|  |
| --- |
|  |

**LAB 7: Indexing Telecom Data for Performance**

|  |  |
| --- | --- |
| **Track** | Data & AI |
| **Role** | Software Development - Backend |
| **Level** | Level 2 |
| **Course** | Intermediate SQL |
| **Scope** | * Analyze existing telecom data for structure and query patterns. * Develop a tailored indexing strategy for telecom data. * Implement indexing with minimal downtime. * Perform performance testing and benchmarking. |
| **Prerequisite** | * Understanding of indexing concepts. * Familiarity with SQL and database management systems (e.g., MySQL, PostgreSQL). * Access to telecom data and analysis tools. |

**Objective**

To improve telecom data query performance by implementing and optimizing various indexing techniques, ensuring efficient data retrieval and reduced execution times while maintaining scalability​

**Tasks**

**Dataset (also available in resources):**

1. **Basic Indexing Techniques**
   * Single-Column Index
   * Composite Index
   * Unique Index
   * Covering Index
   * Full-Text Index
2. **Advanced Indexing Techniques**
   * Partial Index
   * Index for Date-Based Queries
   * Multi-Column Index for Sorting
   * Bitmap Index (If Using Supported DBMS)
3. **Index Maintenance and Optimization**
   * Drop and Recreate a Composite Index
   * Rebuild Fragmented Indexes
   * Reorganize Indexes
   * Monitor Fragmentation Levels in Indexes
   * Drop an Unused Index
4. **Analyzing and Monitoring Index Performance**
   * Monitor Index Usage with EXPLAIN
   * Analyze Query Performance Before and After Index Creation
   * Optimize Queries with Index Hints
   * Index Columns with High Cardinality

**Outcomes**

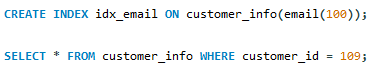
* Improved query performance with reduced execution times.
* Optimized resource utilization during data access.
* Enhanced user experience through faster response times.
* Scalable indexing framework for future data growth.
* Comprehensive documentation of the indexing process.
* Increased stakeholder knowledge on indexing benefits and maintenance.

**Solution**

1. **Basic Indexing Techniques**

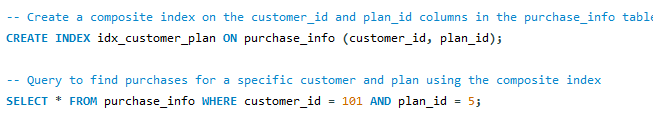
* Single-Column Index:
  + A computer code with blue and yellow text

    Description automatically generated with medium confidenceCreate an index on the customer\_id column in the call\_data\_records table to speed up customer-based lookups.
  + Output:
  + Create an index on the email column in the customer\_info table to speed up email-based lookups.



* + Output:

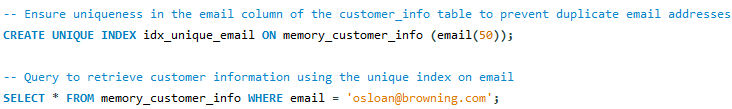


* Composite Index:
  + Create a composite index on the customer\_id and plan\_id columns in the purchase\_info table to optimize queries that filter by both customer and plan.
  + Output:
  + Create a composite index on the plan\_id and amount\_paid columns in the purchase\_info table to optimize queries that filter by both service plan and payment amount.



* + Output:



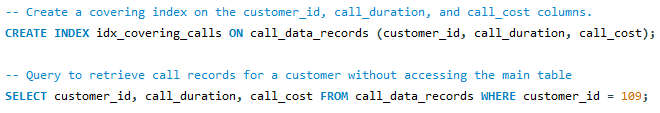
* Unique Index:
  + Ensure uniqueness in the email column of the customer\_info table to prevent duplicate email addresses.
  + Output:
  + Ensure uniqueness in the phone\_number column of the customer\_info table to prevent duplicate phone numbers.



* + Output:



* Covering Index:
  + Create a covering index on the customer\_id, call\_duration, and call\_cost columns in the call\_data\_records table to optimize queries that retrieve all of these columns.

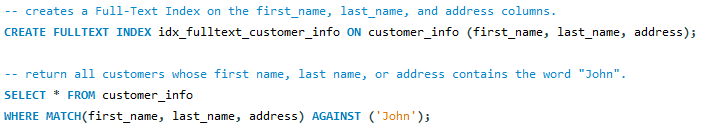
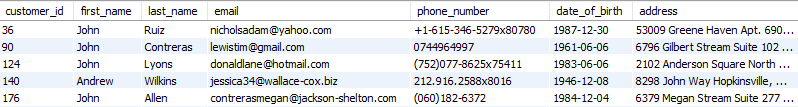


* + Output:
  + Create a covering index on the purchase\_id, plan\_id, and amount\_paid columns in the purchase\_info table to optimize queries that retrieve all of these columns.



* Output:



* Full-Text Index:
  + Create and use a Full-Text Index on the customer\_info table to efficiently perform text-based searches on multiple columns.
  + Output:
  + Create a full-text index on the address and email columns in the customer\_info table to efficiently search for customers by their address or email.

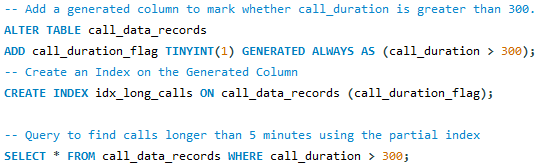


* + Output:

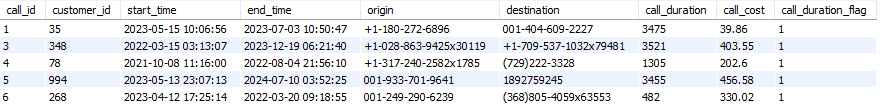


1. **Advanced Indexing Techniques**

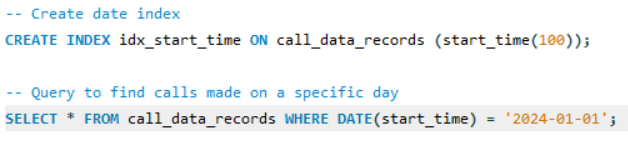
* Partial Index
  + In MySQL, **partial indexes** (i.e., indexes with a WHERE condition) are not supported. Partial indexing is available in other databases like PostgreSQL.
  + Use a generated column and then index it
  + **Add a generated column** to mark whether call\_duration is greater than 300.



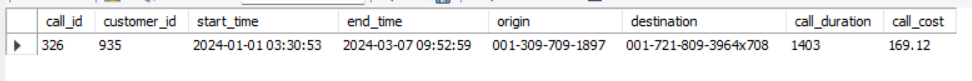
* Output:



* Date-Based Index
  + Create an index on the start\_time column in the call\_data\_records table to optimize queries that filter based on call start time.



* Output:



* Multi-Column Index for Sorting
  + Create an index on the purchase\_date and amount\_paid columns in the purchase\_info table to optimize queries that order purchases by date and amount.

A close-up of a white background

Description automatically generated

* Output:

A screenshot of a data

Description automatically generated

* Bitmap Index
  + If using a DBMS that supports bitmap indexing (like Oracle), create a bitmap index on the plan\_id column in the purchase\_info table for plans with limited unique values.

A close up of text

Description automatically generated

* Output:



1. **Index Maintenance and Optimization**

* Drop and Recreate a Composite Index
  + Drop the existing idx\_customer\_plan index and recreate it with additional columns for better query optimization.

A screenshot of a computer code

Description automatically generated

* Output:



* Rebuild Fragmented Index
  + Rebuild the idx\_customer\_id index on the call\_data\_records table to improve performance after heavy data modification.

A screenshot of a computer code

Description automatically generated

* Output:



* Reorganize Indexes
  + Reorganize indexes in the call\_data\_records table to improve query performance after significant data modification.

A close-up of a computer screen

Description automatically generated

* Output:



* Monitor Fragmentation Levels
  + Use the ANALYZE command to check for fragmentation in the indexes of the call\_data\_records table.

A screenshot of a computer

Description automatically generated

* Output:

A close-up of a computer screen

Description automatically generated

* Drop an Unused Index
  + Drop the idx\_old\_index from the service\_plans table if it's no longer needed for query optimization.

A screenshot of a computer program

Description automatically generated

* Output:



1. **Analyzing and Monitoring Index Performance**

* Monitor Index Usage with EXPLAIN
  + Use the EXPLAIN command to check if the idx\_customer\_plan index is being used in a query.

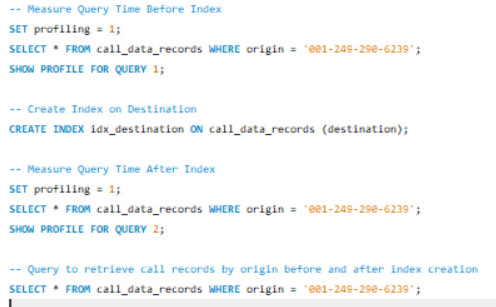
A close-up of a computer screen

Description automatically generated

* Output:



* Analyze Query Performance Before and After Index Creation
  + Measure the execution time of a query before and after creating an index on the destination column of the call\_data\_records table.
  + Step 1: Measure Query Time Before Index
  + Step 2: Create Index on Destination
  + Step 3: Measure Query Time After Index



* Output:



* Optimize Queries with Index Hints
  + Force the use of the idx\_customer\_plan index in a query to optimize query performance manually.

A screenshot of a computer program

Description automatically generated

* Output:



* Index Columns with High Cardinality
  + Create an index on the phone\_number column in the customer\_info table, where each value is unique, to improve lookup performance for specific phone numbers.

A screenshot of a computer code

Description automatically generated

* Output:

****

# (Additional) Task to do

1. Create a Unique Index on Customer Email.
2. Create a Partial Index on Active Customers.
3. Rebuild an Index and Measure Query Performance Before and After.
4. Create a Full-Text Index on Product Descriptions.
5. Monitor Index Fragmentation and Reorganize the Index if Fragmented.