Query Monitor



Aim: Implement Query Monitor (Query Execution plan, query statistics)

Theory: with the DB Query Monitor get information availability and the performance of the database as well as the execution time of the database queries. It can help you identify probable causes of performance degradation of your business applications. If the execution time of the queries are higher than usual it can indicate problem with the database.

Tasks can perform:

- ① Compare the execution times of SQL queries.
- ② Identify poor performing queries
- ③ Allow to delete rows of query results.
- ④ Prevent major outages.
- ⑤ compare the outputs.

In short it keeps an eye on important queries. The monitor dashboard is structured to give you an overview of the important metrics of the monitor.

⑥ Optimizing queries with 'EXPLAIN'
The 'EXPLAIN' statement provides information about how MySQL executes the statements.

- ① Explain works with SELECT, DELETE, UPDATE,
INSERT & REPLACE
- ② Explain is useful for examining queries, involving
partitioned tables
- ③ When EXPLAIN is used with an explainable
statements MySQL displays information from
the optimizer about the statement execution plan.
- ④ For SELECT statement, explain produces
additional execution plan information that
can be displayed using show warnings
- ⑤ With Explain you can also see where you
should add indexes to tables so that
the statement executes faster by using indexes
to find rows.
- ⑥ The optimizer trace may sometimes provide
information complementary to that of
EXPLAIN.
- ⑦ EXPLAIN can also be used to obtain information
about the columns in a table

Conclusion: Hence, Query Monitoring was implemented
successfully in MySQL.

Output:

1) SELECT QUERY

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **sakila** database selected.
- SQL Editor:** Contains the following SQL code:

```
USE sakila;
create index length on film(length);
EXPLAIN
select * from film where length = 120;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. The output is:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	film	NULL	ref	length	length	3	const	9	100.00	Using temporary; Using filesort
- SQLAdditions:** A sidebar with various tools and a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Result 3:** A tab labeled "Output" is visible at the bottom.
- Object Info:** Shows "No object selected".

2) NESTED

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **sakila** database selected.
- SQL Editor:** Contains the following SQL code:

```
USE sakila;
EXPLAIN
select concat(first_name, " ", last_name) AS name, email
from customer where customer_id IN
(select customer_id from rental where inventory_id in
(select inventory_id from inventory where film_id in
(select file_id from film_category join category using (category_id) where category.name = "Action"))))
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. The output is:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	category	NULL	ALL	PRIMARY	NULL	NULL	NULL	16	10.00	Using temporary; Using filesort
1	SIMPLE	file_category	NULL	ref	PRIMARY, fk_file_category_category	fk_file_category_category	1	sakila.category.category_id	62	100.00	Using temporary; Using filesort
1	SIMPLE	inventory	NULL	ref	PRIMARY, idx_fk_film_id	idx_fk_film_id	2	sakila.film_category.film_id	4	100.00	Using temporary; Using filesort
1	SIMPLE	rental	NULL	ref	idx_fk_inventory_id, idx_fk_customer_id	idx_fk_inventory_id	3	sakila.inventory.inventory_id	3	100.00	Using temporary; Using filesort
1	SIMPLE	customer	NULL	eq_ref	PRIMARY, customer_id, customer_id, cus_id	PRIMARY	2	sakila.rental.customer_id	1	100.00	Using temporary; Using filesort
- SQLAdditions:** A sidebar with various tools and a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Result 3:** A tab labeled "Output" is visible at the bottom.
- Object Info:** Shows "No object selected".

3) LEFT JOIN

The screenshot shows the MySQL Workbench interface with a query editor window. The SQL tab contains the following code:

```
1 use sakila;
2 EXPLAIN
3 select stf.first_name , stf.last_name , ad.address , ad.district , ad.postal_code , ad.city_id
4 from staff stf
5 left join address ad
6 on stf.address_id = ad.address_id;
```

The results grid shows the execution plan for the EXPLAIN command:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	stf		ALL					2	100.00	
1	SIMPLE	ad		eq_ref	PRIMARY	PRIMARY	2	sakila.stf.address_id	1	100.00	

The right panel displays the Query Profiler with the following message:
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

4) RIGHT JOIN

The screenshot shows the MySQL Workbench interface with a query editor window. The SQL tab contains the following code:

```
1 USE sakila ;
2 explain
3 select fm.title , count(*) number_of_actors
4 from film fm
5 right join film_actor fm_act
6 on fm.film_id = fm_act.film_id
7 group by fm.title
8 order by number_of_actors desc;
9
```

The results grid shows the execution plan for the EXPLAIN command:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	fm_act		index	idx_fk_film_id	idx_fk_film_id	2		5462	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	fm		eq_ref	PRIMARY	PRIMARY	2	sakila.fm_act.film_id	1	100.00	

The right panel displays the Query Profiler with the following message:
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

5) INNER JOIN

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
USE sakila ;
explain
select film.title , count(*) number_of_actors
from film f
inner join film_actor fim_act
on film.film_id = fim_act.film_id
group by film.title
order by number_of_actors desc;
```

The results grid shows the following data:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	fim		index	PRIMARY, idx_film_id	idx_film_id	514		1000	100.00	Using index; Using temporary; Using filesort
1	2	SIMPLE	fim_act		ref	idx_fk_film_id	idx_fk_film_id	2	sakila.fim.film_id	5	100.00	Using index

6) CROSS JOIN

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
USE sakila ;
explain
select film.title , count(*) number_of_actors
from film f
cross join film_actor fim_act
on film.film_id = fim_act.film_id
group by film.title
order by number_of_actors desc;
```

The results grid shows the following data:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	fim		index	PRIMARY, idx_film_id	idx_film_id	514		1000	100.00	Using index; Using temporary; Using filesort
1	2	SIMPLE	fim_act		ref	idx_fk_film_id	idx_fk_film_id	2	sakila.fim.film_id	5	100.00	Using index

Conclusion:

Thus, we implemented and studied query monitor

Name: Kartik Jolapara

SAPID: 60004200107

DIV: B/B1

ADBMS

Exp5

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

Theory:

Kartik Jolapara
60004200107
B1

ADBMS
[Exp 5]

Page No. 1
Date

Aim: Performing Fragmentation (Range, List, Hash, and key) in DBMS design.

Theory:
Myself partitioning is about altering ideally optimizing the way the database engine physically store data. It allows you to distribute position of table data (aka partitions) across the file system based on a set of user-defined rules (aka the partitioning function).

Types of partitioning:

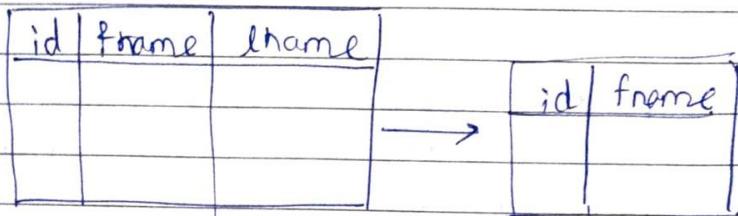
Horizontal partitioning:
horizontal partitioning means that all rows matching the partitioning function will be assigned to different physical partitions.

Key	Room No.
1	66
2	77
3	89
4	44

Key	Room No.
1	66
2	77

II) vertical partitioning.

It allows different table columns to be split into different physical partitions.



Currently MySQL supports horizontal partitioning but not vertical.

Partition Types:-

① Range

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

② List

Partitioning is similar to RANGE except that the partition is selected based on column matching one of discrete values.

③ Hash partitioning:

In hash partitioning, a partition is selected based on the value returned by a user defined expression.

(4) Key

This very similar to HASH partitioning but the hashing function is supplied by MySQL.

Conclusion: From personal experience partitioning is the last part of an optimisation process I'd perform. In general partitioning make the most sense when you are dealing with million record.

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: []

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

```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

PARTITION_NAME	TABLE_ROWS
actions	0
comedy	0
horror	0
romance	0

Hash:

```

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';

```

Result Grid		
	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	1

Key:

```

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';

```

Result Grid		
	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	1

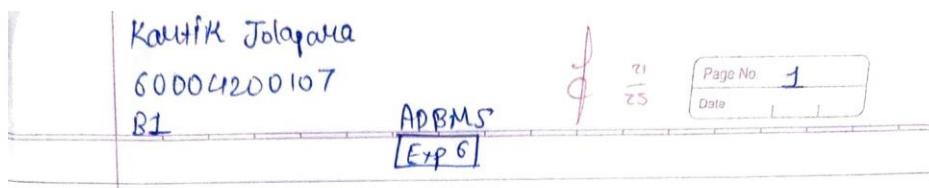
Conclusion:

Thus, we implemented fragmentation using different techniques.

Name: Kartik Jolapara
DIV: B/B1

SAPID: 60004200107

ADBMS Exp6



Aim: Implementation of 2-Phase Protocol

Theory:

The two-phase commit protocol breaks a database commit into two phases to ensure correctness & fault tolerance in a distributed database system.

① Phase I - prepare Phase

- After each slave has locally completed its transaction, it sends a "DONE" message to the controlling site.
- When the controlling site has received "DONE" message from all slaves, it sends a "Prepare" message to slaves.
- The slaves vote on whether they still want to commit or not.
- If a slave ~~wants~~ wants to send a commit, it sends "Ready" message.
- Otherwise, it sends "Not Ready" message.

② Phase II : Commit/Aabort Phase

- After each controlling site has received "Ready" message from all slaves:-
 - ① The controlling site sends "Global commit" msg to slaves.
 - ② The slaves apply the transaction & send a "Commit ACK" message to controlling site.

Disadvantages:

- 1] The major disadvantage of the Two-phase commit protocol is failed when co-ordinator site failure may result in blocking, so a decision either to commit or abort transaction (T) may have to be postponed until co-ordinator recovers again.
- 2] Consider a scenario if a Transaction (T) holds lock on data items of active sites, but amid the execution, if coordinator fails and the active states keep no additional log-record except $\langle \text{need } T \rangle$ like $\langle \text{abort} \rangle$ or $\langle \text{commit} \rangle$. So, it becomes impossible to determine what decision has been made to $\langle \text{commit} \rangle$ / $\langle \text{abort} \rangle$.

Conclusion:

Thus, we successfully studied and implemented the 2-phase Protocol.

Code:



Client

```
import java.io.*; import java.net.*;
public class Client implements Runnable
```

```
{  
    static Socket clientSocket = null;  
    static PrintStream os = null;      static  
    DataInputStream is = null;      static  
    BufferedReader inputLine = null;      static  
    boolean closed = false;      public static  
    void main(String[] args)  
    {  
        int port_number=1111;          String  
        host="localhost";          try {  
            clientSocket = new Socket(host, port_number);  
            inputLine = new BufferedReader(new  
            InputStreamReader(System.in));  
            os = new PrintStream(clientSocket.getOutputStream());  
            is = new DataInputStream(clientSocket.getInputStream());  
        } catch (Exception e)  
        {    System.out.println("Exception occurred : "+e.getMessage());  
    }  
  
        if (clientSocket != null && os != null && is != null)  
        {  
            try  
            {  
                new Thread(new Client()).start();  
            while (!closed)  
            {  
                {  
                    os.println(inputLine.readLine());  
                }  
                os.close();  
            is.close();  
                clientSocket.close();  
            } catch (IOException e)  
            {  
                System.err.println("IOException: " + e);  
            }  
        }  
    }  
}
```

```
        }
    }
}

@SuppressWarnings("deprecation")
public void run() {
    String responseLine;
    try
    {
        while ((responseLine = is.readLine()) != null)
        {
            System.out.println("\n"+responseLine);
            if (responseLine.equalsIgnoreCase("GLOBAL_COMMIT")==true
|| responseLine.equalsIgnoreCase("GLOBAL_ABORT")==true )
            {
                break;
            }
        }
        closed=true;
    }
    catch (IOException e)
    {
        System.err.println("IOException: " + e);
    }
}
}
```

Server

```
import java.io.*; import  
java.net.*; import  
java.util.*;  
  
public class Server {      boolean closed = false,  
inputFromAll = false;  
    List<ClientThread> thread;  
    List<String> data;  
    List<String> decision;  
  
    Server() {  
        thread = new ArrayList<ClientThread>();  
        data = new ArrayList<String>();  
        decision= new ArrayList<String>();  
    }  
    public static void main(String args[])
```

```
{  
    Socket clientSocket = null;  
    ServerSocket serverSocket = null;  
int port_number = 1111;           Server  
server = new Server();           try  
{  
    serverSocket = new ServerSocket(port_number);  
} catch (IOException e) {  
    System.out.println(e);  
}  
while (!server.closed)  
{  
    try {  
clientSocket = serverSocket.accept();  
        ClientThread clientThread = new ClientThread(server,  
clientSocket);  
        (server.thread).add(clientThread);  
        System.out.println("\nNow Total clients are : " +  
(server.thread).size());  
        (server.data).add("NOT_SENT");  
(server.decision).add("NOT_SENT");  
clientThread.start();  
    } catch (IOException e) { }  
}  
try {  
    serverSocket.close();  
} catch (Exception e1) { }  
}  
}  
class ClientThread extends Thread  
{  
    DataInputStream is = null;  
    String line;
```

```
String destClient = "";  
String name;  
PrintStream os = null;  
Socket clientSocket = null;  
String clientIdentity;  
Server server;
```

```
public ClientThread(Server server, Socket clientSocket)
{
    this.clientSocket = clientSocket;
this.server = server;
}

@SuppressWarnings("deprecation")
public void run()
{
    try {
        is = new
DataInputStream(clientSocket.getInputStream());          os = new
PrintStream(clientSocket.getOutputStream());
os.println("Enter your name.");           name = is.readLine();
clientIdentity = name;                  os.println("Welcome " + name + "
to this 2 Phase
Application.\nYou will receive a vote Request now..."); 
os.println("Send Ready or Not Ready after local transaction..");
        while (true)
        {
            line = is.readLine();
if (line.equalsIgnoreCase("NOT READY"))
{
            System.out.println("\nFrom '" + clientIdentity
+ "' : NOT READY\n\nSince NOT READY we will not
wait for inputs from other clients.");
            System.out.println("\nAborted....");

            for (int i = 0; i < (server.thread).size(); i++) {
                ((server.thread).get(i)).os.println("GLOBAL_ABORT"
);
                ((server.thread).get(i)).os.close();
                ((server.thread).get(i)).is.close();
            }
break;
        }
    }
}
```



}



```
if (line.equalsIgnoreCase("READY"))
```

```
{
```

```

        System.out.println("\nFrom '" + clientIdentity + "' :
READY");
        if ((server.thread).contains(this))
        {
            (server.data).set((server.thread).indexOf(this),
"READY");
            for (int j = 0; j < (server.data).size(); j++)
            {
if
(!(((server.data).get(j)).equalsIgnoreCase("NOT_SENT")))
{
                server.inputFromAll = true;
continue;
}
else{
false;
System.out.println("\nWaiting for inputs
from other clients.");
}
}
        if (server.inputFromAll)
{
    System.out.println("All Ready..");
}
}
os.println("VOTE_REQUEST\nPlease enter COMMIT or ABORT to
proceed : ");
line = is.readLine(); if
(line.equalsIgnoreCase("ABORT"))
{
    System.out.println("\nFrom '" + clientIdentity
+ "' : ABORT\n\nSince ABORT we will not wait for inputs from other
clients.");
    System.out.println("\nAborted....");

        for (int i = 0; i < (server.thread).size(); i++) {
((server.thread).get(i)).os.println("GLOBAL_ABORT"

```

```
});  
((server.thread) get(i)).os.close();
```

```

                ((server.thread).get(i)).is.close();
            }
break;
        }
if (line.equalsIgnoreCase("COMMIT")){
    System.out.println("\nFrom '" + clientIdentity + "' :
COMMIT");
    if ((server.thread).contains(this))
    {
        (server.decision).set((server.thread).indexOf(this
), "COMMIT");
        for (int j = 0; j < (server.decision).size(); j++)
        {
if
(!(((server.decision).get(j)).equalsIgnoreCase("NOT_SENT")))
{
    server.inputFromAll = true;
continue;
}
else{
false;
                    System.out.println("\nWaiting for inputs
from other clients.");
                    break;
}
if (server.inputFromAll){
    System.out.println("\n\nCommitted....");
for (int i = 0; i < (server.thread).size(); i++)
{
    ((server.thread).get(i)).os.println("GLOBA
L_COMMIT");
    ((server.thread).get(i)).os.close();
    ((server.thread).get(i)).is.close();
}
break;
}
}
}
}
}

```

```

        server.closed = true;
        clientSocket.close();
    } catch (IOException e) { }

}
}

```

Output:

Scenario: One "not ready"

The image shows four terminal windows arranged in a 2x2 grid, illustrating the 2-phase commit process between three clients: Rishabh, Abhishek, and Yash.

- Top Left Terminal:** Shows the initial state where the total number of clients is 2, then increases to 3. It receives a 'READY' message from Rishabh and waits for others.
- Top Right Terminal:** A client named Rishabh logs in. It sends a 'READY' message to the system and receives a 'VOTE_REQUEST'.
- Bottom Left Terminal:** A client named Abhishek logs in. It sends a 'READY' message to the system and receives a 'VOTE_REQUEST'.
- Bottom Right Terminal:** A client named Yash logs in. It sends a 'NOT READY' message to the system and receives a 'GLOBAL_ABORT' response.

The 'READY' messages indicate that Rishabh and Abhishek are prepared to proceed with the transaction. The 'NOT READY' message from Yash indicates that the transaction will be aborted.

```

Now Total clients are : 2
Now Total clients are : 3
From 'Rishabh' : READY
Waiting for inputs from other clients.
From 'Abhishek' : READY
Waiting for inputs from other clients.
From 'Yash' : NOT READY
Since NOT READY we will not wait for inputs from other clients.
Aborted....
```

```

C:\Users\Admin\OneDrive\Desktop\2phasecommit>java Client
Enter your name.
Rishabh
Welcome Rishabh to this 2 Phase Application.
You will receive a vote Request now...
Send Ready or Not Ready after local transaction..
Ready
VOTE_REQUEST
Please enter COMMIT or ABORT to proceed :
GLOBAL_ABORT
```

```

C:\Users\Admin\OneDrive\Desktop\2phasecommit>java Client
Enter your name.
Abhishek
Welcome Abhishek to this 2 Phase Application.
You will receive a vote Request now...
Send Ready or Not Ready after local transaction..
Ready
VOTE_REQUEST
Please enter COMMIT or ABORT to proceed :
GLOBAL_ABORT
```

```

C:\Users\Admin\OneDrive\Desktop\2phasecommit>java Client
Enter your name.
Yash
Welcome Yash to this 2 Phase Application.
You will receive a vote Request now...
Send Ready or Not Ready after local transaction..
Not ready
GLOBAL_ABORT
```



Scenario: all ready but one “Abort”

The image shows two terminal windows side-by-side. Both windows are titled 'C:\Windows\System32\cmd.e'. The left window shows the following sequence of events:

```
From 'Yash' : READY
All Ready..
From 'Rishabh' : COMMIT
Waiting for inputs from other clients.
From 'Abhishek' : COMMIT
Waiting for inputs from other clients.
From 'Yash' : ABORT
Since ABORT we will not wait for inputs from other clients.
Aborted....
```

The right window shows the following sequence of events:

```
Enter your name.
Rishabh
Welcome Rishabh to this 2 Phase Application.
You will receive a vote Request now...
Send Ready or Not Ready after local transaction..
Ready
VOTE_REQUEST
Please enter COMMIT or ABORT to proceed :
Commit
GLOBAL_ABORT
```

The right window's output indicates that the 'Commit' command was issued, but the 'GLOBAL_ABORT' message was received, suggesting a global abort occurred.

Scenario: all ready and all commit

The image shows two terminal windows side-by-side. Both windows are titled 'C:\Windows\System32\cmd.e'. The left window shows the following sequence of events:

```
Waiting for inputs from other clients.
From 'Yash' : READY
All Ready..
From 'Rishabh' : COMMIT
Waiting for inputs from other clients.
From 'Abhishek' : COMMIT
Waiting for inputs from other clients.
From 'Yash' : COMMIT
Committed....
```

The right window shows the following sequence of events:

```
Enter your name.
Rishabh
Welcome Rishabh to this 2 Phase Application.
You will receive a vote Request now...
Send Ready or Not Ready after local transaction..
Ready
VOTE_REQUEST
Please enter COMMIT or ABORT to proceed :
Commit
GLOBAL_COMMIT
```

The right window's output indicates that the 'Commit' command was issued, and the 'GLOBAL_COMMIT' message was received, suggesting a global commit occurred.

Conclusion:

Thus, we successfully studied and implemented 2PL protocol.

Name: Kartik Jolapara
DIV: B/B1

SAPID: 60004200107

ADBMS Exp7

Aim: Query Execution on an XML database

Theory:

Kartik Jolapara
60004200107

B1

ADBMS
[Exp7]

F 25 Page No. 1 Date

Aim: Query Execution on XML database

Theory:

- ① XML Database is used to store huge amount of information in XML format.
- ② As the use of XML is increasing in every field it is required to have a secured place to store XML documents.
- ③ The data stored in db can be generated using XQuery, serialized and exported into a desired format.
- ④ There are 2 major types of XML databases which are:-

A] XML enabled

The extension provided for conversion of XML documents. It is a relational database.

B] Native XML

Based on the container rather than table format. It can store large amount of XML document & data.

- ⑤ XML makes it easier to express metadata in a portable, reusable format.
- ⑥ Also XML stores the data in plain text format.
- ⑦ It does this so that the data can be easily understood by users.

- ④ For this we have to make sure multiple browser can manage XML documents just like HTML documents.
- ⑤ XML provides a software & hardware independent way of storing, transporting & sharing data.
- ⑥ XML also makes it easier to expand or upgrade to new OS.

Conclusion:

A book catalog consisting of information about books was initially XML.

Code:

```
<html>
<head> <style> table, th,
td { border: 1px solid
black; border-
collapse: collapse;
margin: 5px;
} th, td {
padding: 5px;
} input { margin-
bottom: 5px;
}
</style>
</head>
<body>

<button type="button" onclick="loadXMLDoc()">View
Information about the Books</button>
<br><br>

<table id="data-table"></table>

<script>
    function loadXMLDoc() { var xmlhttp = new
XMLHttpRequest(); xmlhttp.onreadystatechange =
function() { if (this.readyState == 4 &&
this.status == 200) {
```

```
    myFunction(this);
}    };    xmlhttp.open("GET",
"books.xml", true);    xmlhttp.send();
} function myFunction(xml) {    var i;    var xmlDoc =
xml.responseXML;    var
table=<tr><th>Title</th><th>Author</th><th>Genre</th
><th>Price</th><th>Publish
Date</th><th>Description</th></tr>" ;    var x =
xmlDoc.getElementsByTagName("book");
for (i = 0; i <x.length; i++)
{
    table +=
"<tr><td>" +
x[i].getElementsByTagName("title") [0].childNodes[0].n
odeValue +      "</td><td>" +
x[i].getElementsByTagName("author") [0].childNodes[0].
nodeValue +
"</td><td>" +
x[i].getElementsByTagName("genre") [0].childNodes[0].n
odeValue +      "</td><td>" +
x[i].getElementsByTagName("price") [0].childNodes[0].n
```

```

odeValue +
"</td><td>" +

x[i].getElementsByTagName("publish_date")[0].childNodes[0].nodeValue +
"</td><td>" +

x[i].getElementsByTagName("description")[0].childNodes[0].nodeValue +
"</td></tr>";

}

document.getElementById("data-table").innerHTML = table;

}

</script>

</body>
</html>

```

XML used:

```

<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications with
    XML.</description>
  </book>

```

```
<book id="bk102">
    <author>Ralls, Kim</author>
```

```
<title>Midnight Rain</title>
<genre>Fantasy</genre>
<price>5.95</price>
<publish_date>2000-12-16</publish_date>
<description>A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.</description>
</book>
<book id="bk103">
    <author>Corets, Eva</author>
    <title>Maeve Ascendant</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-11-17</publish_date>
    <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
</book>
<book id="bk104">
    <author>Corets, Eva</author>
    <title>Oberon's Legacy</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-03-10</publish_date>
    <description>In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.</description>
</book>
<book id="bk105">
    <author>Corets, Eva</author>
    <title>The Sundered Grail</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-09-10</publish_date>
    <description>The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.</description>
</book>
<book id="bk106">
    <author>Randall, Cynthia</author>
    <title>Lover Birds</title>
    <genre>Romance</genre>
    <price>4.95</price>
    <publish_date>2000-09-02</publish_date>
    <description>When Carla meets Paul at an ornithology conference, tempers fly as feathers get ruffled.</description>
</book>
<book id="bk107">
    <author>Thurman, Paula</author>
    <title>Splish Splash</title>
    <genre>Romance</genre>
    <price>4.95</price>
    <publish_date>2000-11-02</publish_date>
```

```
<description>A deep sea diver finds true love twenty  
thousand leagues beneath the sea.</description>  
</book>  
<book id="bk108">
```

```
<author>Knorr, Stefan</author>
<title>Creepy Crawlies</title>
<genre>Horror</genre>
<price>4.95</price>
<publish_date>2000-12-06</publish_date>
<description>An anthology of horror stories about roaches, centipedes, scorpions and other insects.</description>
</book>
<book id="bk109">
    <author>Kress, Peter</author>
    <title>Paradox Lost</title>
    <genre>Science Fiction</genre>
    <price>6.95</price>
    <publish_date>2000-11-02</publish_date>
    <description>After an inadvertant trip through a Heisenberg Uncertainty Device, James Salway discovers the problems of being quantum.</description>
</book>
<book id="bk110">
    <author>O'Brien, Tim</author>
    <title>Microsoft .NET: The Programming Bible</title>
    <genre>Computer</genre>
    <price>36.95</price>
    <publish_date>2000-12-09</publish_date>
    <description>Microsoft's .NET initiative is explored in detail in this deep programmer's reference.</description>
</book>
<book id="bk111">
    <author>O'Brien, Tim</author>
    <title>MSXML3: A Comprehensive Guide</title>
    <genre>Computer</genre>
    <price>36.95</price>
    <publish_date>2000-12-01</publish_date>
    <description>The Microsoft MSXML3 parser is covered in detail, with attention to XML DOM interfaces, XSLT processing, SAX and more.</description>
</book>
<book id="bk112">
    <author>Galos, Mike</author>
    <title>Visual Studio 7: A Comprehensive Guide</title>
    <genre>Computer</genre>
    <price>49.95</price>
    <publish_date>2001-04-16</publish_date>
    <description>Microsoft Visual Studio 7 is explored in depth, looking at how Visual Basic, Visual C++, C#, and ASP+ are integrated into a comprehensive development environment.</description>
</book>
</catalog>
```

Output:

View Information about the Books					
Search a title <input type="text" value="midnight"/>					
Search an author <input type="text"/>					
Search a genre <input type="text"/>					
Search a price <input type="text"/>					
Search a Publish Date <input type="text"/>					
Title	Author	Genre	Price	Publish Date	Description
XML Developer's Guide	Gambardella, Matthew	Computer	44.95	2000-10-01	An in-depth look at creating applications with XML.
Midnight Rain	Ralls, Kim	Fantasy	5.95	2000-12-16	A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.
Maeve Ascendant	Corets, Eva	Fantasy	5.95	2000-11-17	After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.
Oberon's Legacy	Corets, Eva	Fantasy	5.95	2001-03-10	In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.
The Sundered Grail	Corets, Eva	Fantasy	5.95	2001-09-10	The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.
Lover Birds	Randall, Cynthia	Romance	4.95	2000-09-02	When Carla meets Paul at an ornithology conference, tempers fly as feathers get ruffled.
Splash Splash	Thurman, Paula	Romance	4.95	2000-11-02	A deep sea diver finds true love twenty thousand leagues beneath the sea.
Creepy Crawlies	Knorr, Stefan	Horror	4.95	2000-12-06	An anthology of horror stories about roaches, centipedes, scorpions and other insects.
Paradox Lost	Kress, Peter	Science Fiction	6.95	2000-11-02	After an inadvertent trip through a Heisenberg Uncertainty Device, James Salway discovers the problems of being quantum.
Microsoft .NET: The Programming Bible	O'Brien, Tim	Computer	36.95	2000-12-09	Microsoft's .NET initiative is explored in detail in this deep programmer's reference.
MSXML3: A Comprehensive Guide	O'Brien, Tim	Computer	36.95	2000-12-01	The Microsoft MSXML3 parser is covered in detail, with attention to XML DOM interfaces, XSLT processing, SAX and more.
Visual Studio 7: A Comprehensive Guide	Galos, Mike	Computer	49.95	2001-04-16	Microsoft Visual Studio 7 is explored in depth, looking at how Visual Basic, Visual C++, C#, and ASP+ are integrated into a comprehensive development environment.

Result

Title	Author	Genre	Price	Publish Date	Description
Midnight Rain	Ralls, Kim	Fantasy	5.95	2000-12-16	A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Conclusion:

Thus, we studied and implemented Query Execution on XML database.

Name: Kartik Jolapara

SapID:60004200107

Batch: B/B1

ADBMS Exp8

Aim: Data handling using JSON. (Display user information from JSON file downloaded from Mobile)

Theory:

Kartik Jolapara
60004200107
B1

ADBMS
Exp 8

P 21
25
Page No. 1
Date: 1/1/2023

Aim: Data handling using JSON.

Theory:

- JSON stands for JavaScript Object Notation
- In JSON is basically used for data transmission in web applications.
- JSON is extension of JS language
- JSON can be used for storing & exchanging data on the web.
- It is used as alternative to XML.
- JSON is a light-weight, text based data interchange format.
- Text can read and used as a data format by any programming language
- JSON is language dependent & can be used with modern programming languages.
- All popular programming languages have parsing code & support for JSON data on them
- JSON is self-describing and easy to understand
- JSON supports - pair values, arrays, lists, object sequences.

Conclusion:

Thus we studied and implemented data handling in JSON.

Implementing Queries-

Selecting a DB

```
test> use reddit;
switched to db reddit
reddit> |
```

Showing all Databases

```
reddit> show dbs;
admin    100.00 KiB
config   60.00 KiB
local    72.00 KiB
reddit    8.00 KiB
reddit> |
```

Dropping a Database

```
reddit> use twitter;
switched to db twitter
twitter> db.dropDatabase();
{ ok: 1, dropped: 'twitter' }
twitter> |
```

Creating a collection

```
reddit> db.createCollection("posts");
{ ok: 1 }
reddit> |
```

Dropping a collection

```
reddit> db.comments.drop();
true
reddit> |
```

Inserting multiple documents

```
reddit> db.posts.insertMany([{"title": "Funny Meme", "image": "https://wikipedia.com/troll_face", "votes": 123, "posted_by": "lca_tejas"}, {"title": "Damn reddit", "text": "Your reddit is damn reddit", "votes": 12, "posted_by": "meetp90"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6389fde4880a8e67b5db4a3b"),
    '1': ObjectId("6389fde4880a8e67b5db4a3c")
  }
}
reddit> |
```

Updating documents

```
reddit> db.posts.update({ posted_by: "meetp90" }, { $set: { title: "How to get stuff back!" } });
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
reddit> db.posts.find();
[ {
  _id: ObjectId("6389fde4880a8e67b5db4a3b"),
  title: 'Funny Meme',
  image: 'https://wikipedia.com/troll_face',
  votes: 123,
  posted_by: 'lca_tejas'
},
{
  _id: ObjectId("6389fde4880a8e67b5db4a3c"),
  title: 'How to get stuff back!',
  text: 'Your reddit is damn reddit',
  votes: 12,
  posted_by: 'meetp90'
} ]
```

Querying the documents

```

reddit> db.posts.find({ votes: { $gt: 1000 }, image: { $exists: true } });
[ {
    _id: ObjectId("638a0009880a8e67b5db4a3f"),
    title: 'Soggy cat',
    image: 'https://preview.redd.it/cxjsjx27g7j81.jpg?auto=webp&s=d09b1a5e785562455e3661a186087e398c0546d0',
    votes: 33213,
    posted_by: 'lca_tejas'
}
]
reddit> db.posts.find({ votes: { $gt: 1000 }, text: { $exists: true }, posted_by: 'meetp90' });
[ {
    _id: ObjectId("6389ff86880a8e67b5db4a3e"),
    title: 'Emojis bad',
    text: 'Emojis bad, please laugh',
    votes: 1212,
    posted_by: 'meetp90'
},
{
    _id: ObjectId("638a0009880a8e67b5db4a40"),
    title: 'We did it',
    text: 'We did it reddit! We broke the stock markets around the world and diluted all the world currency!',
    votes: 12134,
    posted_by: 'meetp90'
}
]

```

Deleting documents

```

reddit> db.posts.remove({ votes: { $lt: 100 } });
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }

```

Importing json

```

# mongoimport --db reddit --collection posts --jsonArray --authenticationDatabase admin -u root -p pass@123 --file ./posts.json
2022-12-02T14:42:23.925+0000      connected to: mongodb://localhost/
2022-12-02T14:42:23.932+0000      3 document(s) imported successfully. 0 document(s) failed to import.
# |
{
    _id: ObjectId("638a0ecfecbf380e4d3fafc3"),
    title: 'Hass lo guys, very funny',
    image: 'https://www.google.com/url?sa=i&url=https%3A%2F%2Fknowyourmeme.com%2Fmemes%2Fwalter-white-breaks-down&psig=A
OvVaw3-rrXAvEUTQclpJMPUNmWy&ust=1670075371754000&source=images&cd=vfe&ved=0CA0QjRxqFwoTCNDptOaJ2_sCFQAAAAAdAAAAABAD',
    votes: 2234,
    posted_by: 'kunaljx86'
},
{
    _id: ObjectId("638a0ecfecbf380e4d3fafc4"),
    title: 'No idea',
    text: 'What is this post even',
    votes: 3234,
    posted_by: 'lca_tejas'
},
{
    _id: ObjectId("638a0ecfecbf380e4d3fafc5"),
    title: 'bruh!',
    image: 'bruh sound effect (x100)',
    votes: 22342,
    posted_by: 'meetp90'
}

```

Exporting as json

```

# mongoexport --db reddit --collection posts --out ./posts_db.json --authenticationDatabase admin -u root -p pass@123
2022-12-02T14:46:53.394+0000      connected to: mongodb://localhost/
2022-12-02T14:46:53.402+0000      exported 8 records

# cat posts_db.json
[{"_id": {"$oid": "6389fde4880a8e67b5db4a3b"}, "title": "Funny Meme", "image": "https://wikipedia.com/troll_face", "votes": 123, "posted_by": "lca_tejas"}, {"_id": {"$oid": "6389ff86880a8e67b5db4a3d"}, "title": "Jharkhand is not real", "text": "Jharkhand is not real guys. It is a propaganda tool", "votes": 3321, "posted_by": "lca_tejas"}, {"_id": {"$oid": "6389ff86880a8e67b5db4a3e"}, "title": "Emojis bad", "text": "Emojis bad, please laugh", "votes": 1212, "posted_by": "meetp90"}, {"_id": {"$oid": "638a0009880a8e67b5db4a3f"}, "title": "Soggy cat", "image": "https://preview.redd.it/cxjsjx27g7j81.jpg?auto=webp&s=d09b1a5e785562455e3661a186087e398c0546d6", "votes": 33213, "posted_by": "lca_tejas"}, {"_id": {"$oid": "638a0009880a8e67b5db4a40"}, "title": "We did it", "text": "We did it reddit! We broke the stock markets around the world and dilluted all the world currency!", "votes": 12134, "posted_by": "meetp90"}, {"_id": {"$oid": "638a0ecfecbf380e4d3fafc3"}, "title": "Hass lo guys, very funny", "image": "https://www.google.com/url?sa=i&url=https%3A%2F%2Fknowyourmeme.com%2Fmemes%2Fwalter-white-breaks-down&psig=A0vVaw3-rrXAvEUTQclpJMPUNmW&ust=1670075371754000&source=images&cd=vfe&ved=0CA0QjRxqFwoTCNDpt0aJ2_sCFQAAAAAdAAAAABAD", "votes": 2234, "posted_by": "kunaljx86"}, {"_id": {"$oid": "638a0ecfecbf380e4d3fafc4"}, "title": "No idea", "text": "What is this post even", "votes": 3234, "posted_by": "lca_tejas"}, {"_id": {"$oid": "638a0ecfecbf380e4d3fafc5"}, "title": "bruh!", "image": "bruh sound effect (x100)", "votes": 22342, "posted_by": "meetp90"}]

```

Conclusion:

Thus, we successfully implemented data handling using JSON.

Name: Kartik Jolapara
DIV: B/B1

SAPID: 60004200107

ADBMS Exp9

Aim: Processing of Spatial and Temporal data

Theory:

Kartik Jolapara
60004200107
B1

ADBMS
[Exp 9]

Page No. 1
Date: 1/1/2023

Aim: Processing of spatial & temporal data

Theory:

Temporal Database

- ① Database that store information which is time dependent in one way or another one called Temporal Database
- ② Temporal DB store information about states of real world across time
- ③ Temporal DB is a database with built-in support for handling data involving time
- ④ Examples:
 - 1] Health care systems
 - here history of patients health is required
 - The post record of operation is also necessary
 - 2] Insurance systems
 - we need to keep a track of claims, accident history
 - Also the time when policies are in effect.

Spatial Database

- ① A spatial database is a database that deals.

with information associated with geographical locations such as cities, towns etc.

- ① A spatial database is optimized to store & query data representing objects.
- ② These are the objects which are defined in a geometric space.
- ③ Spatial data is majorly of 3 types.
 - vector data:- data represented as discrete pts, lines & polygons
 - Raster data:- data represented as matrix of square cells.
 - Image data:- data typically form of images.

Conclusion:

Thus, we studied & implemented spatial & temporal data.

Queries:

Temporal:

```
mysql> select * from 2022_july where start_time < "04:00:00";
+-----+-----+-----+-----+-----+-----+
| start_date | start_time | distance | mode | confidence | Places |
+-----+-----+-----+-----+-----+-----+
| 02-07-2022 |          | 18716 | IN_PASSENGER_VEHICLE | MEDIUM |          |
| 06-07-2022 | 03:23:42 | 1669 | MOTORCYCLING | LOW |          |
| 06-07-2022 | 03:35:46 | 11483 | IN_TRAIN | LOW |          |
| 06-07-2022 | 03:58:13 | 1069 | MOTORCYCLING | MEDIUM |          |
| 08-07-2022 | 03:11:39 | 461 | MOTORCYCLING | LOW |          |
| 08-07-2022 | 03:26:15 | 1099 | IN_PASSENGER_VEHICLE | LOW |          |
| 08-07-2022 | 03:42:42 | 7548 | IN_TRAIN | MEDIUM | Anhad Misal |
| 11-07-2022 | 03:48:22 | 1609 | IN_BUS | MEDIUM | Kandivali Bus Station (W) |
| 13-07-2022 | 03:19:59 | 905 | WALKING | LOW |          |
| 13-07-2022 | 03:48:02 | 1896 | IN_BUS | LOW | Kandivali Railway Station (W) |
| 14-07-2022 | 03:54:54 | 8949 | IN_BUS | HIGH |          |
| 15-07-2022 | 03:48:49 | 8500 | IN_PASSENGER_VEHICLE | LOW |          |
| 16-07-2022 | 03:44:35 | 9838 | IN_BUS | MEDIUM | Laxmi Industrial Colony |
| 18-07-2022 | 03:51:31 | 1292 | IN_BUS | LOW |          |
| 18-07-2022 | 03:57:47 | 9400 | IN_BUS | LOW | Anhad Misal |
| 19-07-2022 | 03:54:30 | 10085 | IN_BUS | MEDIUM | Lotus Business Park |
| 20-07-2022 | 03:32:29 | 1721 | IN_BUS | MEDIUM | Bajaj Municipal School |
| 20-07-2022 | 03:47:40 | 9487 | IN_TRAIN | MEDIUM | Andheri Police Station |
| 21-07-2022 | 03:46:02 | 8698 | UNKNOWN_ACTIVITY_TYPE | LOW | Sun-n-Sand Hotel |
| 23-07-2022 | 03:45:05 | 11590 | IN_PASSENGER_VEHICLE | LOW |          |
| 25-07-2022 | 03:34:52 | 10629 | IN_BUS | MEDIUM | Doolally Taproom - Andheri |
| 26-07-2022 | 03:52:48 | 10072 | IN_BUS | MEDIUM | Doolally Taproom - Andheri |
| 27-07-2022 | 03:48:55 | 9483 | IN_BUS | LOW | Doolally Taproom - Andheri |
| 28-07-2022 | 03:29:48 | 9599 | IN_BUS | MEDIUM | RikÄ? - Terrace Bar & Grill |
| 29-07-2022 | 03:31:35 | 1948 | MOTORCYCLING | LOW | Kandivali Bus Station (W) |
| 29-07-2022 | 03:47:34 | 9579 | IN_TRAIN | MEDIUM |          |
+-----+-----+-----+-----+-----+-----+
26 rows in set (0.00 sec)
```

Spatial:

```
mysql> select * from 2022_july where distance>10000;
+-----+-----+-----+-----+-----+-----+
| start_date | start_time | distance | mode | confidence | Places |
+-----+-----+-----+-----+-----+-----+
| 02-07-2022 |           | 18716 | IN_PASSENGER_VEHICLE | MEDIUM |          |
| 06-07-2022 | 03:35:46 | 11483 | IN_TRAIN | LOW |          |
| 13-07-2022 | 13:44:17 | 11855 | IN_TRAIN | LOW |          |
| 19-07-2022 | 03:54:30 | 10085 | IN_BUS | MEDIUM |          |
| 21-07-2022 | 18:59:35 | 41236 | UNKNOWN_ACTIVITY_TYPE | LOW |          |
| 23-07-2022 | 03:45:05 | 11590 | IN_PASSENGER_VEHICLE | LOW |          |
| 24-07-2022 | 06:56:09 | 10674 | MOTORCYCLING | LOW |          |
| 25-07-2022 | 03:34:52 | 10629 | IN_BUS | MEDIUM |          |
| 26-07-2022 | 03:52:48 | 10072 | IN_BUS | MEDIUM |          |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.03 sec)
```

```
mysql> select * from 2022_july where Places="Doolally Taproom - Andheri";
+-----+-----+-----+-----+-----+
| start_date | start_time | distance | mode | confidence | Places |
+-----+-----+-----+-----+-----+
| 25-07-2022 | 03:34:52 | 10629 | IN_BUS | MEDIUM | Doolally Taproom - Andheri |
| 26-07-2022 | 03:52:48 | 10072 | IN_BUS | MEDIUM | Doolally Taproom - Andheri |
| 27-07-2022 | 03:48:55 | 9483 | IN_BUS | LOW | Doolally Taproom - Andheri |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Data:

start_date	start_time	distance	mode	confidence	Places
02-07-2022		18716	IN_PASSENGER_VEHICLE	MEDIUM	
06-07-2022	03:23:42	1669	MOTORCYCLING	LOW	
06-07-2022	03:35:46	11483	IN_TRAIN	LOW	
06-07-2022	03:58:13	1069	MOTORCYCLING	MEDIUM	
08-07-2022	03:11:39	461	MOTORCYCLING	LOW	
08-07-2022	03:26:15	1099	IN_PASSENGER_VEHICLE	LOW	
08-07-2022	03:42:42	7548	IN_TRAIN	MEDIUM	Anhad Misal
11-07-2022	03:48:22	1609	IN_BUS	MEDIUM	Kandivali Bus Station (W)
11-07-2022	04:11:17	9363	IN_TRAIN	MEDIUM	Agarkar Chowk
11-07-2022	04:39:15	2125	IN_SUBWAY	LOW	SAMMOHI BY MOKSHA & HIRAL
11-07-2022	04:48:18	896	IN_PASSENGER_VEHICLE	LOW	Trumpet Sky Lounge
11-07-2022	12:22:26	2972	IN_PASSENGER_VEHICLE	LOW	McDonald's
11-07-2022	12:48:32	9467	IN_TRAIN	MEDIUM	Kandivali Bus Depot
11-07-2022	13:10:00	2263	IN_PASSENGER_VEHICLE	LOW	State Bank of India
13-07-2022	03:19:59	905	WALKING	LOW	
13-07-2022	03:48:02	1896	IN_BUS	LOW	Kandivali Railway Station (W)
13-07-2022	04:04:12	9348	IN_TRAIN	MEDIUM	Agarkar Chowk / Pinky Cinema
13-07-2022	04:33:49	3270	IN_PASSENGER_VEHICLE	LOW	Lotus Business Park
13-07-2022	13:06:26	5297	IN_BUS	LOW	Villa Decor - Premium Bed, Bath & Mattresses Store
13-07-2022	13:44:17	11855	IN_TRAIN	LOW	Kandivali Station (W)
13-07-2022	14:15:27	1969	IN_PASSENGER_VEHICLE	MEDIUM	State Bank of India
14-07-2022	03:54:54	8949	IN_BUS	HIGH	
14-07-2022	04:39:02	578	WALKING	MEDIUM	UK Realty
14-07-2022	14:12:01	6005	IN_BUS	LOW	Yoko Sizzlers
14-07-2022	14:58:50	4087	UNKNOWN_ACTIVITY_TYPE	LOW	Xth Central Mall
15-07-2022	03:48:49	8500	IN_PASSENGER_VEHICLE	LOW	
15-07-2022	04:41:11	9105	IN_BUS	LOW	Rik? - Terrace Bar & Grill
15-07-2022	14:15:09	5691	IN_BUS	MEDIUM	D Mart
15-07-2022	15:08:26	4933	IN_BUS	MEDIUM	La Pino'z Pizza Kandivali
16-07-2022	03:44:35	9838	IN_BUS	MEDIUM	Laxmi Industrial Colony
16-07-2022	13:08:07	2965	MOTORCYCLING	LOW	Andheri Station (W)
16-07-2022	13:32:44	9292	IN_TRAIN	MEDIUM	Kandivali West
16-07-2022	13:53:02	2074	IN_PASSENGER_VEHICLE	LOW	La Pino'z Pizza Kandivali
17-07-2022	08:38:08	880	MOTORCYCLING	LOW	Malad Industrial Estate
17-07-2022	10:23:08	1492	IN_PASSENGER_VEHICLE	LOW	Aadhar Center
17-07-2022	10:57:46	1294	MOTORCYCLING	MEDIUM	La Pino'z Pizza Kandivali
17-07-2022	11:38:53	5829	MOTORCYCLING	LOW	State Bank Of India ATM
18-07-2022	03:51:31	1292	IN_BUS	LOW	
18-07-2022	03:57:47	9400	IN_BUS	LOW	Anhad Misal

Conclusion:

We successfully implemented spatial and temporal queries to make the database more efficient for the user.

Kantik Jolapara
60004200107
B1

ADBMS
Exp 10

23
25

Page No.
Data

I

Aim: Case study on Database security issue and patents related to it.

Patent: US. 11263335 B2 United States

Inventor: Rajesh Krishnaswami Pathmegowthu

Title: Integrated system and method for sensitive data security

Publishing year: 2019

Abstract: A system and a method are provided for integrating a sensitive data discovery engine, a data anonymization engine, a data monitoring module and a data retirement module and managing sensitive data security its lifecycle. The system generator and distributes one or more template including the sensitive data discovery intelligence with metadata discovery results, and data security rules to the DAE, the DMM and the DRM developed on error data sources.

claims Exploration: A system for integrating and managing security of sensitive data comprised of computer program instructions defined by modules of an integrated platform to run and execute one or more applications. A system for integrating and managing security of sensitive data this includes a sensitive data discovering engine that determines sensitive

data using match operations in a scanning pathway on data in each of plurality of similarity and variety of data sources and application.

Conclusion:

After performing a detailed analysis on sensitive data security and its solutions by using integrated systems, the issue and claims mentioned in patent were learned and explained.

Reference:

- Reyesh, KP (2019). Integrated system and method for sensitive data security. US1263335B2 United States.