**Хранит данных в строковом формате (Row-Store Databases) – MySQL, PostgreSQL, …**

Row-store databases—also called transactional databases—are **designed to be efficient at processing transactions: INSERTs, UPDATEs, and DELETEs**. Popular open source row-store databases include MySQL and Postgres. On the commercial side, Microsoft SQL Server, Oracle, and Teradata are widely used. Although **they’re not really optimized for analysis**, for a number of years row-store databases were the only option for companies building data warehouses. Through careful tuning and schema design, these databases can be used for analytics. They are also attractive due to the low cost of open source options and because they’re familiar to the database administrators who maintain them. Many organizations replicate their production database in the same technology as a first step toward building out data infrastructure. For all of these reasons, data analysts and data scientists are likely to work with data in a rowstore database at some point in their career.

To reduce the width of tables, row-store databases are usually modeled in third nor‐ mal form, which is a database design approach that seeks to store each piece of infor‐ mation only once, to avoid duplication and inconsistencies. This is efficient for transaction processing but often leads to a large number of tables in the database, each with only a few columns. To analyze such data, many joins may be required, and it can be difficult for nondevelopers to understand how all of the tables relate to each other and where a particular piece of data is stored. When doing analysis, the goal is usually denormalization, or getting all the data together in one place.

**Хранит данных в столбцовом формате (Column-Store Databases) – ClickHouse, Vertica, Snowflake, …**

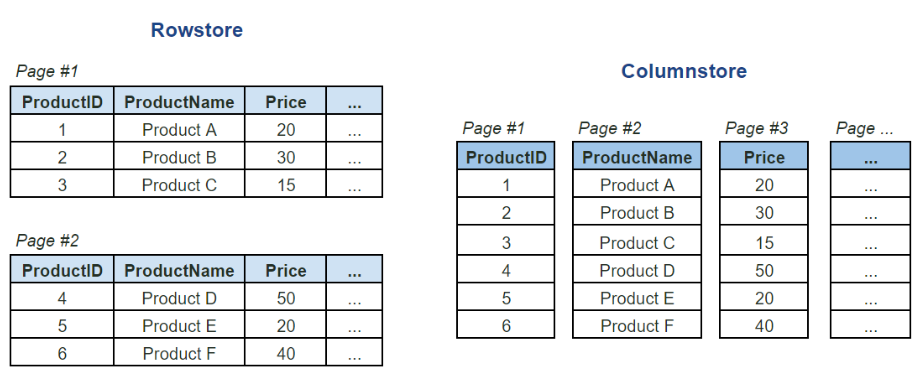
Column-store data‐ bases store the values of a column together, rather than storing the values of a row together. This **design is optimized for queries that read many records** but not neces‐ sarily all the columns.

Column-store databases are efficient at storing large volumes of data thanks to com‐ pression. Missing values and repeating values can be represented by very small marker values instead of the full value. For example, rather than storing “United Kingdom” thousands or millions of times, a column-store database will store a surro‐ gate value that takes up very little storage space, along with a lookup that stores the full “United Kingdom” value. Column-store databases also compress data by taking advantage of repetitions of values in sorted data. For example, the database can store the fact that the marker value for “United Kingdom” is repeated 100 times, and this takes up even less space than storing that marker 100 times.

Column-store databases do not enforce primary keys and do not have indexes. Repeated values are not problematic, thanks to compression. As a result, schemas can be tailored for analysis queries, with all the data together in one place as opposed to being in multiple tables that need to be joined. Duplicate data can easily sneak in without primary keys, however, so understanding the source of the data and quality checking are important.

**Updates and deletes are expensive in most column-store databases**, since data for a single row is distributed rather than stored together. For very large tables, a writeonly policy may exist, so we also need to know something about how the data is gen‐ erated in order to figure out which records to use. The data can also be slower to read, as it needs to be uncompressed before calculations are applied.

Column-store databases are generally the gold standard for fast analysis work. They use standard SQL (with some vendor-specific variations), and in many ways working with them is no different from working with a row-store database in terms of the queries you write



**Other Types of Data Infrastructure**

Databases aren’t the only way data can be stored, and there is an increasing variety of options for storing data needed for analysis and powering applications. File storage systems, sometimes called data lakes, are probably the main alternative to database warehouses. NoSQL databases and search-based data stores are alternative data stor‐ age systems that offer low latency for application development and searching log files.

**Hadoop**, also known as **HDFS (for “Hadoop distributed filesystem”),** is an open source file storage system that takes advantage of the ever-falling cost of data storage and computing power, as well as distributed systems. Files are split into blocks, and Hadoop distributes them across a filesystem that is stored on nodes, or computers, in a cluster. The code to run operations is sent to the nodes, and they process the data in parallel. Hadoop’s big breakthrough was to allow huge amounts of data to be stored cheaply. Many large internet companies, with massive amounts of often unstructured data, found this to be an advantage over the cost and storage limitations of traditional databases

**NoSQL**

NoSQL is a technology that allows for data modeling that is not strictly relational. It allows for very low latency storage and retrieval, critical in many online applications. The class includes key-value pair storage and graph databases, which store in a nodeedge format, and document stores

Examples of these data stores that you might hear about in your organization are **Cassandra**, **Couchbase**, **DynamoDB**, **Memcached**, **Giraph**, and **Neo4j**. Early on, NoSQL was marketed as making SQL obsolete, but the acronym has more recently been marketed as “not only SQL.” For analysis purposes, using data stored in a NoSQL key-value store for analysis typically requires moving it to a more traditional SQL data warehouse, since NoSQL is not optimized for querying many records at once. Graph databases have applications such as network analysis, and analysis work may be done directly in them with special query languages. The tool landscape is always evolving, however, and perhaps someday we’ll be able to analyze this data with SQL as well.

Search-based data stores include **Elasticsearch** and **Splunk**. Elasticsearch and Splunk are often used to analyze machine-generated data, such as logs. These and similar technologies have non-SQL query languages, but if you know SQL, you can often understand them