Sqoop?

- Sqoop (SQL-to-Hadoop) is an open-source tool for efficiently transferring (import and export of structured data) mass data between:

  relational databases (RDBMSs like MySQL, PostgreSQL, Oracle, SQL Server, etc.) and

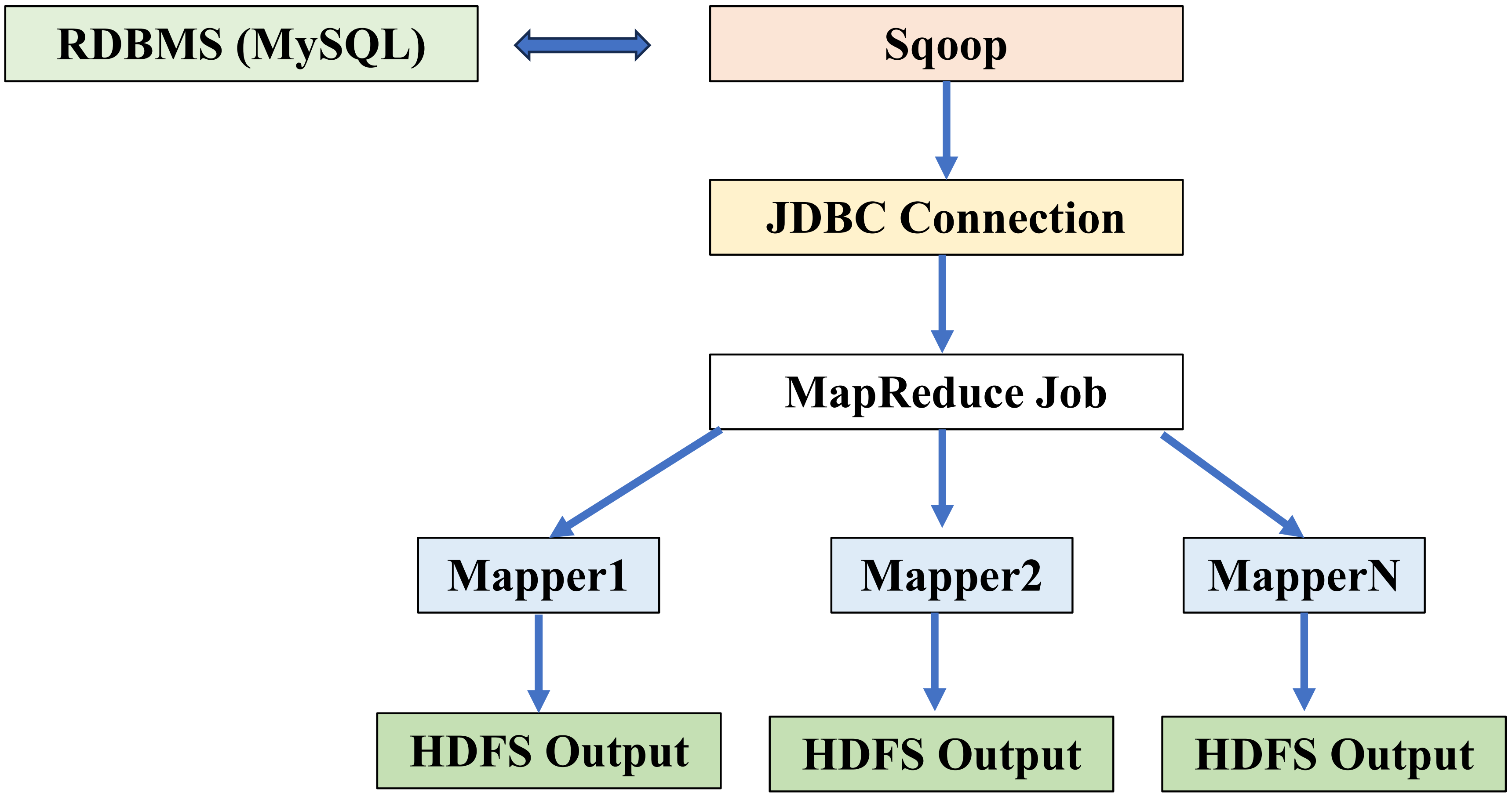
  Hadoop (HDFS) ecosystem.

- Useful when:
  - Import data from relational databases into Hadoop for processing with MapReduce, Hive, or Pig.
  - Export processed data back from Hadoop to the relational database.
  - Direct loading into Hive or HBase.

Relational Database: stores data in tables (like spreadsheets), and tables are related to each other through keys (like a *primary key* or a *foreign key*). RDBMS (Relational Database Management System) software helps to create, read, update, and delete (*CRUD*) the data stored in relational databases.

Sqoop Architecture

A basic Sqoop architecture looks like this:



JDBC: Java Database Connectivity

Sqoop Working

working mechanism of Sqoop when **importing data from a relational database to Hadoop**:

1. User Command:

sqoop import --connect jdbc:mysql://localhost/employees --username user --password pass --table dept

2. JDBC Connection:

Sqoop uses JDBC to connect to the specified RDBMS and fetch metadata about the table (columns, data types, etc.).

3. Splitter Logic:

Sqoop splits the table data into ranges for parallel import. It selects a numeric column (e.g., id) and decides how to divide the dataset.

4. MapReduce Job Creation:

Sqoop generates a MapReduce job to import the data. Each Mapper fetches a slice of the data (based on the range split).

No reducer is used because data transformation is not required during import/export.

Sqoop Working

5. Parallel Import into HDFS:

Each mapper task pulls a chunk of data from the database in parallel and writes it to HDFS.

6. Output Format: Data can be stored in:

- Text format (default)
- SequenceFile format
- Avro, Parquet (with extensions)

- You can directly load data into Hive using --hive-import.
- You can export data from HDFS back to the database using:

```
sqoop export --connect jdbc:mysql://localhost/employees --usernam
user --password pass --table dept --export-dir /user/hadoop/dept
```

JDBC - Java Database Connectivity

- It is an API (Application Programming Interface) that allows Java programs to connect to and interact with relational databases (like MySQL, PostgreSQL, Oracle, etc.).
- It like a bridge: Java ↔ JDBC Driver ↔ Database
- When a program needs to read from or write to a database, it uses JDBC to open a connection, send SQL queries, and fetch results.
- Sqoop is a tool to transfer data between relational databases and Hadoop (HDFS, Hive, HBase, etc.).
- When Sqoop required to connect to an external database (like MySQL or Oracle), it uses a JDBC connection under the hood.
- In other word: Sqoop uses JDBC drivers to connect to databases, read/write data, and **import/export** it into Hadoop.

How JDBC Connection work in Sqoop?

1. *--connect* specifies a JDBC URL that tells Sqoop which database and where.
2. Sqoop loads the **JDBC driver** for that database (like the MySQL JDBC Driver).
3. Sqoop opens a **JDBC connection** to the database.
4. Sqoop runs SQL queries to fetch the data.
5. Sqoop stores the fetched data into Hadoop storage.

Term	Meaning
JDBC	Java API for database access
JDBC Driver	Software library for a specific database
JDBC Connection in Sqoop	How Sqoop talks to relational databases to import/export data

Sqoop Import Command

sqoop import \ **# Start a Sqoop import job**

*--connect jdbc:mysql://localhost:3306/employees_db * **# JDBC URL: database type (mysql), server (localhost), port (3306), database name (employees_db)**

*--username root * **# Database username for authentication**

*--password yourpassword * **# Password for authentication (not recommended in plain text — better to use --password-file)**

*--table employees * **# The table in the database to import**

--target-dir /user/hadoop/employees **# HDFS directory where imported data will be stored**

For different databases, JDBC URLs are slightly different:

Database	Example JDBC URL
MySQL	jdbc:mysql://localhost:3306/dbname
PostgreSQL	jdbc:postgresql://localhost:5432/dbname
Oracle	jdbc:oracle:thin:@localhost:1521:orcl

where

- **jdbc**: = protocol
- **mysql/postgresql/oracle**: = database type
- **localhost:3306**: = server and port
- **dbname**: = name of the database

```
jdbc:mysql://localhost:3306/employees_db
|___| |_____| |_________| |__________|
  |      |        |            |
protocol db_type server:port  database_name
```

Meaning:

- **jdbc: tells Java we are using JDBC.**
- **mysql: which RDBMS we are connecting to (could be mysql, postgresql, oracle, etc.).**
- **localhost:3306: server (localhost) and port (3306 is default for MySQL).**
- **employees_db: specific database name inside MySQL server.**