DIFFERENT TYPES OF DATABASE

STEVEN JEFF F. GUNSI
3-BSCS-A

TYPES OF DATABASES

WHAT IS DATABASES?

A database is a collection of data that is kept on a computer. Databases are used for anything from storing photos on your computer to shopping online and stock market analysis. Databases allow computers to store important data in an orderly, searchable format.

Database technology has progressed over time, and so have the various sorts of databases. There are currently a variety of database types, each with its own set of strengths and drawbacks dependent on how they are built. Understanding the various types of databases is very crucial for organizations to ensure that they have the most effective setup; nevertheless, some individuals may need to grasp this a little more.

RELATIONAL DATABASE

A relational database stores data that is arranged around other data. The link between a person purchasing online and their shopping cart is a fantastic example of a relational database. When data integrity is an issue or scalability isn't a priority, relational databases are frequently the best choice.

A relational database is a form of database that stores and allows access to data elements that are linked. The relational model, a simple and obvious means of representing data in tables, is the foundation of relational databases. Each row in a table in a relational database is a record with a unique ID called the key. The attributes of the data are stored in the table's columns, and each record usually contains a value for each attribute, making it simple to construct links between data points.

Here's an example of two tables that a small business could use to process product orders. Each record in the first table contains the customer's name, address, shipping and billing information, phone number, and other contact information. Each piece of data (attribute) has its own column, and each row has its own unique ID (key) assigned by the database. Each record in the second table—a customer order table—includes the client's ID, the product ordered, the quantity, the size and color selected, and so on—but not the customer's name or contact information.

The ID column is the only thing these two tables have in common (the key). However, the relational database may construct a relationship between the two tables because of that common column. The database can then go to the customer order table, pull the correct information about the product order, and use the customer ID from that table to look up the customer's billing and shipping information in the customer info table when the company's order processing application submits an order to the database. The warehouse can then fetch the correct product, the consumer may receive their order on schedule, and the corporation can collect payment.

The logical data structures—data tables, views, and indexes—are separated from the physical storage structures in the relational paradigm. As a result of this separation, database administrators can manage physical data storage without affecting logical data access. Renaming a database file, for example, does not rename the tables contained within it.

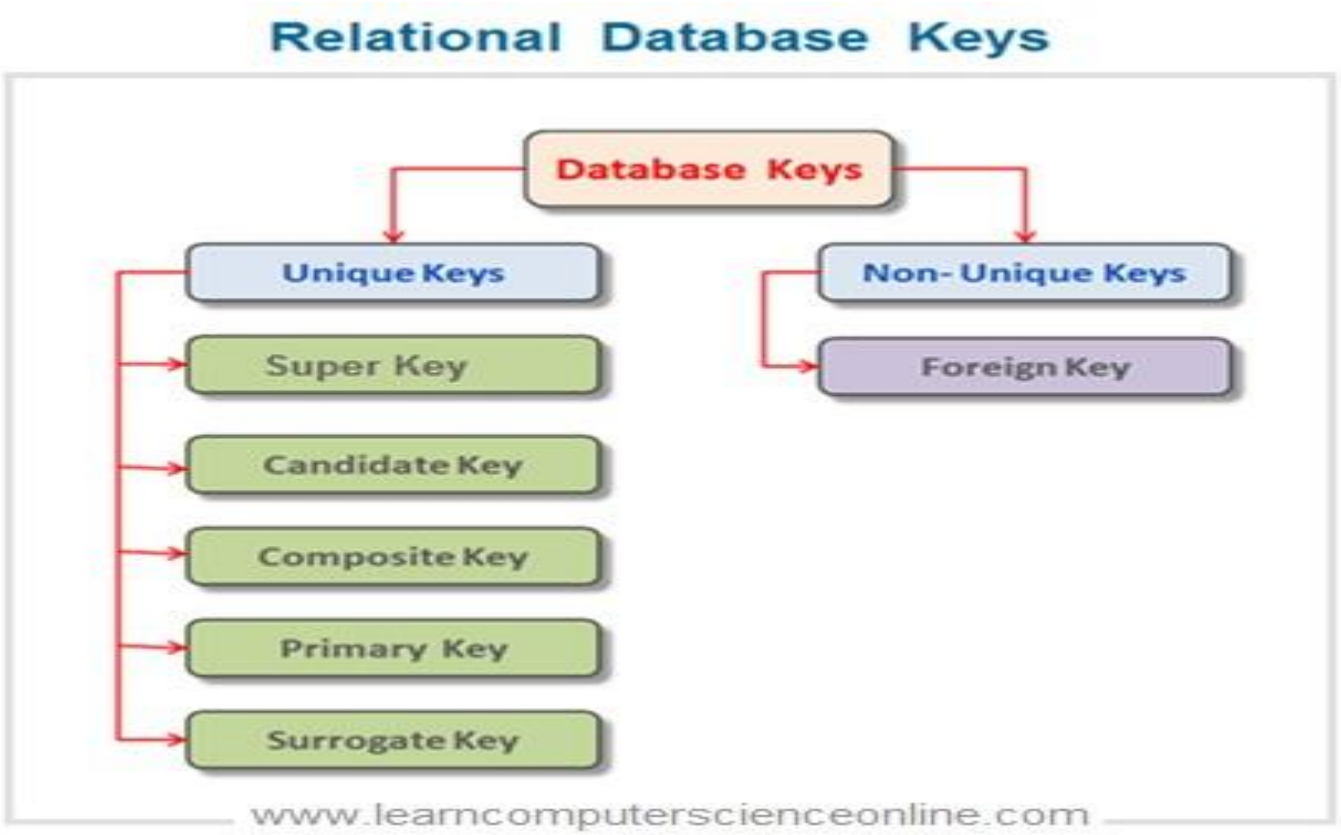
Database operations, which are clearly defined actions that enable programs to alter the data and structures of the database, are further divided into logical and physical categories. Physical operations

indicate how that data should be accessible and then carry out the task, while logical operations allow an application to specify the material it requires.

Relational databases follow particular integrity criteria to ensure that data is always valid and accessible. For example, an integrity rule can state that duplicate rows in a table are not allowed in order to prevent erroneous data from entering the database.

For a wide range of information demands, companies of many types and sizes employ the simple yet effective relational model. Relational databases are used to manage massive amounts of mission-critical customer information, track inventories, conduct ecommerce transactions, and much more. A relational database can be used for any information purpose where data points must be linked and handled in a secure, rules-based, and consistent manner.

Since the 1970s, relational databases have been used. The advantages of the relational model have made it the most frequently recognized database model today.



ANALYTICAL DATABASE (OLAP)

OLAP - Online Analytical Processing

For business intelligence (BI) analysis, an analytical database stores and handles massive data, such as business, market, and customer data. Analytical databases are designed with speed and scalability in mind.

Big data management is a specialty of analytical database software for corporate applications and services. Analytical databases are designed to respond quickly to queries and deliver advanced insights. They're also more scalable than traditional databases, and they're generally columnar databases that can efficiently write and read data to and from hard disk storage to reduce query response times. Column-based storage, in-memory loading of compressed data, and the ability to search data by numerous attributes are all features of analytical databases.

For severe analytical workloads, analytical database software is designed to quickly evaluate enormous volumes of data, performing up to 1,000 times quicker than an operational database. Business analysts, researchers, financial market analysts, big data analysts, geospatial analysts, and data scientists rely on analytical databases that can manage large amounts of data to be available.

An analytical database's historical data is compared to operational data. Historical data is information that isn't current but is only a few hours old. Analytical and operational data are compared to establish the optimal transaction and other business or research activities.

Examples of Analytical Databases

Market data — Historical price and volume data for financial markets for testing trading strategies.

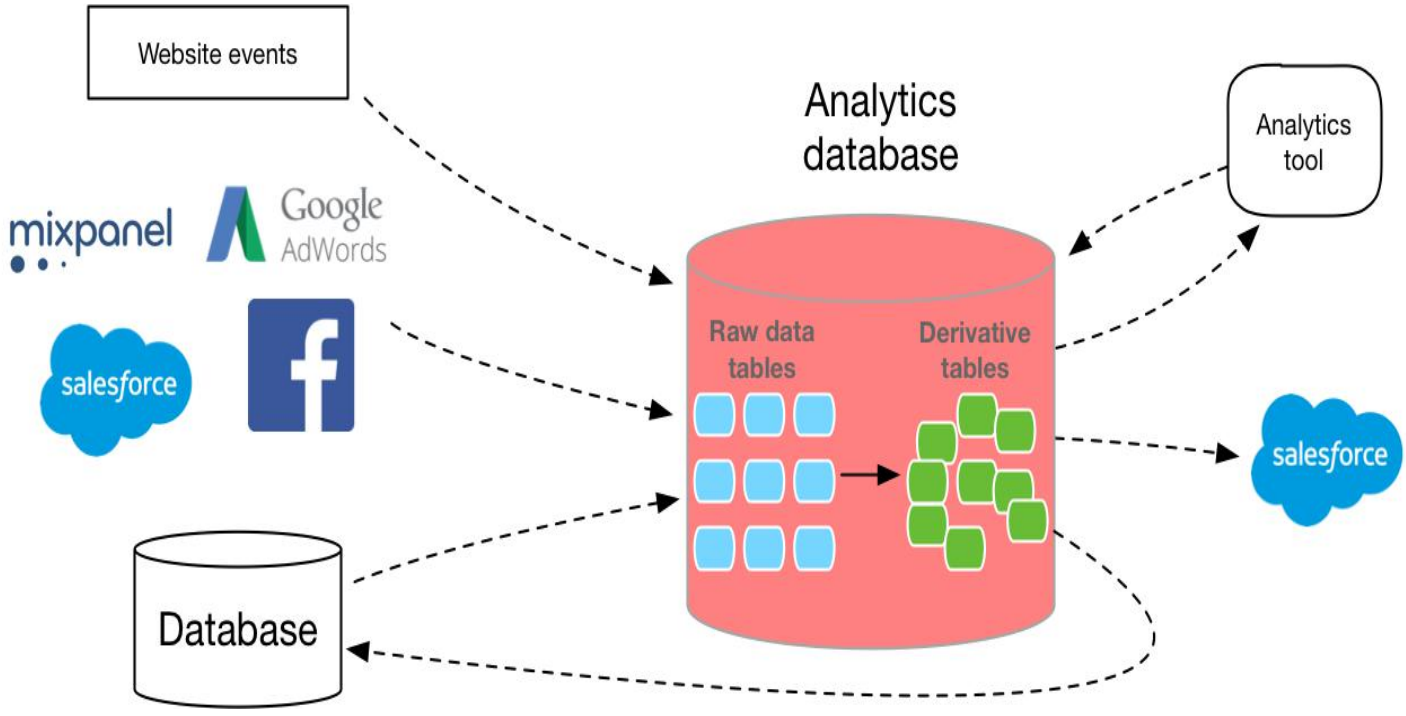
Transactional data — Historical transactions that can include purchasing patterns for improved marketing.

Sensor data — Historical data from sensors that monitor situations like the weather.

Natural language data — Study of social media posts for research purposes.

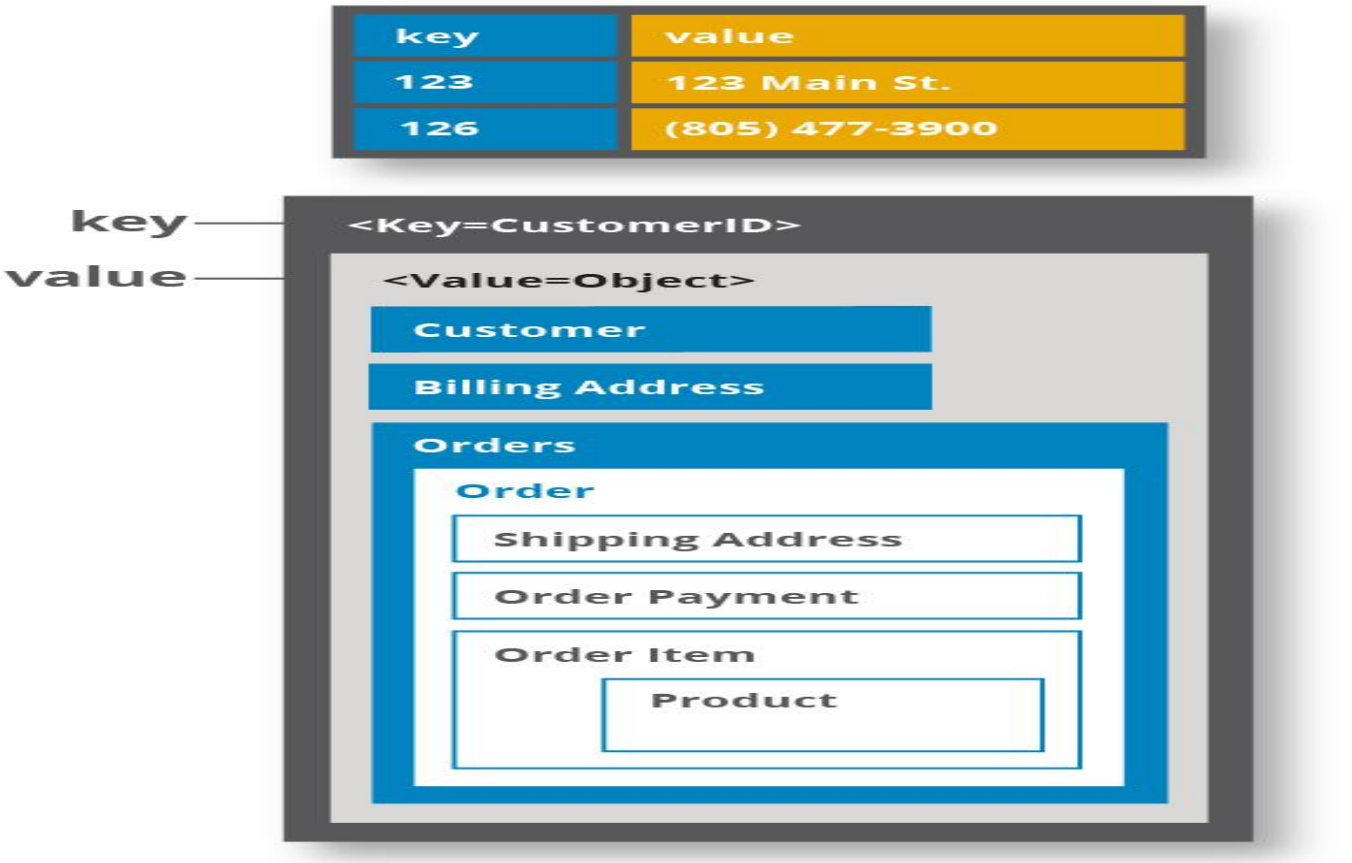
Process data — Study of processes to better understand logistics and find bottlenecks.

Machine data — Software and hardware-generated data from products to improve efficiency.



KEY-VALUE

Key–value database, sometimes known as a dictionary or hash table, is a data storage paradigm for storing, retrieving, and maintaining associative arrays, and a data structure more often known as a dictionary or hash table today. Dictionaries are made up of a collection of objects, or records, each of which has a number of fields containing data. These records are stored and retrieved using a key that is used to locate data inside the database and uniquely identifies the record.



A **key-value store**, or key-value database, is a type of data storage software program that stores data as a set of unique identifiers, each of which have an associated value. This data pairing is known as a "key-value pair." The unique identifier is the "key" for an item of data, and a value is either the data being identified or the location of that data.
The key could be anything, depending on the database software's limits, but it must be unique in the database to avoid ambiguity while looking for the key and its value. Any value, including a list or another key-value pair, might be used as the value. You can define a data type for the value in some database tools.

Data is kept in tables made up of rows and columns in a standard relational database design. Many attributes of the data to be stored in the table are specified in advance by the database developer. This opens up a lot of possibilities for optimizations like data compression and efficiency around aggregations and data access, but it also limits your options.

Instead of doing complex aggregations, key-value stores are often significantly more flexible and offer very fast read and write performance. This is because the database is looking for a single key and returning its related value rather than completing complex aggregations.

A key-value store has a number of advantages over standard row-column-based databases. A key-value store can be particularly quick for read and write operations because to the basic data format that lends it its name. And, as we generate more data without traditional structures, key-value stores are a valuable asset in modern programming.

Furthermore, because key-value stores do not require placeholders for optional data such as "null," they may have lower storage requirements and grow virtually linearly with the number of nodes.

COLUMN-FAMILY

A column family is a database entity that has linked data columns in its columns. It's a tuple (pair) made up of a key–value pair, with the key mapped to a collection of columns as the value. A column family acts as a "table," with each key-value pair acting as a "row," similar to relational databases. A tuple (triplet) consists of a column name, a value, and a timestamp for each column. This data would be placed together in a table with other unrelated data in a relational database table.

There are two sorts of column families:

1. **The standard column family** is made up of only columns.

A NoSQL object that comprises columns of linked data is known as the standard column family. It's a tuple (pair) made up of a key–value pair, with the key mapped to a collection of columns as the value. A standard column family can be thought of as a "table," with each key–value pair being a "row."cEach column is a tuple (triplet), consisting of a column name, a value, and a timestamp.

2. **Super column family** is contained a map of super column family.

A NoSQL object that contains column families is known as a super column family. A super column family is similar to a "view" on a number of tables in relational databases. It is a tuple (pair) that consists of a key–value combination, where the key is mapped to a value that are column families. It can also be viewed as a table map.

GRAPH

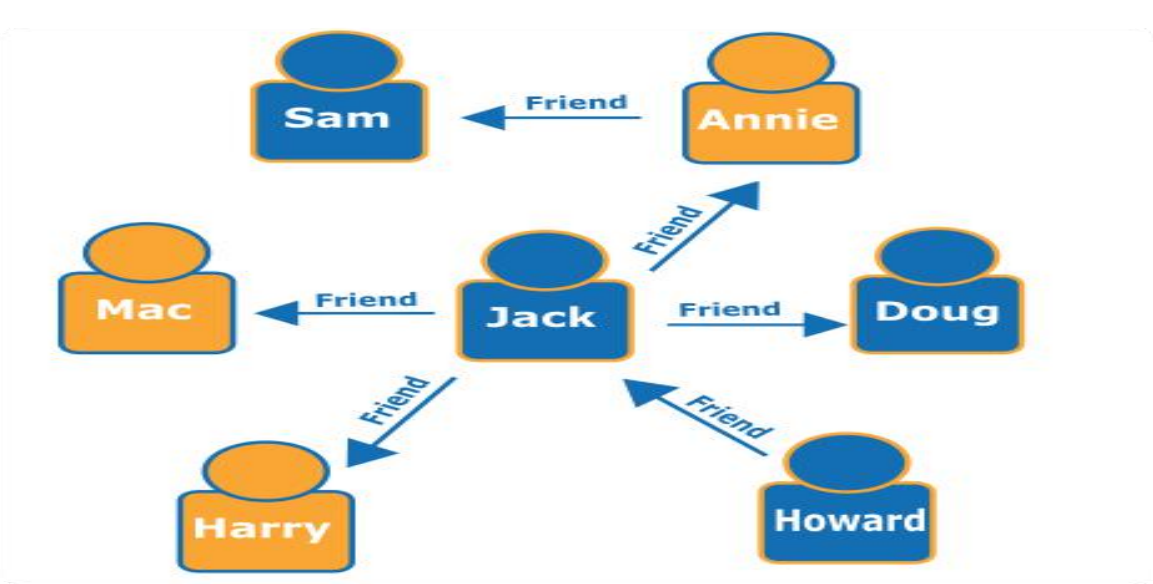
Graph databases are databases that place equal emphasis on data and connections between it. The data in this database isn't restricted to established models. When you do a search in most other databases, you can find links between data. With a graph database, these relationships are preserved alongside the original data in the database. When your primary goal is to manage the links between your data, this results in a more efficient and speedier database.

Graph databases are designed specifically for storing and navigating relationships. Relationships are first-class citizens in graph databases, and they account for the majority of the database's value. Nodes are used to store data entities, and edges are used to store relationships between things in graph databases. An edge has a start node, an end node, a type, and a direction, and it can be used to define parent-child connections, actions, and ownership, among other things. The amount and types of relationships that a node can have are limitless.

A graph can be navigated along specified edge types or over the entire graph in a graph database. Because the relationships between nodes are not calculated at query time but are persisted in the database, traversing the joins or relationships in graph databases is highly fast. When you need to construct linkages between data and query these associations fast, graph databases are useful for use cases like social networking, recommendation engines, and fraud detection.

A social network graph is depicted in the graph below. You may find out who a person's "friends of friends" are by looking at the people (nodes) and their relationships (edges)—for example, Howard's pals.

Sophisticated fraud protection is possible with graph databases. You can leverage relationships in graph databases to conduct financial and buy transactions in near-real time. You can discover that, for example, a potential purchaser is using the same email address and payment card as a known fraud instance using quick graph queries. Numerous people linked with a personal email account, or multiple people sharing the same IP address but live in various physical addresses, can all be easily detected using graph databases.



DOCUMENT

A document database is a nonrelational database that stores and queries data as JSON-like documents. By employing the same document-model format as their application code, document databases make it easier for developers to store and query data in a database. Documents and document databases can adapt to the needs of applications due to their flexible, semistructured, and hierarchical nature. The document model is particularly suited to use cases where each document is unique and evolves over time, such as catalogs, user profiles, and content management systems.

Flexible indexing, strong ad hoc searches, and analytics over collections of documents are all possible with document databases.

In the following example, a JSON-like document describes a book.

```
[
 {
          "year" : 2013,
          "title" : "Turn It Down, Or Else!",
          "info" : {
          "directors" : [ "Alice Smith", "Bob Jones"],
          "release_date" : "2013-01-18T00:00:00Z",
          "rating" : 6.2,
          "genres" \: ["Comedy", "Drama"],
          "image_url": : "http://ia.media-
imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@@._V1_SX400_.jpg",
"plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
"actors" : ["David Matthewman", "Jonathan G. Neff"]
 }
 },
 {
 "year": 2015,
"title": "The Big New Movie",
 "info": { "plot": "Nothing happens at all.",
 "rating": 0
 }
 }
]
```

For content management systems such as blogs and video platforms, a document database is an excellent solution. Each entity that the program tracks can be kept as a separate document in a document database. For a developer, updating an application when requirements change is easier with a document database. Furthermore, if the data model needs to be updated, just the documents that are affected must be updated. There is no need to alter the schema, and no database downtime is required to implement the modifications.

For storing catalog information, document databases are efficient and useful. Varied products in an e-commerce platform, for example, typically have different numbers of attributes. In relational databases, managing hundreds of characteristics is wasteful, and reading performance suffers as a result. Each product's properties can be documented in a single document using a document database, making it easier to maintain and understand. Changing the characteristics of one product has no bearing on the others.

REFERENCES

- https://www.google.com/search?q=what+is+relational+as+a+type+of+databases&hl=en&sxsrf=AOaemvJLfT92Z3a83bYTMR4ZAl_aWjliRw:1637592722178&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjHo_ejnKz0AhU_slYBHS_BCFwQ_AUoAXoECAEQAw&biw=1366&bih=657&dpr=1#imgrc=1bzfOkwpPs15YM
- https://www.oracle.com/ph/database/what-is-a-relational-database/#:~:text=A%20relational%20database%20is%20a,of%20representing%20data%20in%20tables.
- https://www.omnisci.com/technical-glossary/analytical-database
- https://en.wikipedia.org/wiki/Key%E2%80%93value_database
- https://hazelcast.com/glossary/key-value-store/
- https://en.wikipedia.org/wiki/Column_family#:~:text=A%20column%20family%20is%20a,pair%20being%20a%20%22row%22.
- https://aws.amazon.com/nosql/graph/
- https://aws.amazon.com/nosql/document/#:~:text=The%20document%20database%20defined,use%20in%20their%20application%20code.