A logo of a university

Description automatically generated**COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CMPS 451 – Database Management Systems**

**Fall 2023**

**Project Report – Phase 2**

**1-Name, IDs and emails of the group members:**

1-**Name**: Ziad Abdelazim - **ID**: 201905466 -**Email:** [za1905466@qu.edu.qa](mailto:za1905466@qu.edu.qa)

2-**Name**: Yousef Mohamed- **ID**: 202003576-**Email**: [ym2003576@qu.edu.qa](mailto:ym2003576@qu.edu.qa)

3-**Name**: Marwan Hashish- **ID**: 201701546-**Email**: [mh1701546@qu.edu.qa](mailto:mh1701546@qu.edu.qa)

***Instructor: Dr. Rehab Duwairi***

**2-A description of backend and frontend design choices:**

1. **Backend-Design:**

**We modified our phase 1 tables such that we created constraints for attributes that were primary key and unique, so it is easier to identify them when displaying the Index metadata.**

The chosen tables we used: STUDENT – SECTION-STUDENT TAKES (relation between student & section tables).

In this part we were focusing on creating our catalog schema:

**A diagram of a student section

Description automatically generated**

**-**Display Table: This option allows the user to view the content of a given table.

-Display Metadata and statistics: Allows the user to have three options of viewing: either to display table metadata , display column and display index, it will be explained in detail in section 3 of this report where we discuss the summary of our project metadata.

-Display Cost Estimator: Allows the user to choose a set of SELECT operations and calculate its cost (Using a primary key and equality operator-Using a primary key with range operator -Using a non-primary key and equality operator-Using a non-primary key with range operator), and

1. **Frontend-Design:**

**We used JFrame in Java swing to create our UI design, it contains the following:**

A screenshot of a computer

Description automatically generated

**Why did we use JFrame in Java swing?**

1- Rich GUI Components

2-Cross-Platform Compatibility

3-Customization

4-Event-Drive Programming

Using JDBC connections to databases in Java:

JDBC offers several advantages for Java developers, including:

1. **Cross-platform compatibility:** JDBC allows Java applications to connect to databases running on different operating systems.
2. **Support for multiple database vendors:** JDBC provides a standard interface for connecting to various database vendors.
3. **Relative ease of use:** JDBC's API is well-structured and makes it relatively straightforward to use.
4. **Performance optimization:** JDBC supports efficient data access mechanisms, including prepared statements.
5. **Maturity and stability:** JDBC is a well-established and widely supported technology.
6. **Deep integration with Java ecosystem:** JDBC integrates seamlessly with other Java technologies.
7. **Widespread adoption:** JDBC is the industry standard for Java database connectivity.

In essence, JDBC simplifies data access for Java applications and offers a versatile and reliable tool for developing database-driven applications.

**3-Summary of the metadata:**

For every table, the user can choose one of 3 services (Display Data- Cost Estimator- Display Metadata & Statistics), when the user chooses to display data, a table full of the table attributes and records will be displayed. As for when the user chooses the third option to display Metadata & statistics a window with 3 buttons (Display Table-Display Column-Display Index), In the Display table we recorded:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute*** | NUM\_ROWS | AVG\_ROW\_LEN | AVG\_SPACE | BLOCKS | EMPTY\_BLOCKS | SAMPLE\_SIZE |
| ***Explanation*** | Estimates the number of rows affected by a particular operation in the execution plan, aiding in understanding the anticipated data volume and cardinality for that step. | Estates the average row length (in bytes) for a specific operation in the execution plan, helping assess the memory and storage requirements for the result set | Represents the estimated average space used per row in a specific operation's result set. | Represents the estimated number of database blocks accessed or affected by an operation in the execution plan, offering insights into the I/O and storage impact of that query operation. | Estimates the number of empty database blocks that an operation may encounter, aiding in the identification of possible storage inefficiencies. | Is the size of the sample used by the query optimizer in Oracle to gather statistics and formulate the execution plan. It's essential because the optimizer often uses statistical sampling to estimate attributes like Num\_Rows and AVG\_ROW\_LEN. |

**And in Display Column we recorded:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Attribute*** | COLUMN\_NAME | | COLUMN\_ID | DATA\_TYPE | DATA\_LEN | NULLABLE | DATA\_DESTINCT | NUM\_NULL | CHAR\_LEN | DATA\_PRCESISION |
| ***Explanation*** | Serves as the identifier for a column, providing a unique name for each column within the table's schema. | Denotes the position or order of a column within a database table, assisting in establishing the sequence in which columns are arranged in the table's schema. | | Specifies the data type assigned to a column, defining the type of data it can store, such as VARCHAR2, NUMBER, DATE, etc. This attribute characterizes the nature of data held within the column. | Defines the capacity of a column to hold data. For character-based data types (e.g., VARCHAR2), it represents the maximum character count allowed, and for numeric types (e.g., NUMBER), it reflects the number of digits used to represent data | Is a binary attribute in database metadata that denotes whether a column permits NULL values. It is typically represented as 'Y' for allowing NULLs and 'N' for disallowing them. This attribute defines the null value constraint for a column. | Represents the number of distinct or unique values present in the column.  It provides insights into the diversity of data within the column. | Records the count of NULL values within the column.  It helps in understanding the completeness of data and potential data quality issues. | Is relevant for character-based data types and represents the maximum character length for the column.  It is distinct from DATA\_LEN, as it measures character count rather than bytes or digits. | Is primarily applicable to numeric data types like NUMBER.  It indicates the maximum number of significant digits in the data, offering precision information for numeric value. |

**And in Display Index we recorded:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute*** | INDEX\_NAME | NUM\_ROWS | UNIQUNESS | STATUS | INDEX\_TYPE | JOIN\_INDEX |
| ***Explanation*** | Stores the name or identifier of the index.  It provides a unique name for each index within the database schema, making it easy to reference. | Represents the estimated number of rows or records in the table that the index is associated with.  It offers insights into the cardinality of the indexed data, helping the query optimizer make efficient query execution decisions. | Indicates whether the index enforces uniqueness constraints on the data.  It is often represented as 'UNIQUE' if the index enforces uniqueness and 'NON-UNIQUE' if it doesn't. | Reflects the status or condition of the index.  Common values include 'VALID' (the index is functional), 'INVALID' (the index is not functional), or other states that describe the index's health**.** | Specifies the type or method of indexing used.  Examples include 'B-tree' for standard indexes, 'Bitmap' for bitmap indexes, or other index types used to optimize query performance | Indicates whether the index is designed for optimizing join operations.  It is often represented as 'YES' if the index is specifically created for joining tables and 'NO' otherwise. |

**4- Types of queries supported in our program:**

* Display table
* Display table metadata
* Display column metadata
* Display index metadata
* Selection using a primary key and equality operator.
* Selection using a primary key with range operator
* Selection using a non-primary key and equality operator.
* Selection using a non-primary key with range operator.
* Selection using equi-join.

**5- Link to the code:**

[**https://github.com/yousef-ym2003576/DBMS-PROJECT**](https://github.com/yousef-ym2003576/DBMS-PROJECT)

**6- Screen shots of metadata:**

**STUDENT Table Metdata:**

A screenshot of a computer

Description automatically generated

**STUDENT Column Metdata:**

A screenshot of a computer

Description automatically generated

**STUDENT INDEX Metdata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES Table Metdata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES COLUMN Metdata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES INDEX Metdata:**

A screenshot of a computer

Description automatically generated

**7-Screen shots of runs for every type of query:**

**A screenshot of a computer screen

Description automatically generatedMain window:**

A screenshot of a service

Description automatically generated**STUDENT Sub-main window:**

**STUDENT DISPLAY DATA window:**

A screenshot of a computer

Description automatically generated

**STUDENT DISPLAY TABLE metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT DISPLAY COLUMN metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT DISPLAY INDEX metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT SELECT USING A PRIMARY KEY AND EQUALITY OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT SELECT USING A PRIMARY KEY WITH RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT SELECT USING A NON- PRIMARY KEY WITH EQUALITY OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT SELECT USING A NON- PRIMARY KEY WITH RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT USING EQUI-JOIN:**

A screenshot of a computer

Description automatically generated

**MAIN WINDOW:**

A screenshot of a computer screen

Description automatically generated

**STUDENT\_TAKES Submian:**

A screenshot of a service

Description automatically generated

**STUDENT \_TAKES DISPLAY DATA:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES TABLE metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES COULMN metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES INDEX metadata:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES SELECT USING A PRIMARY KEY AND EQUALITY OPREATOR:**A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES SELECT USING A PRIMARY KEY WITH A RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES SELECT USING A NON- PRIMARY KEYAND EQUALITY OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES SELECT USING A NON- PRIMARY KEY WITH RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**STUDENT\_TAKES JOIN USING EQUI-JOIN:**

A screenshot of a computer

Description automatically generated

**MAIN WINDOW:**

A screenshot of a computer screen

Description automatically generated

**SECTION SUBMIAN WINDOW:**

A screenshot of a service

Description automatically generated

**SECTION DISPLAY DATA:**A screenshot of a computer

Description automatically generated

**SECTION TABLE METDATA:**

A screenshot of a computer

Description automatically generated

**SECTION COLUMN METDATA:**

A screenshot of a computer

Description automatically generated

**SECTION INDEX METDATA:**

A screenshot of a computer

Description automatically generated

**SECTION COULMN METDATA:**

A screenshot of a computer

Description automatically generated

**SECTION SELECT USING A PRIMARY KEY AND EQUALITY OPREATOR:**

A screenshot of a computer

Description automatically generated

**SECTION SELECT USING A PRIMARY KEY WITH RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**SECTION SELECT USING NON- PRIMARY KEY AND EQUALITY OPREATOR:**

A screenshot of a computer

Description automatically generated

**SECTION SELECT USING NON-PRIMARY KEY WITH RANGE OPREATOR:**

A screenshot of a computer

Description automatically generated

**SECTION JOIN USING EQUI-JOIN OPREATOR:**

A screenshot of a computer

Description automatically generated