Denormalization

by Sophia

WHAT'S COVERED

In this lesson, you will explore the need to denormalize a database, in three parts. Specifically, this lesson covers:

1. Need for Denormalization
2. Invoice Example
3. Analytical Databases

1. Need for Denormalization

The process of denormalization involves restoring a database to lower normalization levels to introduce redundancy.

KEY CONCEPT

A normalized database minimizes redundancy and ensures data integrity by organizing data efficiently. A **denormalization** process, on the other hand, intentionally duplicates data or reintroduces redundancy to increase query performance, simplify data retrieval, or optimize certain operations.

Denormalization is commonly used to improve read performance at the expense of data modification efficiency in scenarios where read performance is of greater importance. By denormalizing data, you can store it in a format that optimizes running queries and reports by reducing complex joins. This approach can be particularly advantageous in applications where fast data retrieval is critical, such as reporting systems or data warehouses.

HINT

Denormalization can introduce data inconsistencies if redundant data is not handled properly. Denormalization must be weighed against the benefits of improved query performance and data maintenance complexities.

Denormalization should also be used with caution and selectively since excessive denormalization can compromise data integrity and hinder database scalability and flexibility.

Denormalization's ability to improve query performance lies in the way it reduces the number of tables and therefore the number of joins. Although an optimal transactional database should be at least in 3NF, the further you normalize, the more tables are created. You have to join the tables to generate useful information from the database. The more joins you have, the more input/output operations and processing are required. Although most databases can handle this processing quite effectively, it can become more of a challenge for much larger databases.

There are also some data anomalies that may not make sense to split off.

> EXAMPLE You saw in an earlier lesson how you could normalize to 3NF by pulling out city, state, and ZIP data into a separate table. However, depending on the data and how it will be used, that might not always be the best course of action. Yes, it's true that creating that separate table will reduce some redundancy, but it could actually slow down operations if a lot of searching is required because the extra joins create extra processing labor compared to looking something up in a single table.

TERM TO KNOW

Denormalization

A process that intentionally duplicates data or reintroduces redundancy to increase query performance, simplify data retrieval, and optimize certain operations.

2. Invoice Example

You can also have situations where you want to store pre-aggregated data or derived data. For example, an invoice table can be denormalized by directly storing precalculated total amounts and item details. When retrieving and reporting invoices, this would improve query performance.

In a normalized invoice database, invoices and items are typically stored separately. Detailed information about individual invoice items, including quantities, prices, and other information, would be listed in the invoice items table. For a complete view of an invoice and its associated items, you must join the invoice and invoice items tables.

> EXAMPLE Denormalization could be applied in read-heavy scenarios where frequent invoice retrieval and reporting are important, by directly including aggregated or precalculated data in the invoice table. Your invoice table might include a “total_amount” column that stores the sum of all item amounts for that invoice. Further, you could denormalize the invoice table by holding concatenated strings of item details (e.g., “Item A (Qty: 2), Item B (Qty: 1), Item C (Qty: 3)”).

This denormalization would improve query performance for invoice details significantly. Calculations and complex joins would be reduced when generating invoices or running reports that use invoice data. Data that has been denormalized must be consistent with data that has been normalized. The invoice and item tables are updated according to appropriate procedures to maintain data integrity.

You could also have a temporary denormalized table with data stored in a format where you may have repeating groups of columns. This type of query may be impossible to generate the data required purely

through SQL and may need other programming languages to store that information.

3. Analytical Databases

An **analytical database** is a database that exists primarily to generate queries and reports based on historical data and is typically optimized for that purpose. Its counterpart is a **transactional database**, which is a database that exists mainly to accept and store new transactions and is typically optimized to reduce redundancy and ensure data integrity.

When you do not need to worry about the constant data insertion, update, or deletion, denormalization will not be an issue. Therefore, analytical databases are frequently denormalized because it speeds up reporting. In contrast, transactional database are usually well normalized, at least through 3NF, because it speeds up data entry and ensures data integrity.

A **data warehouse** is a centralized and organized repository of data that pulls from multiple sources to create a big data lookup pool that can be queried to gain enterprise-wide insights. It's a type of analytical database and shares many of the same characteristics, including the benefits of denormalization.

Data warehouses typically maintain tables that summarize or aggregate data with precomputed metrics such as sums, averages, or counts. Denormalizing data warehouses can be beneficial in specific scenarios where read-heavy operations are prevalent since they are built to support analytical and reporting tasks. Denormalized versions of the original data can be used to generate summary tables, enabling users to access frequently requested metrics more quickly without recalculating them from raw data. Many business information (BI) tools that companies use to query data warehouses work better with flat, denormalized data structures.

TERMS TO KNOW

Analytical Database
: A database that is used primarily to generate reports based on data that doesn't frequently change. It is optimized for fast querying.

Transactional Database
: A database that is used primarily to accept and store new transactions. It is optimized for fast data input.

Data Warehouse
: A centralized data repository collected from various sources, optimized for analysis, reporting, and decision making.

SUMMARY

In this lesson, you learned data **denormalization** involves reverting from higher to lower levels of normalization in a database to introduce redundancy. Normalized databases minimize redundancy and ensure data integrity by organizing data efficiently. Denormalization, however, duplicates data or reintroduces redundancy to improve query performance, simplify data retrieval, or optimize certain

operations. In situations where read performance is more important than data modification efficiency, denormalization is commonly used. By denormalizing data, complex joins can be reduced, and query speed can be increased by storing it in a format that is suitable for specific queries or reports. It is often used in data warehouses, reporting systems, and business intelligence (BI) applications where faster data retrieval is essential.

In the **invoice example**, you learned that there are trade-offs associated with denormalization when it comes to query performance. A high level of data redundancy requires careful management to ensure data consistency, and it can result in data integrity issues if not managed properly. The purpose of denormalization is to optimize specific read-heavy operations by using it selectively and judiciously. Denormalization should be balanced against the potential complexities of data maintenance when weighing its benefits. To denormalize data, ensure it aligns with the overall goals and objectives of the application or data warehouse. This is done by considering the specific requirements and performance considerations of the database system. As you learned regarding **analytical databases**, if you are not as worried that data redundancies or anomalies will occur and are more worried about performance, having denormalized data that has gone through a transactional database may not be an issue.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

Analytical Database
A database that is used primarily to generate reports based on data that doesn't frequently change. It is optimized for fast querying.

Data Warehouse
A centralized data repository collected from various sources, optimized for analysis, reporting, and decision making.

Denormalization
A process that intentionally duplicates data or reintroduces redundancy to increase query performance, simplify data retrieval, and optimize certain operations.

Transactional Database
A database that is used primarily to accept and store new transactions. It is optimized for fast data input.