Globant
we are ready

We create **innovative software products** that appeal to **global audiences**

Big Data Training

Sqoop

DECEMBER 2014

Speakers

Leandro Mora

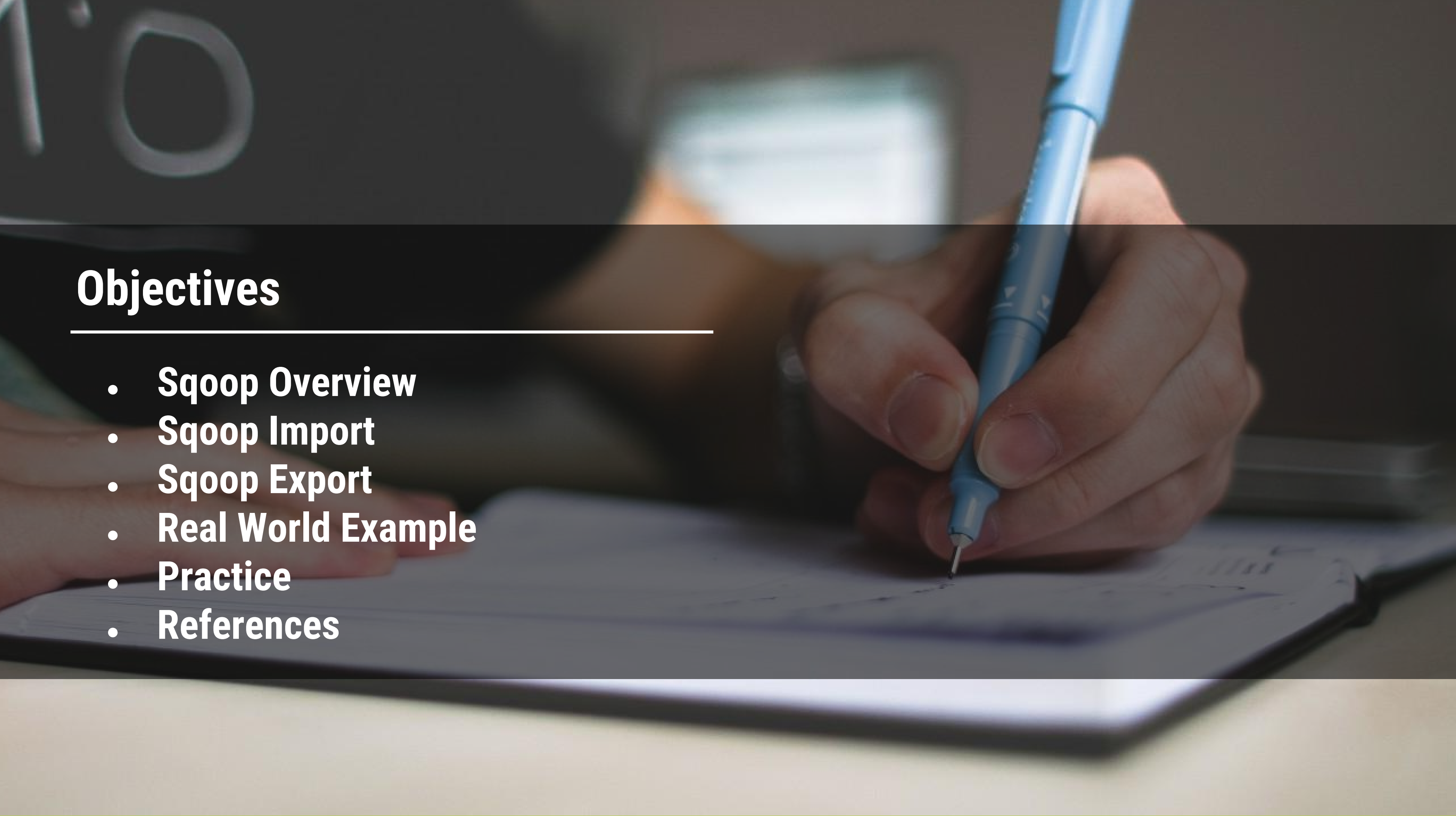
Big Data Specialist

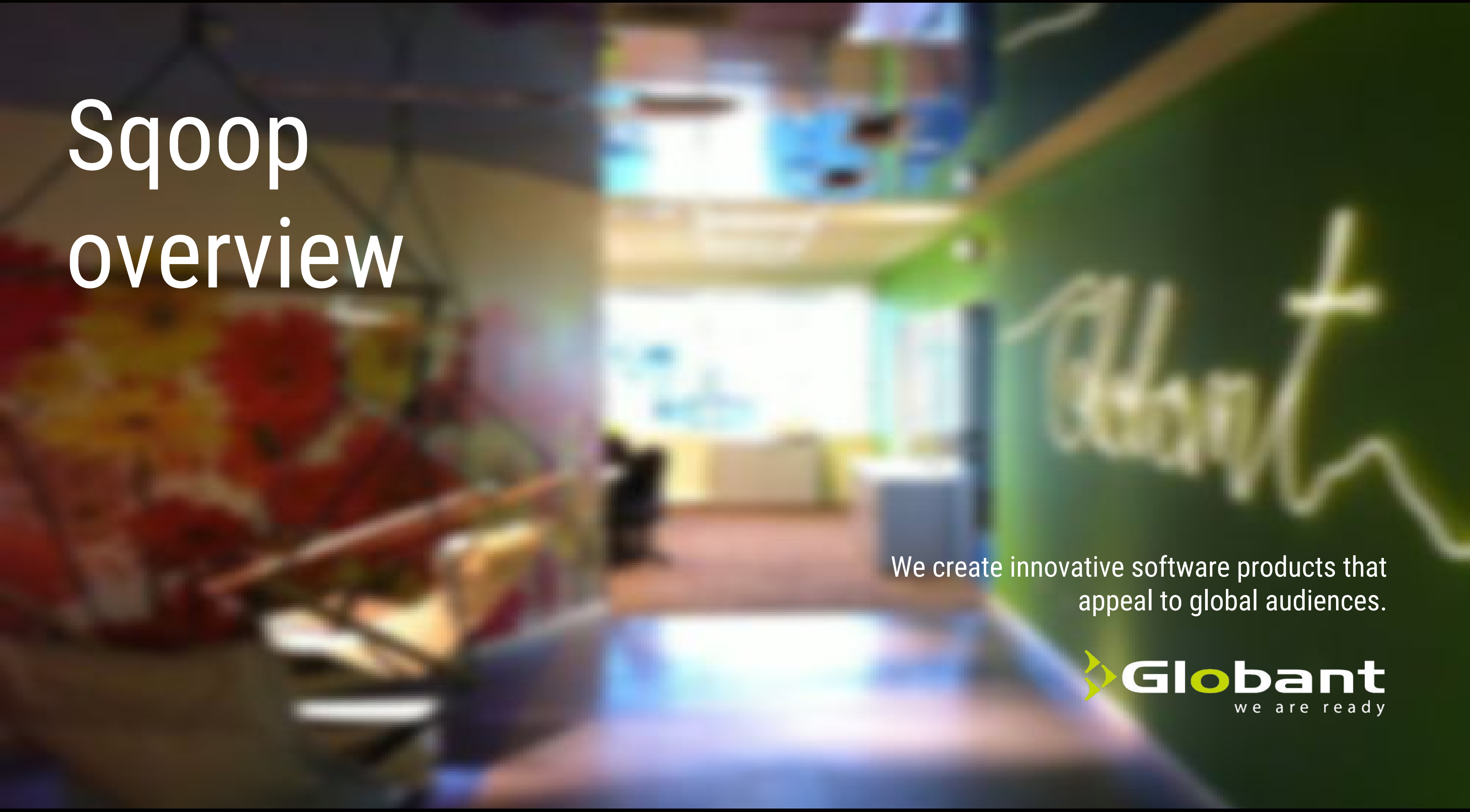
Gonzalo Zarza

Big Data Specialist

Objectives

- Sqoop Overview
- Sqoop Import
- Sqoop Export
- Real World Example
- Practice
- References





Sqoop overview

We create innovative software products that appeal to global audiences.

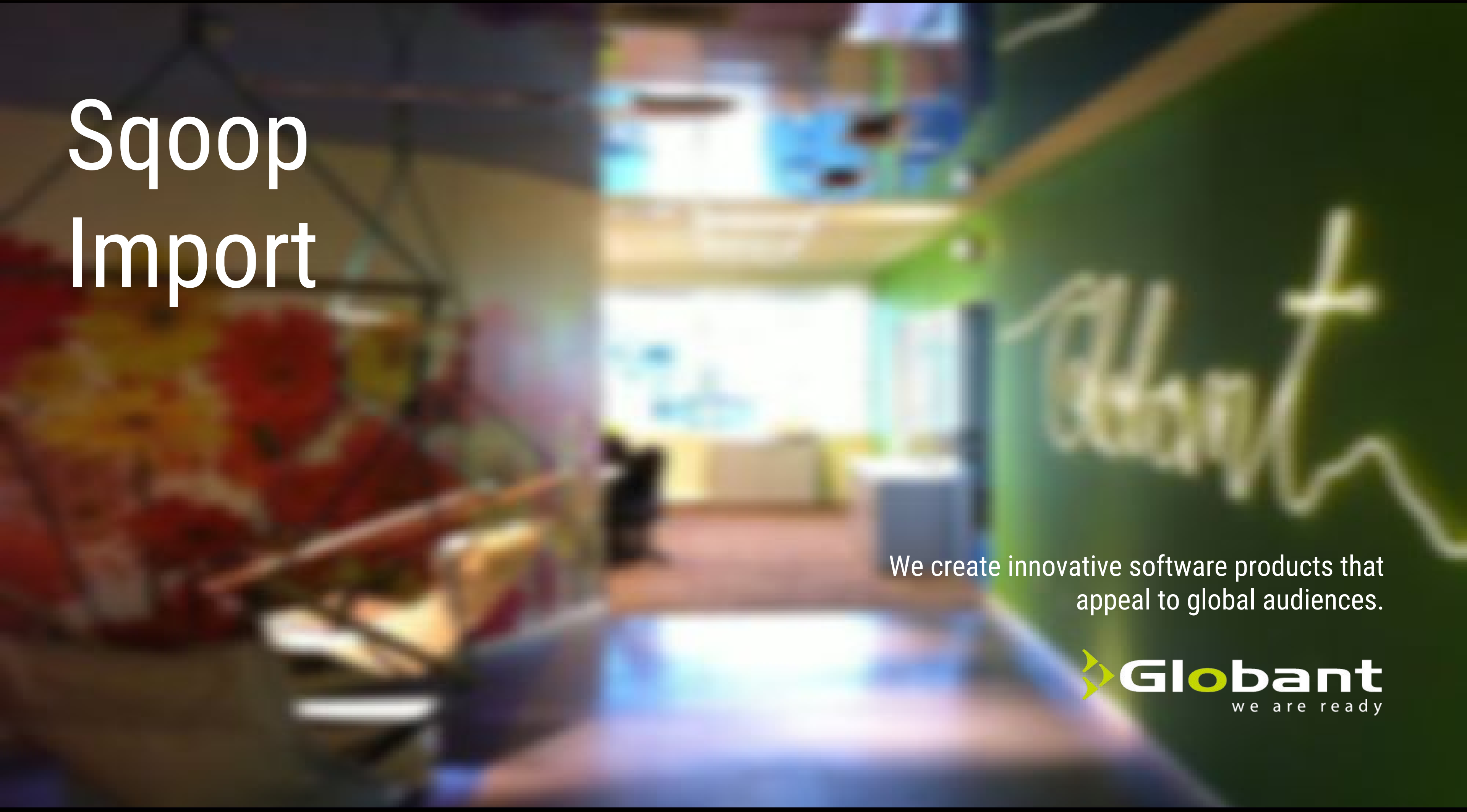
Globant
we are ready

What is Sqoop?

Sqoop is a tool designed to **transfer data** between Hadoop and relational databases.

- **Import** data **from** a relational database management system (RDBMS) **into** the Hadoop Distributed File System (**HDFS**).
- **Transform** the data in Hadoop MapReduce.
- **Export** data **into** an **RDBMS**.

Sqoop **uses MapReduce** to import and export the data, which provides parallel operation as well as fault tolerance.



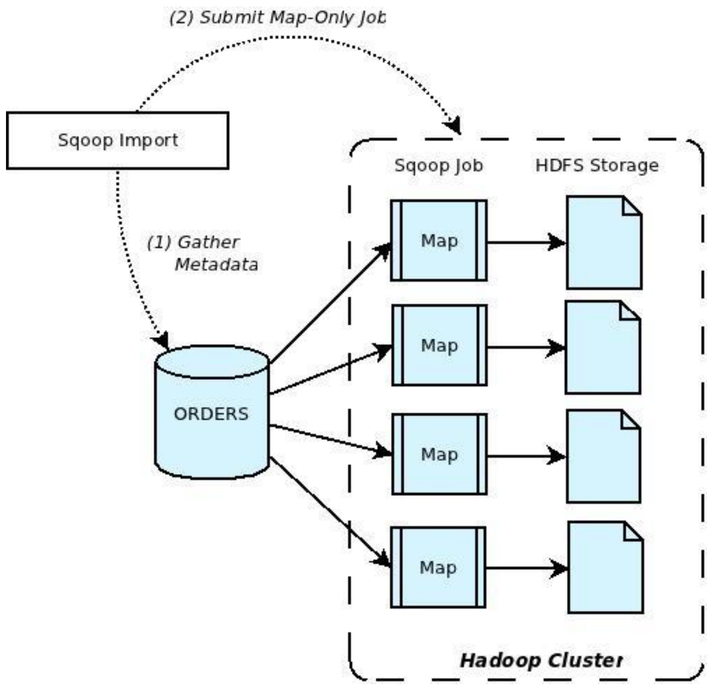
Sqoop Import

We create innovative software products that appeal to global audiences.

Globant
we are ready

Import - Overview

- The input to the import process is a **database table**.
- Sqoop will read the table **row-by-row** into HDFS.
- The output of this import process is a **set of files** containing a copy of the imported table.
- The import process is **performed in parallel**. For this reason, the output will be in multiple files.
- These files may be delimited text files (for example, with commas or tabs separating each field), or binary Avro or SequenceFiles containing serialized record data.



Import - Details

- A **Java class** is generated during the import process, which can encapsulate one row of the imported table. This class is used during the import process.
- The Java **source code** for this class is also **provided** so it can be used in subsequent MapReduce processing of the data.
- This class can serialize and deserialize data to and from the SequenceFile format.
- It can also parse the delimited-text form of a record.

- It is possible to control the specific row range or columns imported.
- We can specify particular delimiters and escape characters for the file-based representation of the data, as well as the file format used.
- It is also possible to control the class or package names used in generated code.

Import - Some Arguments

- **--append**: Append data to an existing dataset in HDFS
- **--columns <col,col,col…>**: Columns to import from table
- **--table <table-name>**: Table to read
- **-e,--query <statement>**: Import the results of statement.
- **--where <where clause>**: WHERE clause to use during import
- **-m,--num-mappers <n>**: Use n map tasks to import in parallel

Limited to simple queries where there are no ambiguous projections and no OR conditions in the WHERE clause.

Import - Example (1/2)

```
$ sqoop-import (generic-args) (import-args)
$ sqoop import --connect jdbc:mysql://database.example.com/employees
--username aaron -P

$ sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id ==
b.id) WHERE $CONDITIONS' -m 1 --target-dir /user/foo/joinresults
```

- Sqoop natively supports several databases.
- Sqoop can be used with any JDBC-compliant database.

Import - Example (2/2)

```
$ sqoop-import (generic-args) (import-args)
$ sqoop import --connect jdbc:mysql://database.example.com/employees
--username aaron -P

$ sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id ==
b.id) WHERE $CONDITIONS' -m 8 --target-dir /user/foo/joinresults
```

Default: four tasks are used. Some databases may see improved performance by increasing this value to 8 or 16.
Do not increase the degree of parallelism higher than that available within your MapReduce cluster. Likewise, do not increase the degree of parallelism higher than that which your database can reasonably support.

Import - Incremental Imports

Retrieve only rows newer than some previously-imported set of rows.

- **--check-column <col>**: Specifies the column to be examined when determining which rows to import.
- **--incremental <mode>**: Specifies how Sqoop determines which rows are new. Legal values for mode include **append** and **lastmodified**.
- **--last-value <value>**: Specifies the maximum value of the check column from the previous import.

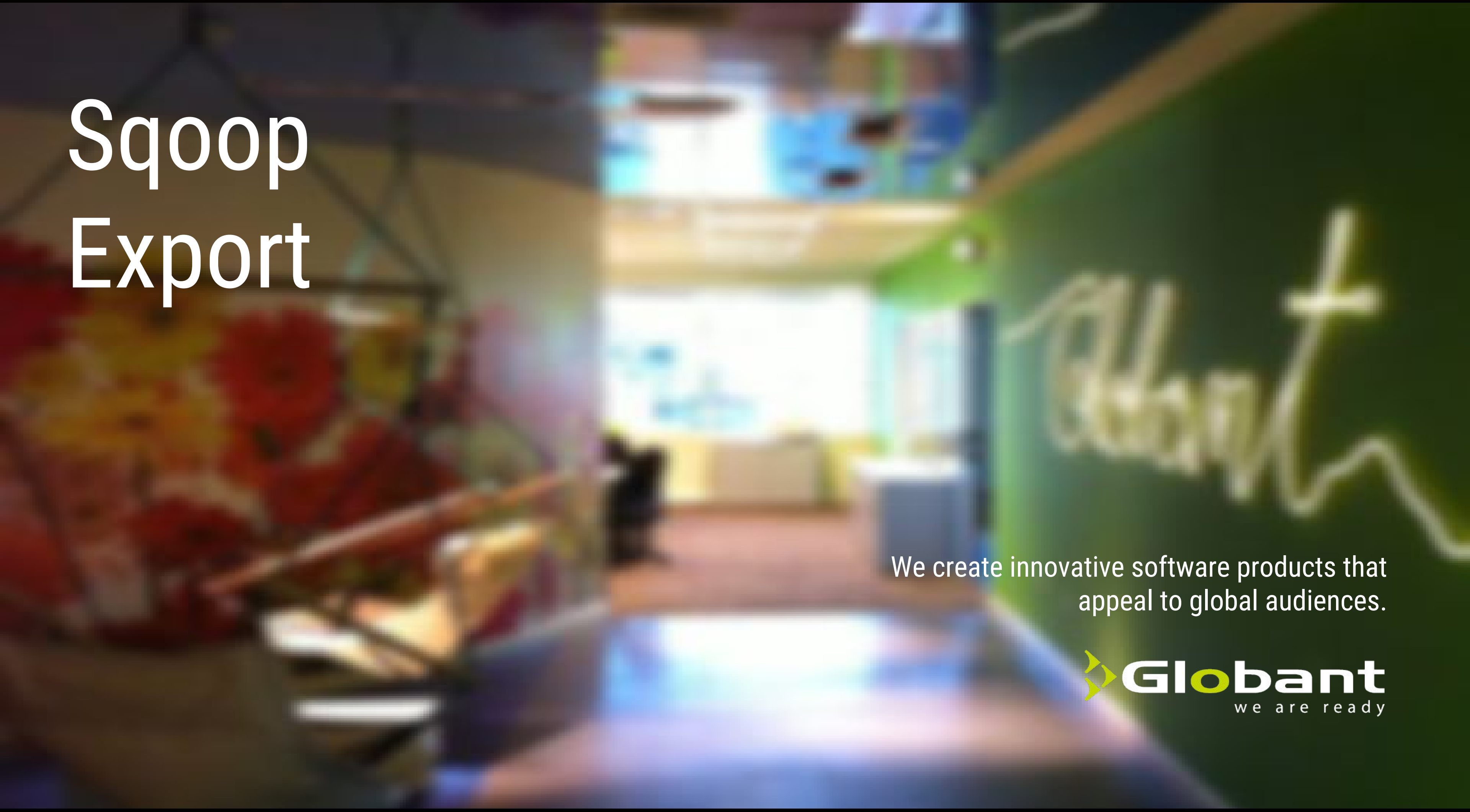
At the end of an incremental import, the value which should be specified as --last-value for a subsequent import is printed to the screen.

Import - Import all tables

The following conditions must be met:

- Each table must have a **single-column primary key**.
- You must intend to import **all columns** of each table.
- You must not intend to use non-default splitting column, nor impose any conditions via a WHERE clause.

```
$ sqoop import-all-tables --connect jdbc:mysql://db.foo.com/corp
Folder found in hdfs:
/user/someuser/EMPLOYEES
/user/someuser/PAYCHECKS
/user/someuser/OFFICE_SUPPLIES
```



Sqoop Export

We create innovative software products that appeal to global audiences.

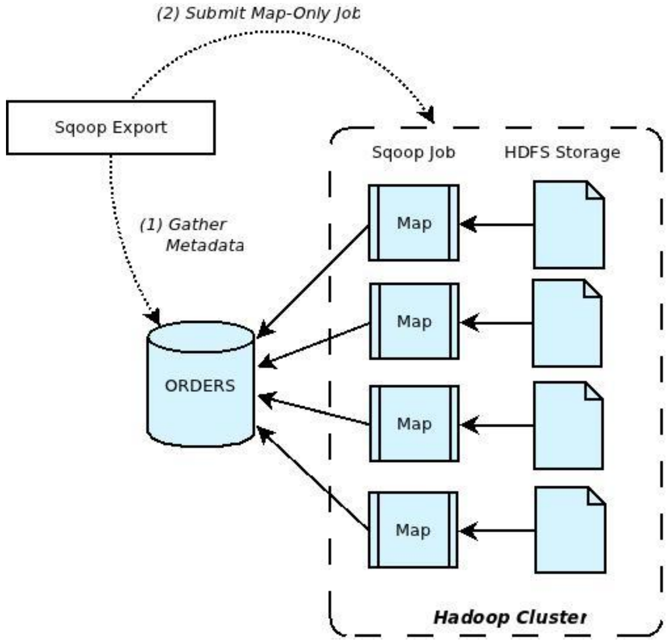
Globant
we are ready

Export - Overview

The export tool exports a set of files from HDFS back to an RDBMS.

Sqoop's export process will:

- Read a set of delimited text files from HDFS in parallel.
- Parse them into records.
- Insert them as new rows in a target database table.



Export - Considerations

- The target **table must exist** in the database.
- The default operation is to transform these into a set of INSERT statements that inject the records into the database.
- In "update mode," Sqoop will generate UPDATE statements that replace existing records in the database.
- In "call mode" Sqoop will make a stored procedure call for each record.

Export - Some Arguments

- **--export-dir <dir>**: HDFS source path for the export
- **--table <table-name>**: Table to populate
- **--update-mode <mode>**: Specify how updates are performed when new rows are found with non-matching keys in database. Legal values for mode include **updateonly** (default) and **allowinsert**.
- **-m,--num-mappers <n>**: Use n map tasks to export in parallel

Export Parallelism: -m,--num-mappers
By default, Sqoop uses four tasks in parallel for the export process. Additional tasks may offer better concurrency. If the database is already bottlenecked on updating indices, invoking triggers, and so on, then additional load may decrease performance.

Export - Staging table (1/2)

- Sqoop breaks down export process into multiple transactions
- It is possible that a failed export job may result in partial data being committed to the database. This can further lead to subsequent jobs failing due to insert collisions in some cases, or lead to duplicated data in others.
- Solution: specifying a staging table via the **`--staging-table`** option which acts as an auxiliary table that is used to stage exported data. The staged data is finally moved to the destination table in a single transaction.

Export - Staging table (2/2)

- You must create the staging table prior to running the export job.
- This table must be structurally identical to the target table.
- This table should either be empty before the export job runs, or the **`--clear-staging-table`** option must be specified.
- If the staging table contains data and the **`--clear-staging-table`** option is specified, Sqoop will delete all of the data before starting the export job.

Not available when export is invoked using the **--update-key** option for updating existing data, and when stored procedures are used to insert the data.

Export - Insert and Updates

- Default: `sqoop-export` appends new rows to a table; each input record is transformed into an INSERT statement that adds a row to the target database table.
- If your table has constraints (PK, FK, unique) and already contains data, you must take care to avoid inserting records that violate these constraints.
- The export process will fail if an INSERT statement fails. This mode is primarily intended for exporting records to a new, empty table intended to receive these results.

- If you specify the `--update-key` argument, Sqoop will instead modify an existing dataset in the database. Each input record is treated as an UPDATE statement that modifies an existing row.
- The row a statement modifies is determined by the column name(s) specified with `--update-key`.

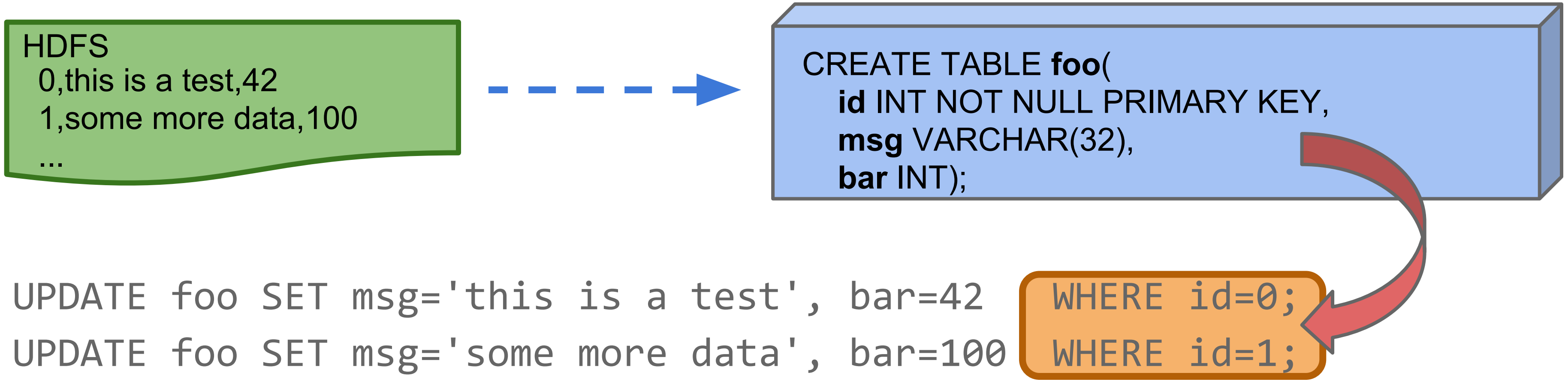
Export - Insert Example

```
$ sqoop export --connect jdbc:mysql://db.example.com/foo --table bar
--export-dir /results/bar_data
```

Sqoop performs a set of INSERT INTO operations, without regard for existing content. If Sqoop attempts to insert rows which violate constraints in the database (for example, a particular primary key value already exists), then **the export fails**.

Export - Update-based Example

```
$ sqoop-export --table foo --update-key id --export-dir
/path/to/data --connect
```



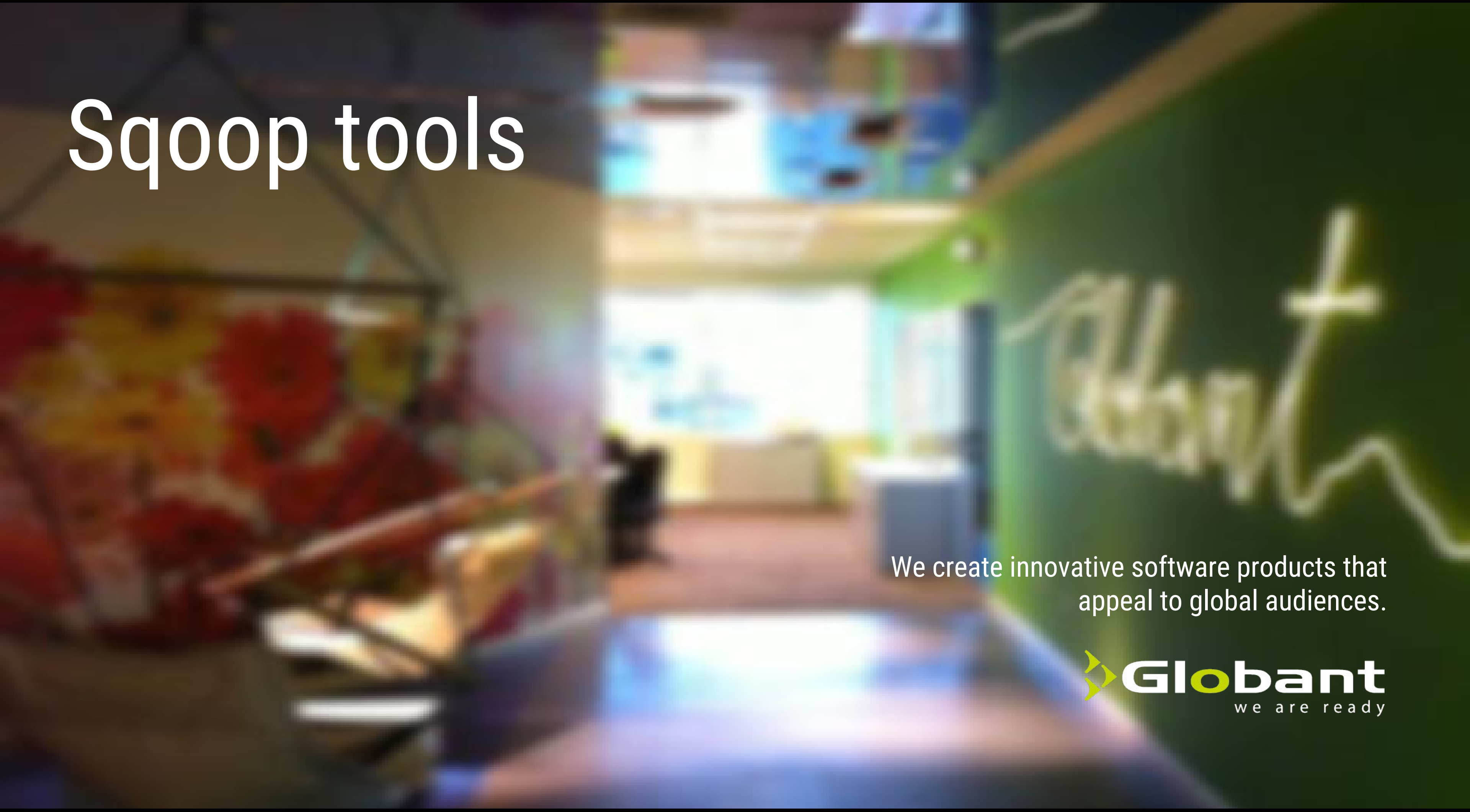
```
HDFS
  0,this is a test,42
  1,some more data,100
  ...
```

```
CREATE TABLE foo(
    id INT NOT NULL PRIMARY KEY,
    msg VARCHAR(32),
    bar INT);
```

```
UPDATE foo SET msg='this is a test', bar=42      WHERE id=0;
UPDATE foo SET msg='some more data', bar=100     WHERE id=1;
```

- An update-based export will **not** insert new rows into the database.
- Use `--update-mode` argument with **`allowinsert`** mode if you want to update rows if they exist in the database already or insert rows if they do not exist yet.

Sqoop tools



We create innovative software products that
appeal to global audiences.

Globant
we are ready

Sqoop Tools and Commands

Some commands:

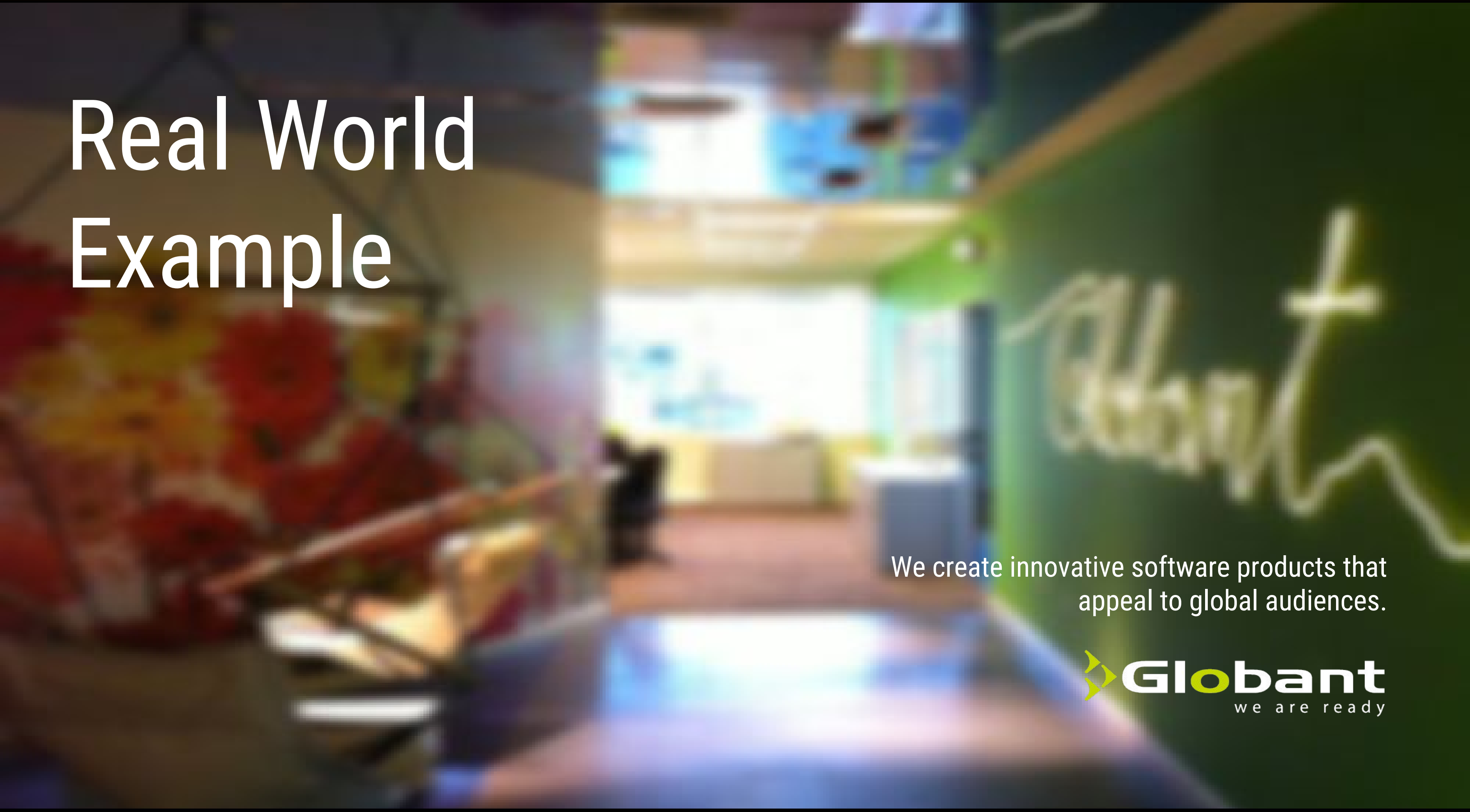
- **codegen**: Generate code to interact with database records
- **create-hive-table**: Import a table definition into Hive
- **eval**: Evaluate a SQL statement and display the results
- **export**: Export an HDFS directory to a database table
- **import**: Import a table from a database to HDFS
- **import-all-tables**: Import tables from a database to HDFS
- **list-databases**: List available databases on a server
- **list-tables**: List available tables in a database

Equivalent Commands and Alias

```
sqoop help import
sqoop import --help
```

Alias:
"sqoop-import"

sqoop-codegen
sqoop-create-hive-table
sqoop-eval
sqoop-export
sqoop-help
sqoop-import sqoop-import-all-tables
sqoop-job
sqoop-list-databases
sqoop-list-tables
sqoop-merge sqoop-metastore
sqoop-version

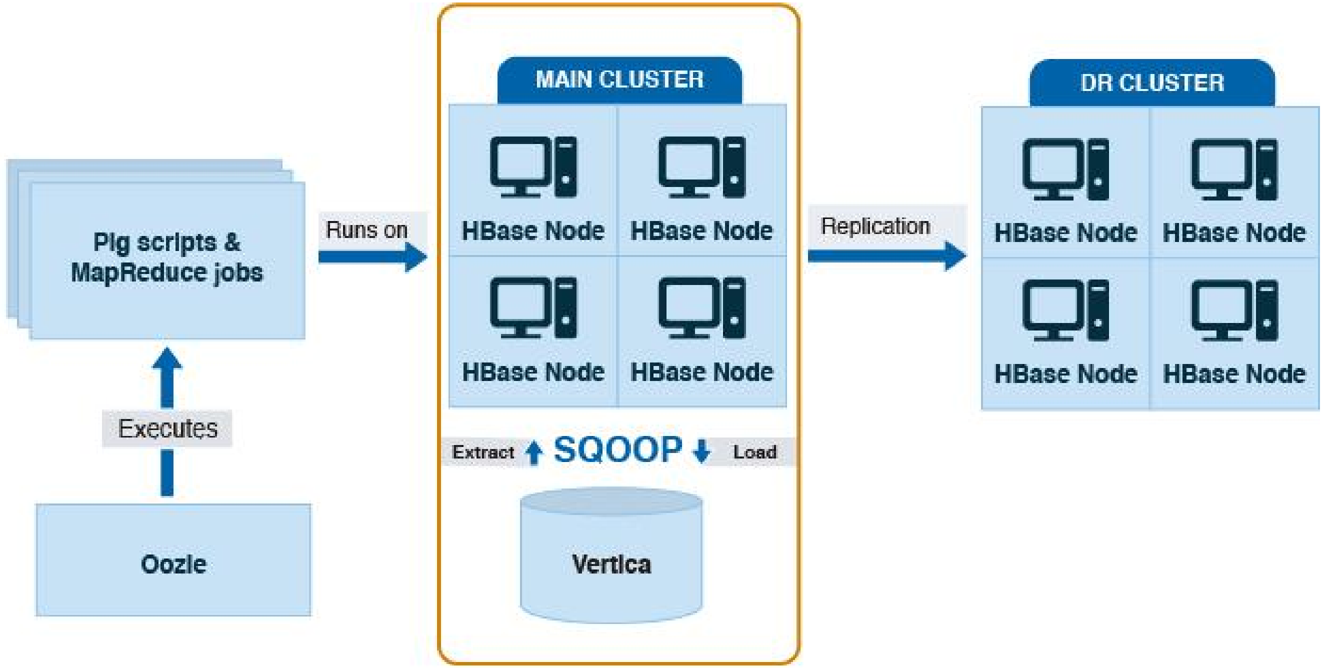


Real World Example

We create innovative software products that appeal to global audiences.

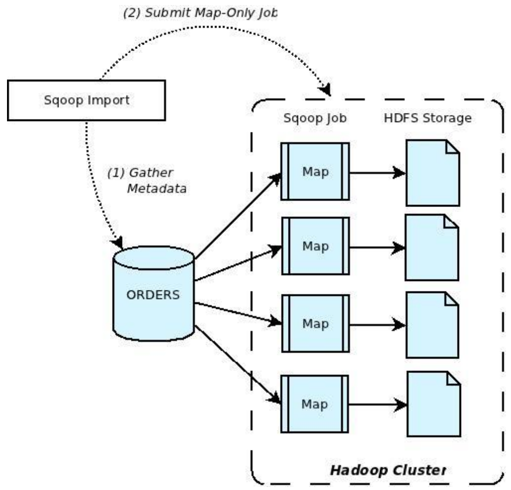
Globant
we are ready

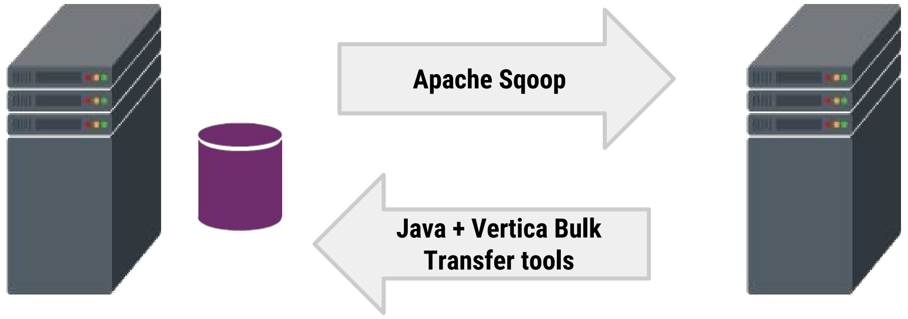
Real World Example

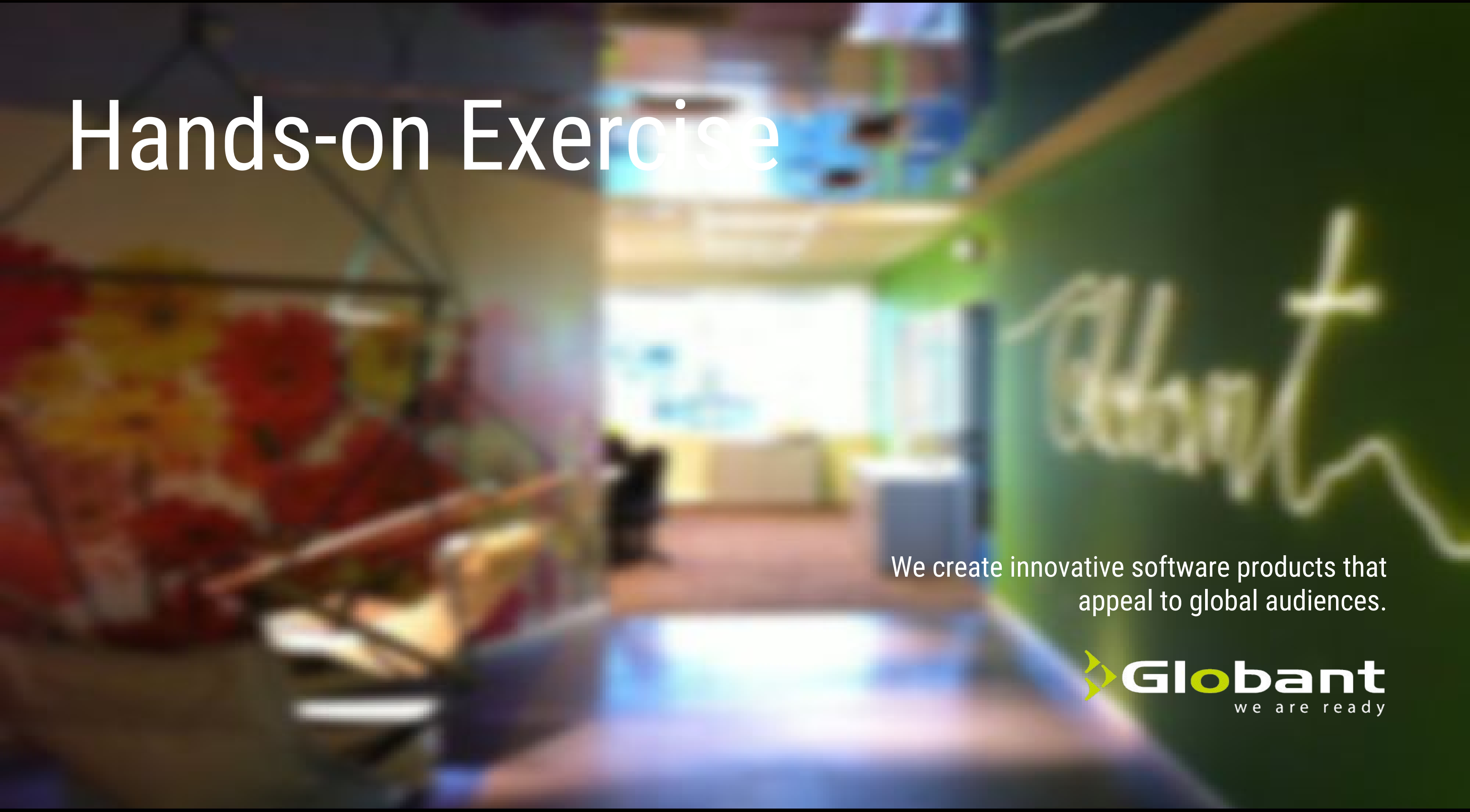


Real World Example

```
time sqoop import
    --connect
        "jdbc:vertica://host:port/database"
    --username
        user -P
    --connection-manager
        net.jpmchase.sqoop.manager.VerticaManager
    --table
        input_database.input_table
    --split-by
        key
    --m
        18
    --verbose
```







Hands-on Exercise

We create innovative software products that appeal to global audiences.

Globant
we are ready

Sqoop - Exercise 1

1A. Import a dataset from MySql into HDFS

Given a real dataset that contains data about Crimes in Chicago from 2001 till now (already stored in MySQL), import into HDFS **only** the crimes of 2005.

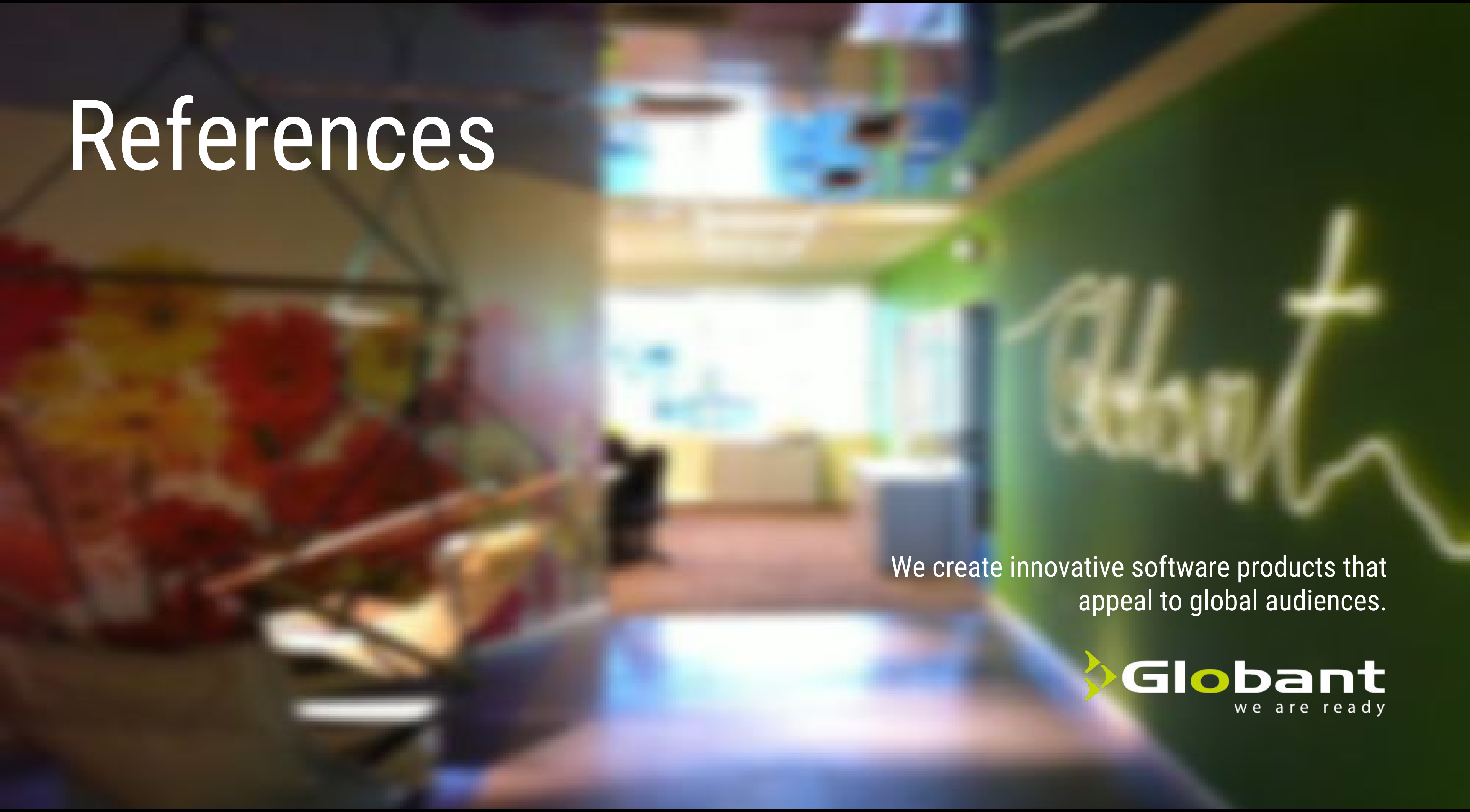
- Dataset: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
- MySQL Table: bdtraining.chicago_crimes

1B. Export a dataset from HDFS into MySQL

Export the data that you have stored in HDFS in the previous exercise into a new table in MySQL.

Hint 1: Check the output that Sqoop writes to command line, there's always useful data there.

Hint 2: To improve the performance, vary the number of mappers by applying different values to --num-mappers <n>.



References

We create innovative software products that appeal to global audiences.

Globant
we are ready

References

- **Sqoop User Guide.** Apache - http://sqoop.apache.org/docs/1.4.3/SqoopUserGuide.html
- **Hadoop: The Definitive Guide, 2nd Edition** (Chapter 15). O'Reilly Media / Yahoo Press - Online @ Globant's Big Data Training Site
- **Apache Sqoop Cookbook**. O'Reilly Media - Online @ Globant's Big Data Training Site

Thanks