**PROG8410-23S-Section 2 – NoSQL Database Implementation**

**LAB 2 Assignment**

1. **Perform all JOIN operations on different tables.**
   1. **INNER JOIN**

The INNER JOIN returns only the rows from the first table and the second table where there is a match on the condition.

**Tables Used:**

A picture containing text, screenshot, font, design

Description automatically generated

**Query Execution & Results (Query Results show the rows from the employees & departments table where there is a match with the department id):**

A screenshot of a computer

Description automatically generated

* 1. **LEFT OUTER JOIN**

The LEFT OUTER JOIN returns all the row from the left table and matching rows from the right table.

**Tables Used:**

A picture containing text, font, screenshot

Description automatically generated

**Query Execution & Results (Query Results show all rows from the category table & matching rows from product table and when there is no match found in the product table then it will return NULL values):**

A screenshot of a computer

Description automatically generated

* 1. **RIGHT OUTER JOIN**

The RIGHT OUTER JOIN returns all the row from the right table and matching rows from the left table.

**Tables Used:**

A picture containing text, screenshot, font, design

Description automatically generated

**Query Execution & Results (Query Results show all rows from the employees table & matching rows from departments table and when there is no match found in the product table then it will return NULL values):**

A screenshot of a computer

Description automatically generated

* 1. **FULL OUTER JOIN**

The FULL OUTER JOIN returns all the rows from the left and the right table including all the row that doesn’t match as well.

**Tables Used:**

A picture containing text, screenshot, font, design

Description automatically generated

**Query Execution & Results (Query Results show all rows from the employees table & departments table including those rows which doesn’t match):**

A screenshot of a computer

Description automatically generated

* 1. **SELF JOIN**

The SELF JOIN is like other Joins but then the table will be joined by itself by using different alias name. The two instances of the same table will be related using this join condition.

**Table Used:**

A screenshot of a computer

Description automatically generated with low confidence

**Query Execution & Results (Query Results show 2 instances of the same employees table):**

A screenshot of a computer

Description automatically generated with medium confidence

**Citation** – Self & our class Notes

<https://www.w3schools.com/sql/sql_join_self.asp>

1. **Difference between Normalization VS. Denormalization:**

|  |  |
| --- | --- |
| **Normalization** | **Denormalization** |
| Normalization is a method where the information is splitting into different tables to minimize duplication and irregularity thus achieving the data integrity | Denormalization is a method where the information is collaborated into a single table, to make the retrieval process quicker |
| Normalization is used in Online Transaction Processing (OLTP) which helps the process of insertion, deletion and updating the inconsistent data quicker | Denormalization is used in Online Analytical Processing (OLAP) which helps the process of doing research and investigation quicker |
| In Normalization, it is easy to maintain the data integrity | In Denormalization, it is harder to maintain the data integrity |
| Normalization eliminates duplication of information | Denormalization maximizes the duplicate information |
| Normalization maximizes the number of tables and joins | Denormalization minimizes the number of tables and joins |
| In Normalization, the disk space is optimized as it eliminates the duplicate data | In Denormalization, the disk space is wasted because identical information is stored in different places |

**Citation** – <https://www.linkedin.com/pulse/normalization-vs-denormalization-rohit-prasad>

1. **Discovering NoSQL: Features, Types, and Examples**

NoSQL database doesn’t deal with any tables or tabular relationships, unlike relational databases for persisting and pulling the data. This database existed since 1960 but is not referred to the same term. One of the core benefits of this database is that any number of new columns can be added dynamically during the run time without touching the table structure also this database provides an easy way to expand the database size mainly when dealing with many users and huge information volumes are handled and this would be the best fit in big data space and real-time cloud and web applications. NoSQL databases prioritize speed, partition tolerance, and availability at the expense of consistency. This flexibility allows them to solve a variety of business problems. However, NoSQL databases do have their drawbacks. They are primarily designed for data storage and are not ideal for transaction management, where relational databases are better suited. Additionally, the lack of widely adopted business standards for NoSQL can make managing larger databases challenging, especially since no well-known GUI mode tools are available. Lastly, some NoSQL systems store data as JSON, resulting in the creation of large documents.

**Features**

1. **Multiple data model compatibility**

NoSQL database is compatible with various data models which in turn will be easy for data handling. This accepts unstructured, semi-structured and structured at the same speed and ease. This database best suits that application which requires specific data models and can be used efficiently in agile methodology. As this allows the same data can be used in different types of data models without creating any separate database.

1. **Enhanced scalability and availability**

As a user, I have found that NoSQL solutions can be a game-changer for cloud applications. With their serverless, peer-to-peer architecture and consistent properties among all nodes, they offer simplified scalability without compromising on performance. In fact, NoSQL databases can greatly improve read and write speeds and ensure continuous availability, making them a top choice for handling massive data volumes. Additionally, the sharding technique used for horizontal scaling allows for efficient preservation of data ordering. Overall, NoSQL is highly adaptable and can handle complex data management needs with ease.

1. **Global data distribution**

With cutting-edge NoSQL databases, data distribution at a global scale is no longer a daunting task. This is made possible through multiple cloud regions and data centres, which allow for seamless read-and-write operations across various locations. Unlike relational databases that rely on centralized, location-dependent applications for their read and write operations, globally distributed NoSQL databases distribute multiple copies of data to ensure that information is readily available wherever it is needed, resulting in minimal wait times.

1. **Minimal downtime**

NoSQL databases have become a top choice for businesses due to their robustness and minimal downtime. The serverless architecture behind these databases ensures business continuity, while the creation of multiple copies of data across nodes provides extra security. In case of a node malfunction, access to the data can still be granted through a different node's copy, making NoSQL databases a reliable option for companies.

**Type of NoSQL databases**

1. **Key-value pair**

Storing data in a key-value pair NoSQL database is based on a unique key and a specific data item. This type of storage can include new sets of key-value pairing captured as objects. However, it may not be the most efficient method for querying or updating partial values. Column

1. **Column**

In the world of databases, column arrangement plays a crucial role in the efficient management of data. In NoSQL databases, this arrangement is based on the column family, and keys are used to point to several columns. These types of databases are commonly used for managing applications such as business intelligence, data warehouses, library card catalogues, and customer relationship management (CRM) systems. With their flexibility and scalability, NoSQL databases are becoming increasingly popular in today's data-driven world.

1. **Document**

Document-oriented NoSQL databases store data in key-value pairs using JSON or XML. They're flexible and commonly used for blogging, CMS, eCommerce, and real-time analytics. They use versioned documents but aren't ideal for complex transactions. Popular examples include MongoDB, CouchDB, Riak, Amazon SimpleDB, and Lotus Notes.

1. **Graph**

Graph NoSQL databases use a flexible graph model to store entities and relationships as nodes and edges. They're great for logistics, social networks, and spatial data analysis. Neo4J, Infinite Graph, and FlockDB are popular solutions.

1. **Multi-model**

Multi-model NoSQL databases offer flexibility and reduce data redundancy, allowing Agile programming. They can transform data and have a shared backend for consistent inter-model data. OrientDB, ArangoDB, and MarkLogic Server are good options for complex projects with multiple data views.

**Examples of NoSQL**

1. **MongoDB**

MongoDB is a popular database system known for its flexibility and ease of use. It can store different types of data and allows users to search for specific information. It also has a replication feature that ensures data is always available, even if there is a problem with the main copy. MongoDB supports different types of indexes, which makes it easy to find specific data quickly.

1. **Apache CouchDB**

CouchDB is an open-source NoSQL database that stores documents as JSON and uses JavaScript for indexing. It has a schema-free data model for easy record management and is backed by an active dev community. It's optimized for document storage, data synchronization, and offline-first setups. CouchDB is relied on for creating scalable, reliable infrastructure.

1. **Oracle NoSQL DB**

Managing data effectively is vital, and Oracle NoSQL Database is an excellent solution for this purpose. This robust system can handle key-value and JSON table data models and is adaptable for on-premises or cloud use. It guarantees reliable response times, top-notch security, high availability, data replication, cost-effective pricing, and serverless scaling. With the ability to access it using different development languages, you can analyse data natively with parallel scalability and cross-collection queries.

1. **Riak**

Riak is a NoSQL database solution that is open-source and highly distributed. It was developed by Basho Technologies and written in Erlang. Riak has automatic data distribution and built-in fault-tolerance replication, which make it stand out from other solutions in this space. It's suitable for web, mobile, and cloud applications. There's a free, open-source version of Riak, as well as a commercial version for enterprise use. Riak's fault tolerance is due to its high distribution across several nodes and masterless implementation, which helps avoid a single point of failure. Users generally prefer Riak for building cloud file systems and high-volume read-and-write applications.

1. **Objectivity InfiniteGraph**

InfiniteGraph is a graph database that can handle complex and interconnected datasets. It can identify hidden trends and patterns in data quickly. The database is scalable and can handle small or large datasets with ease. It is cloud-powered and can be used in different environments. InfiniteGraph is ideal for industries such as governance, healthcare, telecommunications, cybersecurity, finance, manufacturing, and networking. It can execute difficult queries like locating all paths or the shortest path between two items with ease.

**Citation** – <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-nosql/#:~:text=Such%20databases%20are%20mainly%20used,Cassandra%2C%20HBase%2C%20and%20Hypertable>.