EXPLORE AI ACADEMY

Optimising queries

Optimising queries

Why should we optimise queries?

Optimising SQL queries ensures that they execute in the most **efficient** manner possible.

Optimisation **enhances performance** and **reduces system resource usage**, which becomes critical as datasets increase.

Leveraging database **indexing**, applying effective **data filtering**, and using appropriate query **techniques** help us to optimise queries.

Database indexing

> Database **indexing** involves creating **specific structures** on the **database table** to improve the **speed** of query operations.

Without indexes, the database engine scans every row of a table to find the relevant rows for a query, which is **time-consuming** and **inefficient**, especially with large datasets.

Indexes allow the system to **quickly locate** and retrieve the required data, leading to significantly **faster** query execution.

Think of SQL indexes as a **book's glossary**, guiding the system **directly** to the data it needs, skipping redundant row checks and enhancing query **speed**.

Using indexing effectively

> Indexes can be **ineffective** if **not appropriately considered** during query design or database alterations.

Inefficient queries can slow down data retrieval and **negatively impact overall system performance**.

Hence, a **balance** is vital between optimising read and write operations and maintaining the effectiveness of indexes.

Using **functions** on indexed columns with **WHERE** or **JOIN** is **ineffective** as it computes the function for every row in the table instead of using the index.

Keep indexes **updated**. As datasets grow and change, the effectiveness of an index might diminish.

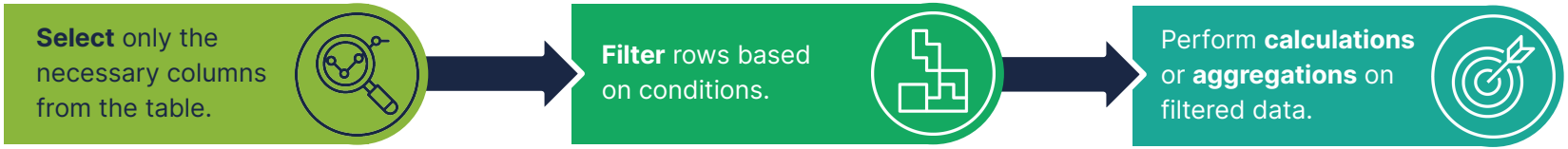
We must strike a **balance** because while indexes speed up data **retrieval**, they can slow down **write** operations.

Optimising query performance

> To ensure **efficiency** and **optimal resource utilisation** we need to **limit** the **data processed** by queries. This involves refining the **sequence** of operations to handle only necessary data.

Processing large volumes of data, especially when performing calculations on unnecessary or irrelevant data, can be resource-intensive and slow. **Streamlining** the data to be processed can significantly improve query performance.

The optimal sequence of operations



Query readability

> SQL queries should be easily **interpretable** by humans. **Clear** and **legible** queries **reduce errors** during execution, **simplify future modifications**, and foster **collaboration** among team members.

How do we achieve this?

Descriptive naming conventions

Table and column names should clearly indicate their contents, and naming conventions should be consistently applied.

Structured formatting

Consistently apply coding conventions, indentations, and clear segmentations.

Appropriate comments

Always provide brief explanations for complex sections or crucial operations in the code.

Optimising queries

Query maintainability

> Ensuring SQL maintainability refers to the practice of writing SQL code in a manner that is **organised** and easy to **understand**.

Maintainable SQL code is essential because it **reduces** the **time** and **effort** required for future **debugging**, facilitates **easier updates**, and ensures the **sustainability** of data projects

An organised and comprehensible codebase also enhances **collaboration** among team members.

Use straightforward queries: Write clear and concise SQL statements that avoid unnecessary complexities.

Implement Common Table Expressions (CTEs): CTEs allow for breaking down complex queries into simpler, more readable parts which can be referenced multiple times within the main query.

Organise and document code: Ensure that every segment of the code is logically organised and thoroughly documented, making it easier for others (or your future self) to understand the intent and functionality.

Query adaptability

> We should intentionally design SQL queries and structures to ensure that they are **flexible**, **scalable**, and ready to **meet evolving industry needs**.

The technology landscape and business requirements **change rapidly**. If our SQL projects lack adaptability, they **risk becoming obsolete or requiring significant rework.**

How to ensure adaptability

Design the database schema, relationships, and queries so that they can be **easily modified**, **extended**, or **scaled** to accommodate changing requirements or data structures.

This involves using **modular structures**, **normalisation**, and **avoiding hard coding** specific values or conditions.

Optimising queries

Mastering query optimisation

Database **indexing** helps to improve the **speed** of query operations.

We should process data by **filtering first**, **then perform necessary calculations**. This ensures resource effectiveness and faster results.

Queries should be designed with clear intent, emphasising **readability** for easy comprehension, **maintainability** for future adjustments, and **adaptability** to meet evolving project needs.