Reading: Indexes, partitions, and other ways to optimize

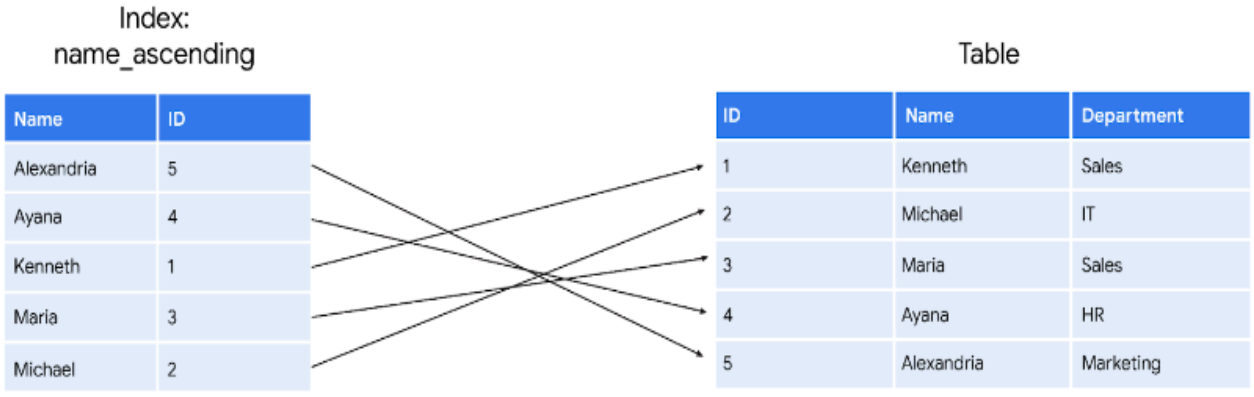
Optimization for data reading

One of the continual tasks of a database is reading data. Reading is the process of interpreting and processing data to make it available and useful to users. As you have been learning, database optimization is key to maximizing the speed and efficiency with which data is retrieved in order to ensure high levels of database performance. Optimizing reading is one of the primary ways you can improve database performance for users. Next, you will learn more about different ways you can optimize your database to read data, including indexing and partitioning, queries, and caching.

Indexes

Sometimes, when you are reading a book with a lot of information, it will include an index at the back of the book where that information is organized by topic with page numbers listed for each reference. This saves you time if you know what you want to find– instead of flipping through the entire book, you can go straight to the index, which will direct you to the information you need.

Indexes in databases are basically the same– they use the keys from the database tables to very quickly search through specific locations in the database instead of the entire thing. This is why they're so important for database optimization– when users run a search in a fully indexed database, it can return the information so much faster. For example, a table with columns ID, Name, and Department could use an index with the corresponding names and IDs.



Name	ID
Alexandria	5
Ayana	4
Kenneth	1
Maria	3
Michael	2

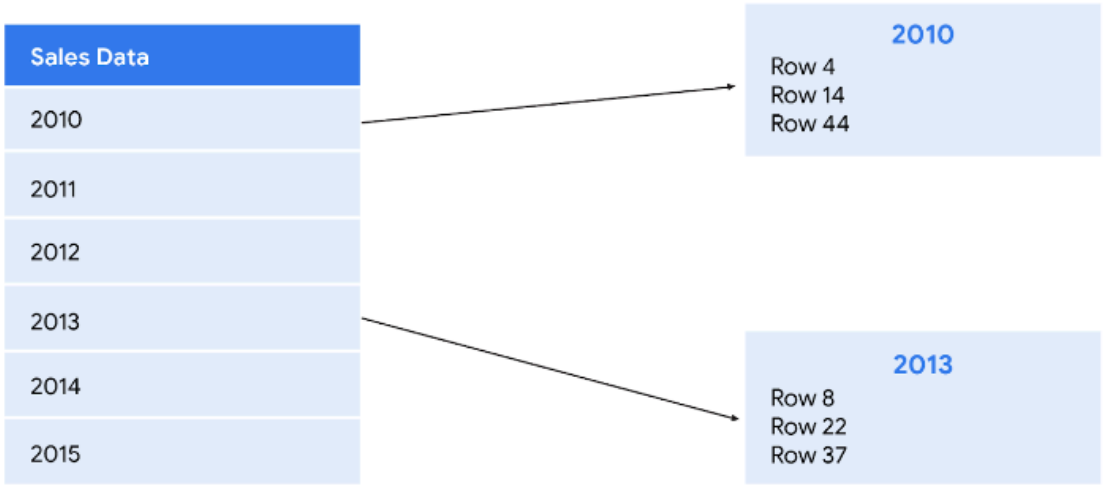
ID	Name	Department
1	Kenneth	Sales
2	Michael	IT
3	Maria	Sales
4	Ayana	HR
5	Alexandria	Marketing

Now the database can easily locate the names in the larger table quickly for searches using those IDs from the index.

Partitions

Data partitioning is another way to speed up database retrieval. There are two types of partitioning: vertical and horizontal. Horizontal partitioning is the most common, and involves designing the database so that rows are organized by logical groupings instead of stored in columns. The different rows are stored in different tables– this reduces the index size and makes it easier to write and retrieve data from the database.

Instead of creating an index table to help the database search through the data faster, partitions split larger, unwieldy tables into much more manageable, smaller tables.



In this example, the larger sales table is broken down into smaller tables– these smaller tables are easier to query because the database doesn't need to search through as much data at one time.

Other optimization methods

In addition to making your database easier to search through with indexes and partitions, you can also optimize your actual searches for readability or use your system's cached memory to save time retrieving frequently used data.

Queries

Queries are requests for data or information from a database. In many cases, you might have a collection of queries that you run regularly; these might be automated queries that generate reports, or regular searches made by users.

If these queries are not optimized, they can take a long time to return results to users and take up database resources in general. There a few things you can do to optimize queries:

1. **Consider the business requirements:** Understanding the business requirements can help you determine what information you really need to pull from the database and avoid putting unnecessary strain on the system by asking for data you don't actually need.
2. **Avoid using SELECT* and SELECT DISTINCT:** Using SELECT* and SELECT DISTINCT causes the database to have to parse through a lot of unnecessary data. Instead, you can optimize queries by selecting specific fields whenever possible.
3. **Use INNER JOIN instead of subqueries:** Using subqueries causes the database to parse through a large number of results and then filter them, which can take more time than simply JOINing tables in the first place.

Additionally, you can use pre-aggregated queries to increase database read functionality. Basically, pre-aggregating data means assembling the data needed to measure certain metrics in tables so that the data doesn't need to be re-captured every time you run a query on it.

If you're interested in learning more about optimizing queries, you can check out Devart's article on SQL Query Optimization.

Caching

Finally, the cache can be a useful way to optimize your database for readability. Essentially, the cache is a layer of short-term memory where tables and queries can be stored. By querying the cache instead of the database system itself, you can actually save on resources. You can just take what you need from the memory.

For example, if you often access the database for annual sales reports, you can save those reports in the cache and pull them directly from memory instead of asking the database to generate them over and over again.

Key takeaways

This course has focused a lot on database optimization and how you, as a BI professional, can ensure that the systems and solutions you build for your team continue to function as efficiently as possible. Using these methods can be a key way for you to promote database speed and availability as team members access the database system. And coming up, you're going to have opportunities to work with these concepts yourself!
