Session 7

Non-relational DBMS

Caching

JDBC

sourcemind

What Is a Non-Relational Database?

A **non-relational database** is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored.

For example, data may be stored as simple **key/value pairs**, as **JSON documents**, or as a **graph consisting of edges and vertices**.

sourcemind

The benefits of a non-relational database

There are several advantages to using non-relational databases, including:

- Massive dataset organization

In the age of Big Data, non-relational databases can not only store massive quantities of information, but they can also query these datasets with ease. Scale and speed are crucial advantages of non-relational databases.

sourcemind

The benefits of a non-relational database

- Flexible database expansion

Data is not static. As more information is collected, a non-relational database can absorb these new data points, enriching the existing database with new levels of granular value even if they don’t fit the data types of previously existing information.

The benefits of a non-relational database

- Multiple data structures

The data now collected from users takes on myriad forms, from numbers and strings, to photo and video content, to message histories. A database needs the ability to store these various information formats, understand relationships between them, and perform detailed queries. No matter what format your information is in, non-relational databases can collate different information types together in the same document.

The benefits of a non-relational database

- Built for the cloud

A non-relational database can be massive. And as they can, in some cases, grow exponentially, they need a hosting environment that can grow and expand with them. The cloud’s inherent scalability makes it an ideal home for non-relational databases.

sourcemind

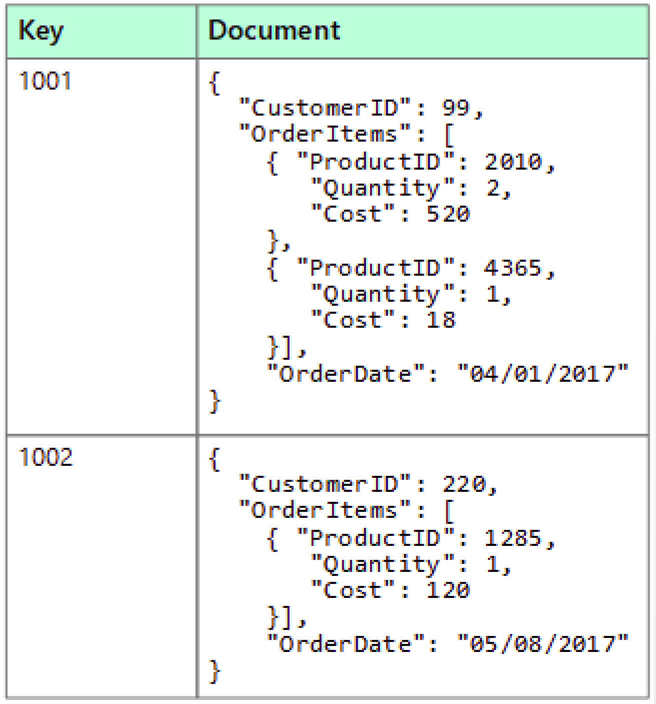
Major categories of NoSQL database

- Document data stores
- Columnar data stores
- Key/value data stores
- Graph data stores
- Time series data stores
- Object data stores
- External index data stores

sourcemind

Document data stores

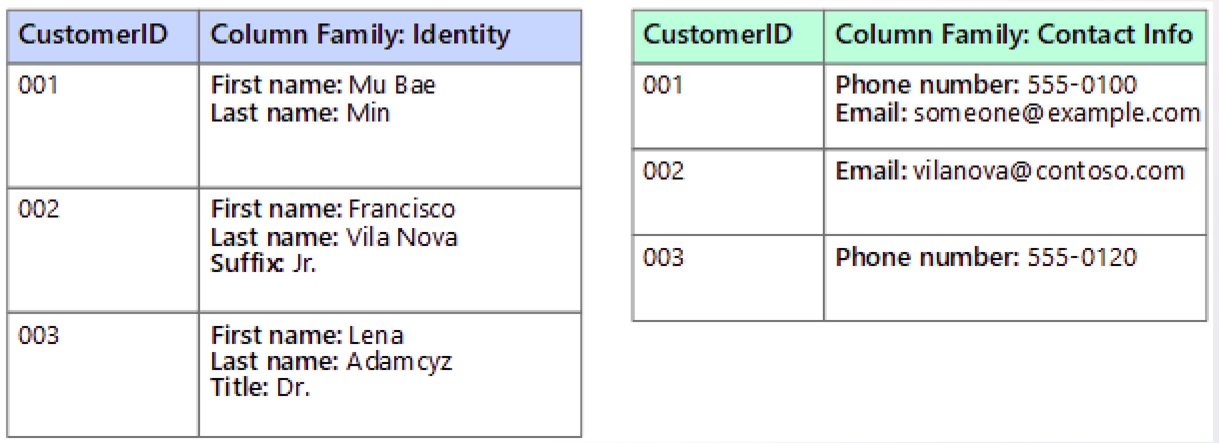
A document data store manages a set of named string fields and object data values in an entity that's referred to as a document.



Key	Document
1001	`{ "CustomerID": 99, "OrderItems": [ { "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }`
1002	`{ "CustomerID": 220, "OrderItems": [ { "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }`

Columnar data stores

A columnar or column-family data store organizes data into columns and rows. In its simplest form, a column-family data store can appear very similar to a relational database, at least conceptually.



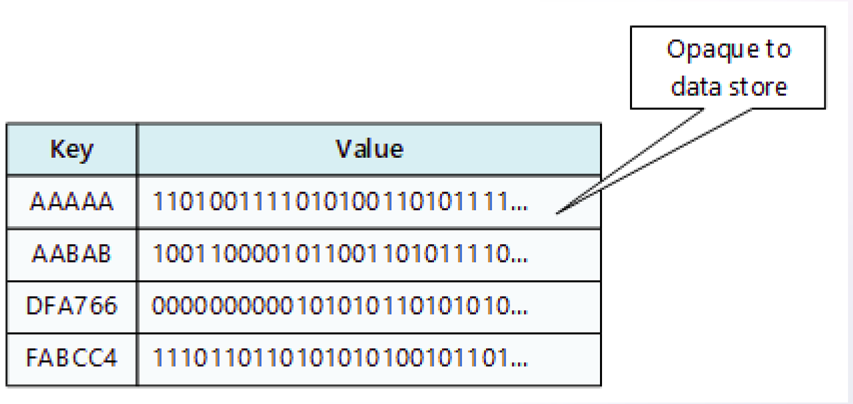
CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

sourcemind

Key/value data stores

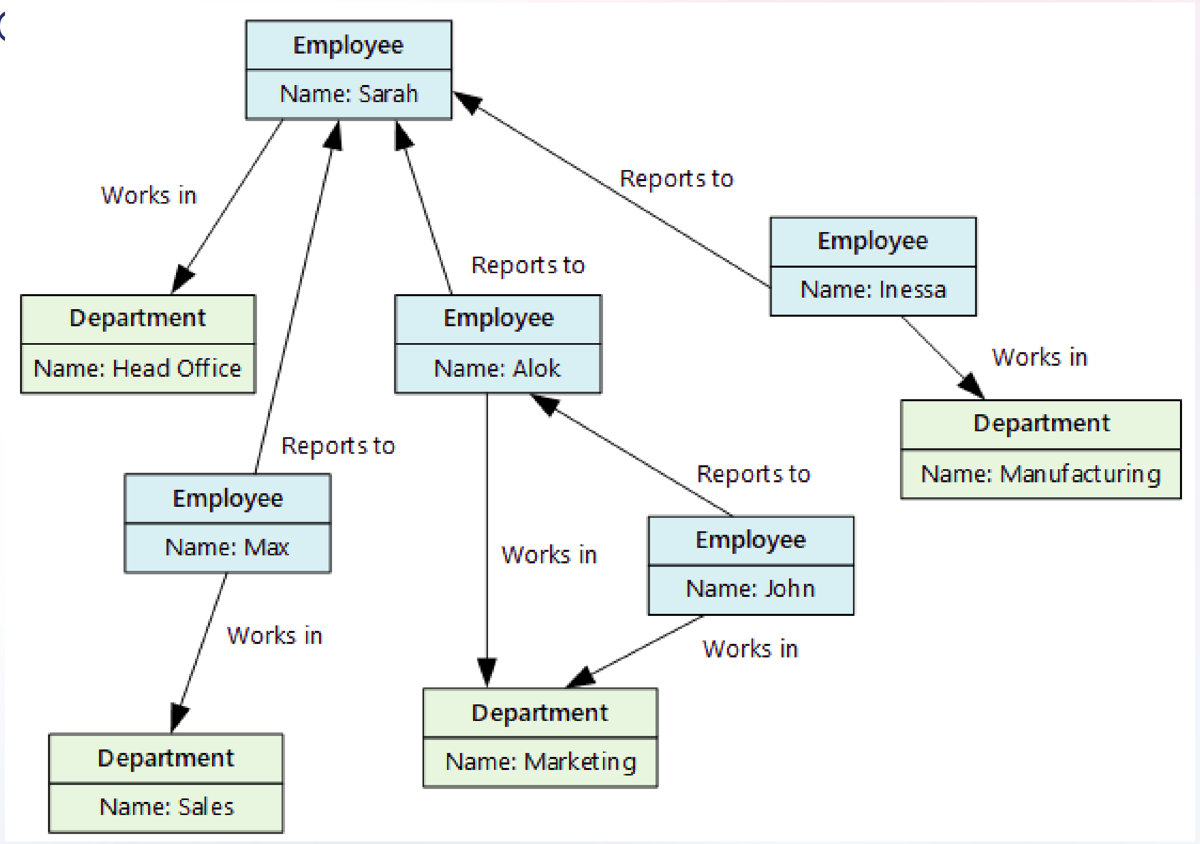
A key/value store is essentially a large hash table. You associate each data value with a unique key, and the key/value store uses this key to store the data by using an appropriate hashing function.


Opaque to data store

Key	Value
AAAAA	1101001111010100110101111...
AABAB	1001100001011001101011110...
DFA766	0000000000101010110101010...
FABCC4	1110110110101010100101101...

Graph data stores

A graph data store manages two types of information, nodes and edges. Nodes represent entities, and edges specify the relationships between these entities. Both nodes and edges can have properties that provide information about that node or edge, similar to columns in a table. Edges can also have a direction indicating the nature of the relationshi



sourcemind

Cassandra

Open Source NoSQL Database

- ➢ https://cassandra.apache.org/_/index.html

Get started with Cassandra

- ➢ https://cassandra.apache.org/_/quickstart.html

Tarball or package download

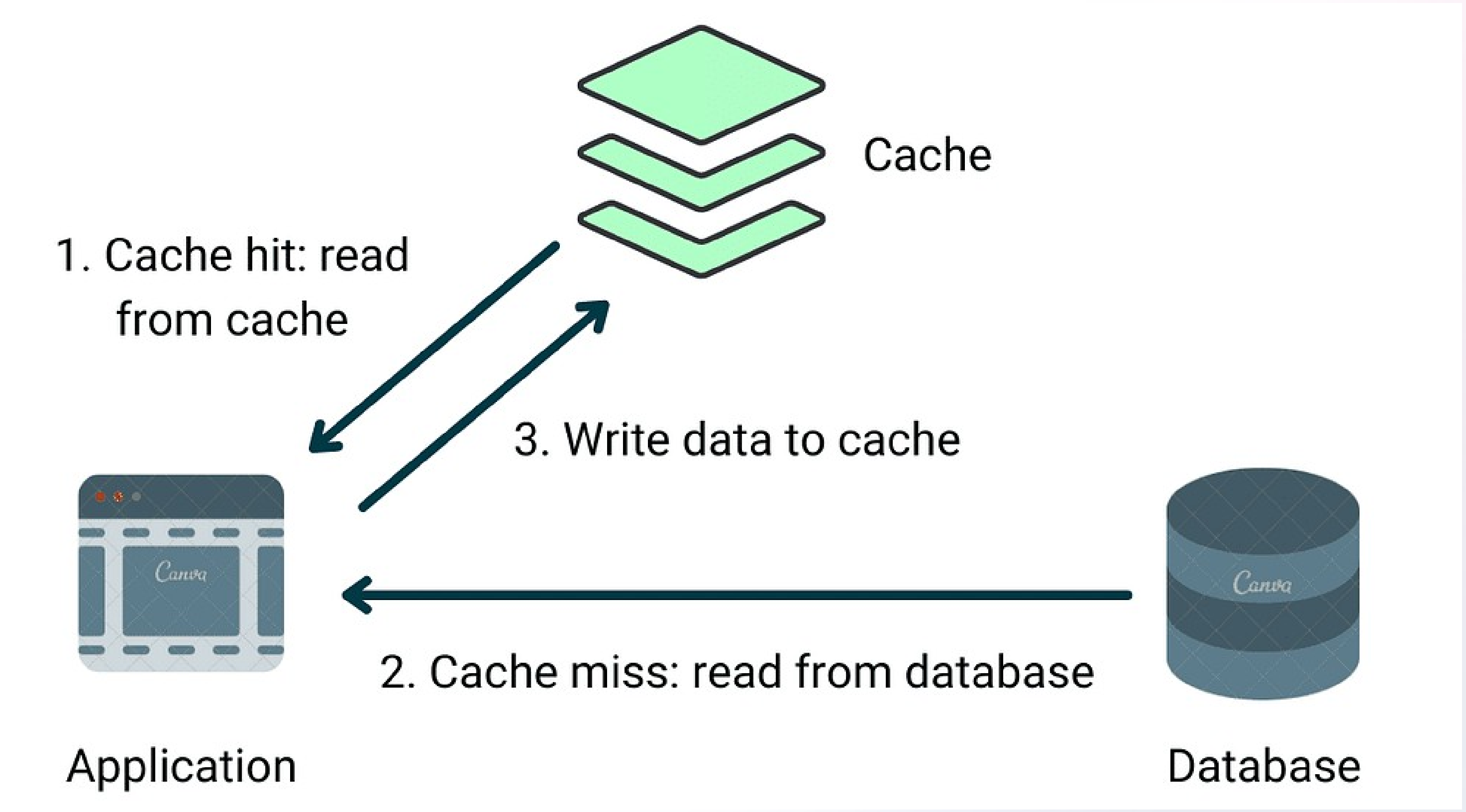
- ➢ https://cassandra.apache.org/_/download.html

Caching

Caching is a process of storing data in a temporary storage location called **cache** to improve the speed of data access.

A cache is a high-speed storage that stores a small proportion of critical data so that future requests for that data can be served faster.

Caching


Cache
1. Cache hit: read from cache
3. Write data to cache
Canva
Canva
2. Cache miss: read from database
Application
Database

sourcemind

Caching Solutions

Redis: (Remote Dictionary Server) is an in-memory data structure store, used as a distributed, in-memory key-value database, cache and message broker.

Memcached: it is a general purpose distributed memory caching system.

Aerospike: is a flash memory and in-memory open-source distributed key-value NoSQL database management system.

JDBC

sourcemind

JDBC

Java Database Connectivity (JDBC) is an API for Java that enables Java programs to interact with relational databases like MySQL, PostgreSQL, Oracle, etc. It provides a standard interface for Java programs to communicate with databases and execute SQL queries.

JDBC Architecture

The JDBC architecture consists of two main layers:

- JDBC API
- JDBC Driver

JDBC API

The JDBC API provides a set of classes and interfaces that allow Java programs to interact with databases. Some of the important interfaces in JDBC API are:

- **Connection**: it represents a connection to a database.
- **Statement**: it is used to execute SQL queries.
- **PreparedStatement**: it is used to execute parameterized SQL queries.
- **ResultSet**: it represents the result set of a SQL query.

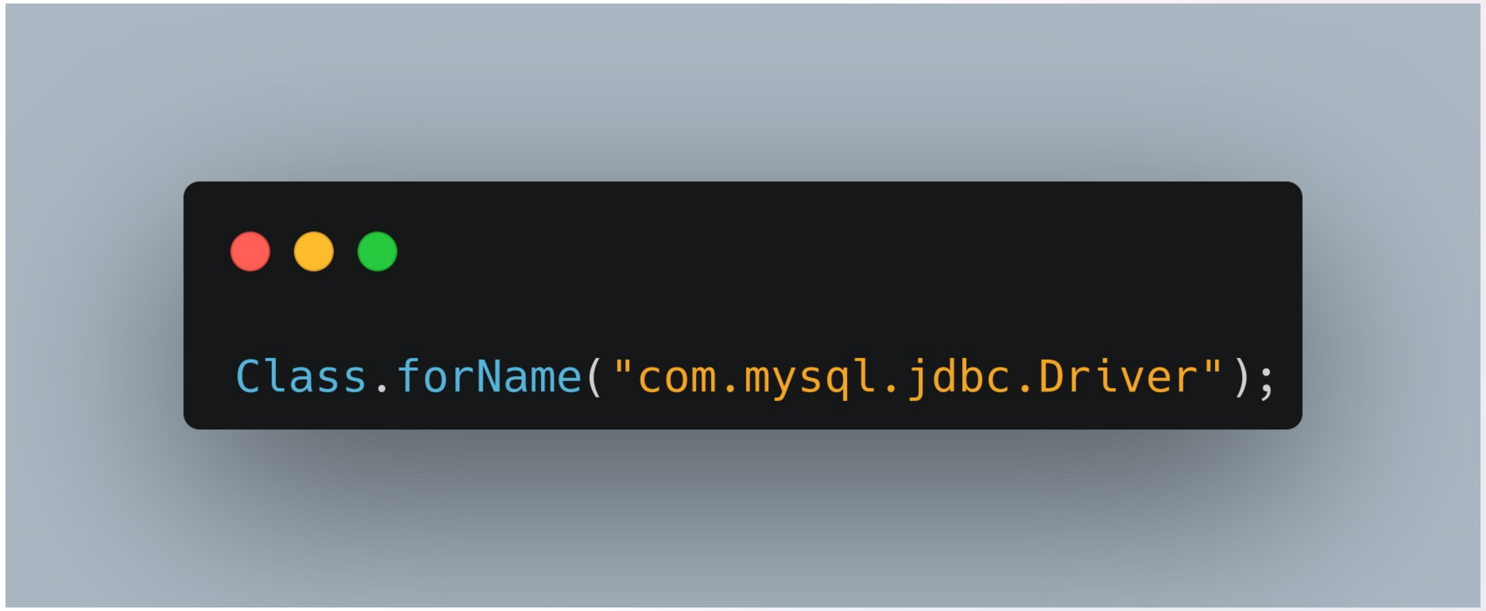
JDBC Driver

A JDBC driver is a software componentthat allows Java programs to connect to a database. There are four types of JDBC drivers:

https://jdbc.postgresql.org/download/

Example 1 : connection to a MySQL database

First, we need to load the MySQL JDBC driver:

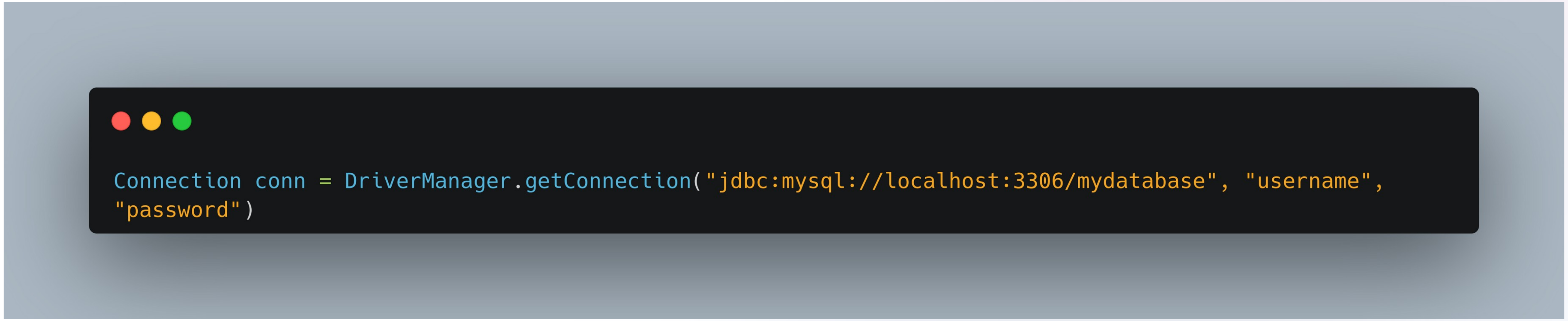


```
Class.forName("com.mysql.jdbc.Driver");
```

sourcemind

Example 1 : connection to a MySQL database

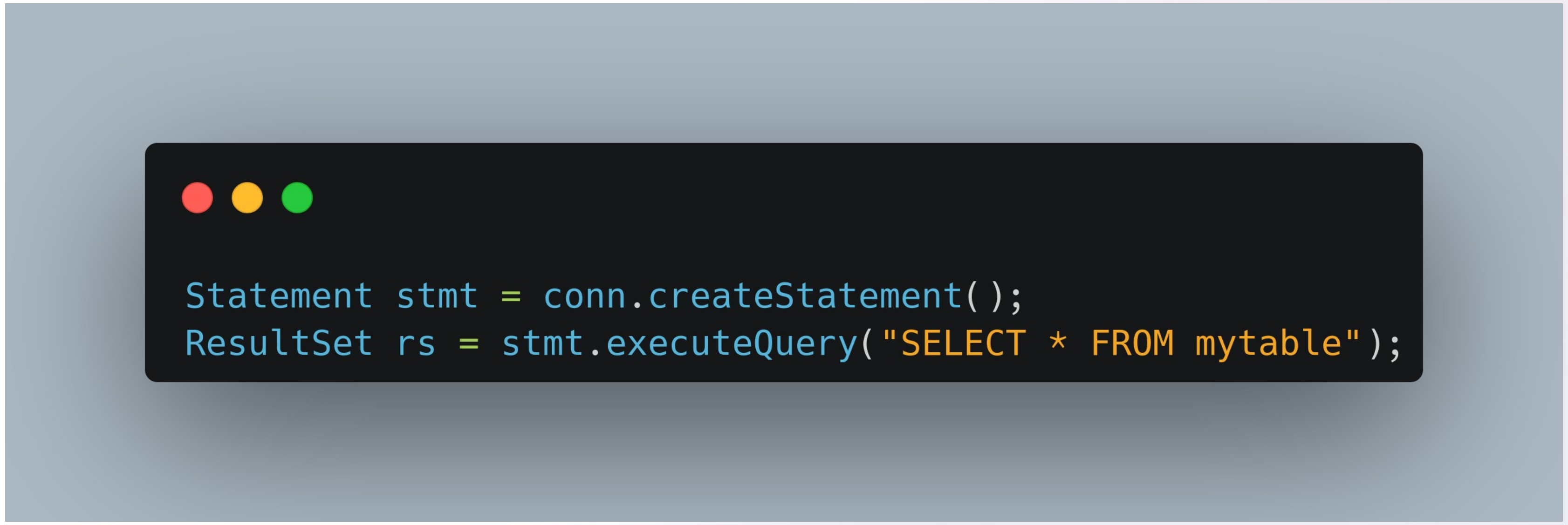
Next, we need to establish a connection to the database:



```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase", "username",
"password")
```

Example 1 : connection to a MySQL database

Once we have a connection, we can execute SQL querie using Statement or PreparedStatement:

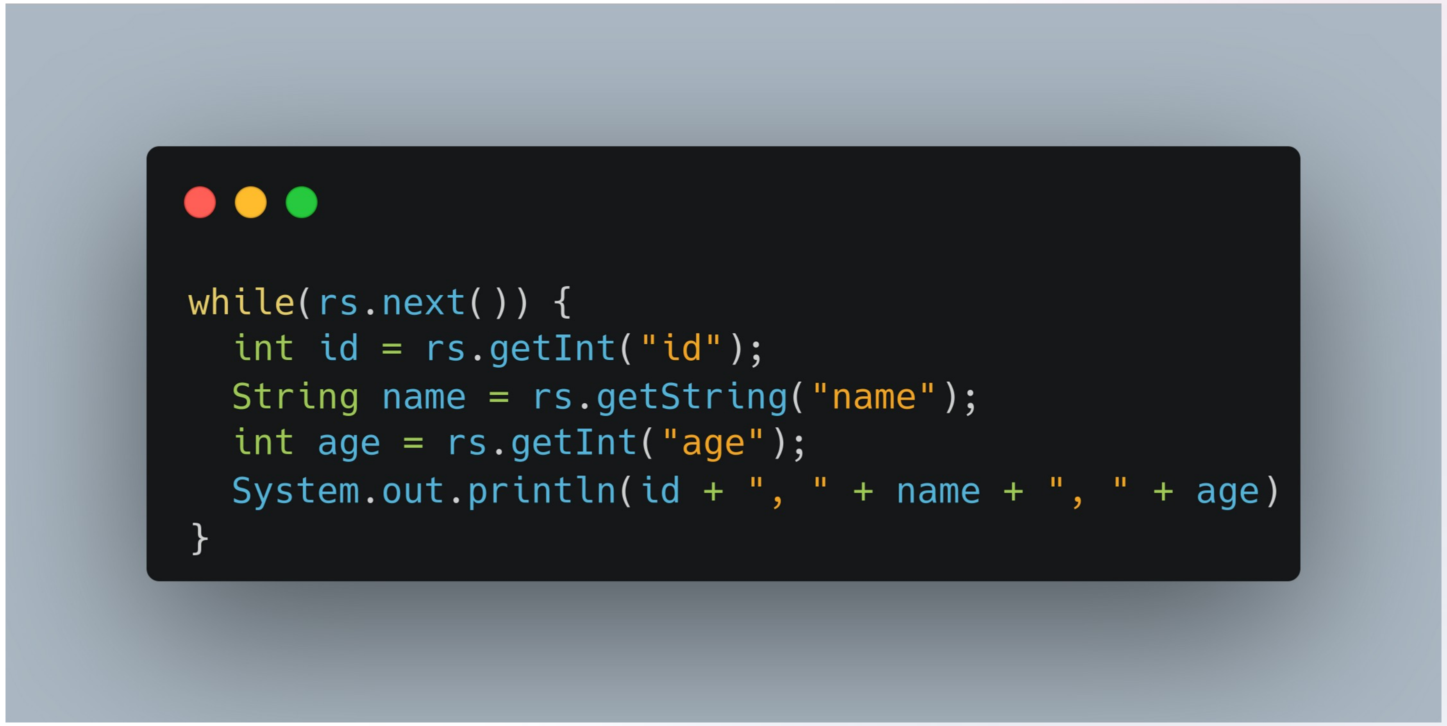


```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM mytable");
```

sourcemind

Example 1 : connection to a MySQL database

Finally, we can process the result set:



```
while(rs.next()) {
  int id = rs.getInt("id");
  String name = rs.getString("name");
  int age = rs.getInt("age");
  System.out.println(id + ", " + name + ", " + age)
}
```

sourcemind

Example 1 : connection to a MySQL database

PreparedStatement:

```
PreparedStatement p = c.prepareStatement("select i from t where i = ?");
p.setInt(1,99);
rs=p.executeQuery();
```

Example 2 : connection to a PostgreSQL database

https://jdbc.postgresql.org/documentation/use/

End

sourcemind