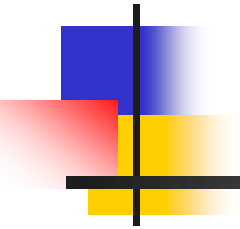


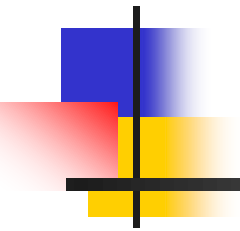
Data Mining – Chapter 4

Kyuseok Shim

Seoul National University

http://kdd.snu.ac.kr/~shim

Extended from the slides of the book "Data Mining: Concepts and Techniques (3rd ed.)" provided by Jiawei Han, Micheline Kamber, and Jian Pei



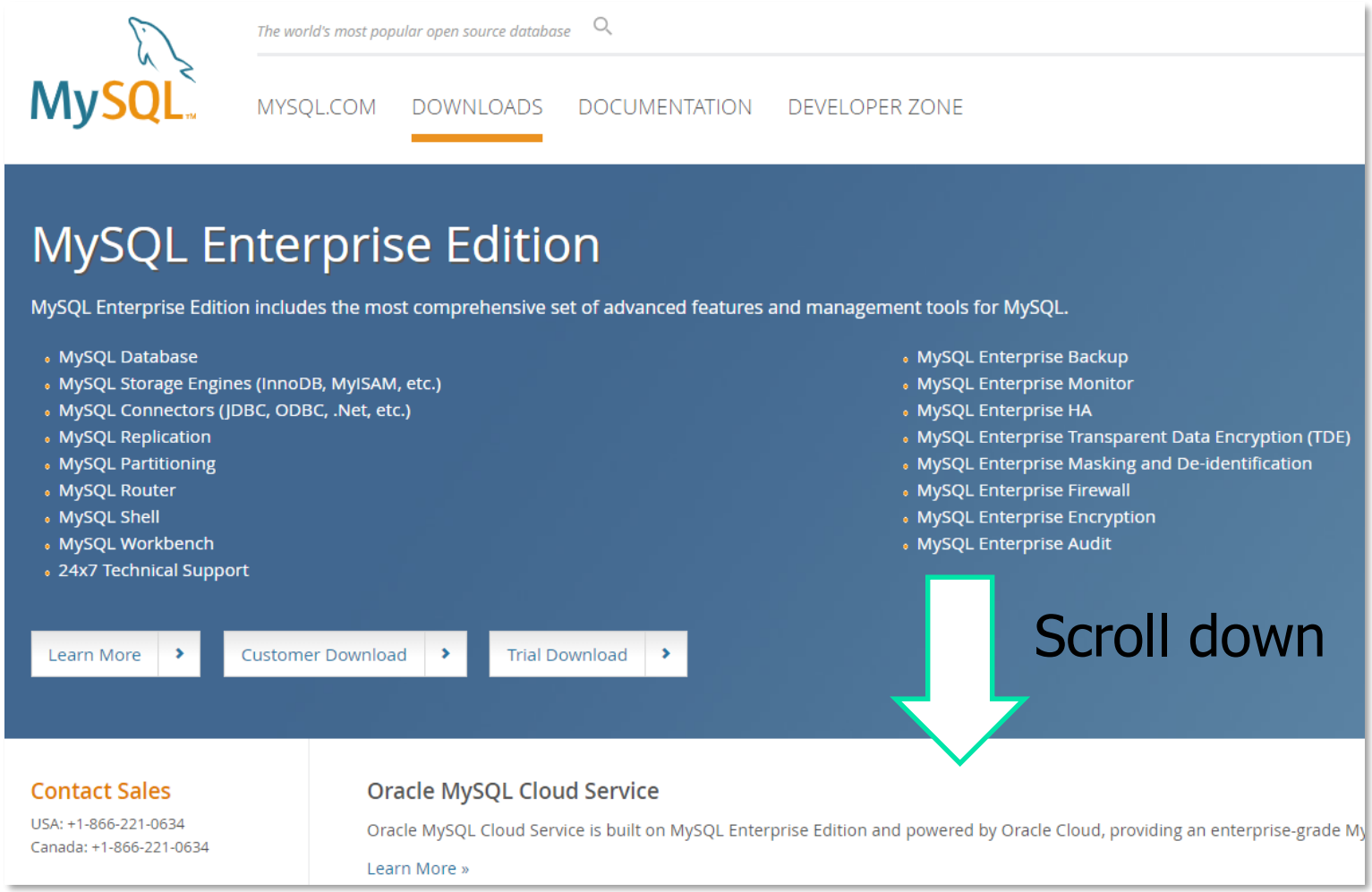
Install MySQL

MySQL

- Open-source relational database management system (RDBMS)

Install MySQL

- https://www.mysql.com/downloads/



Install MySQL

Oracle MySQL Cloud Service

Oracle MySQL Cloud Service is built on MySQL Enterprise Edition and powered by Oracle Cloud, providing an enterprise-grade MySQL database service.

Learn More »

MySQL Cluster CGE

MySQL Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

- MySQL Cluster
- MySQL Cluster Manager
- Plus, everything in MySQL Enterprise Edition

Learn More »
Customer Download » (Select Patches & Updates Tab, Product Search)
Trial Download »

MySQL Community (GPL) Downloads »

Install MySQL

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

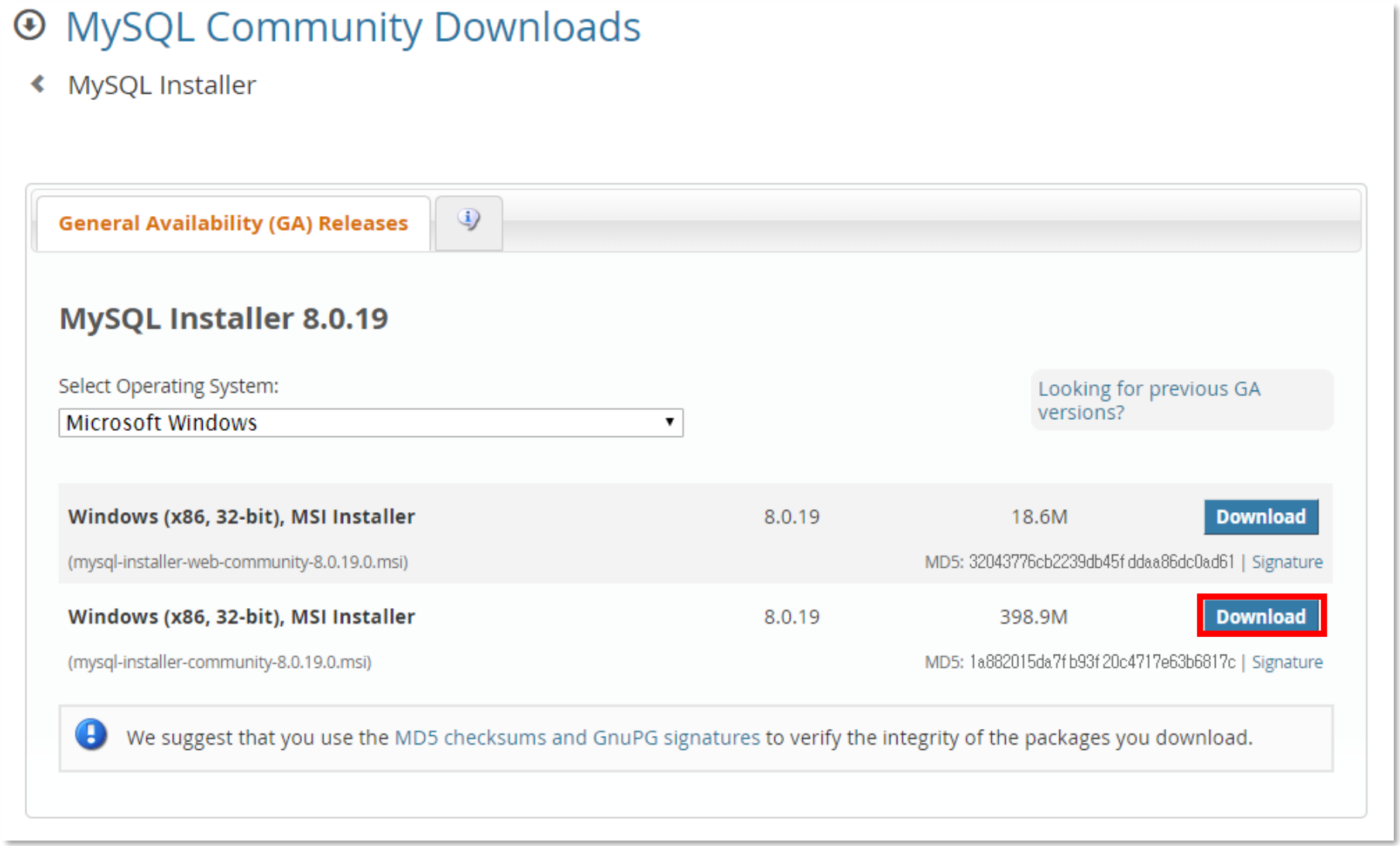
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench

- MySQL Installer for Windows
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Notifier

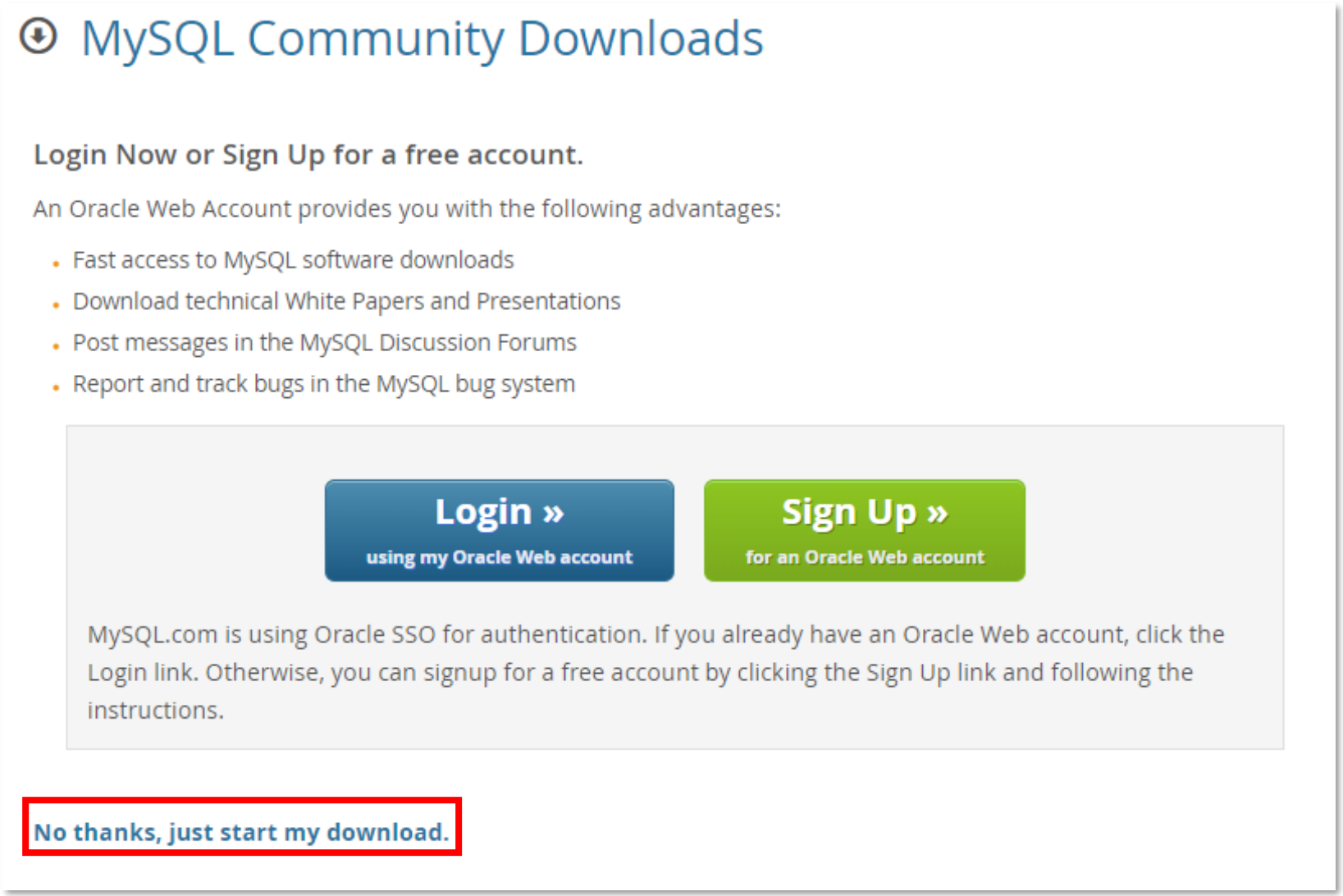
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

Install MySQL

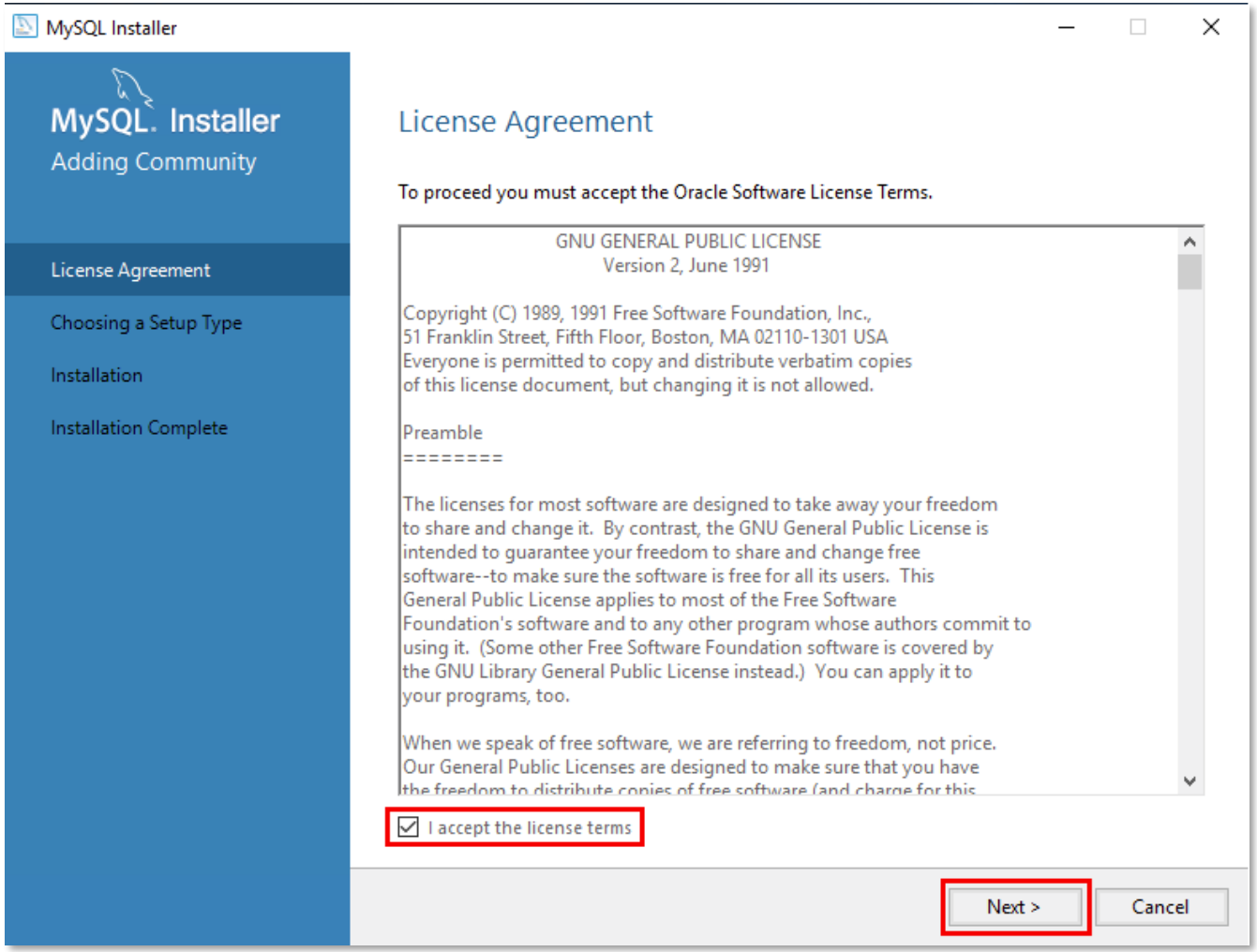


Install MySQL

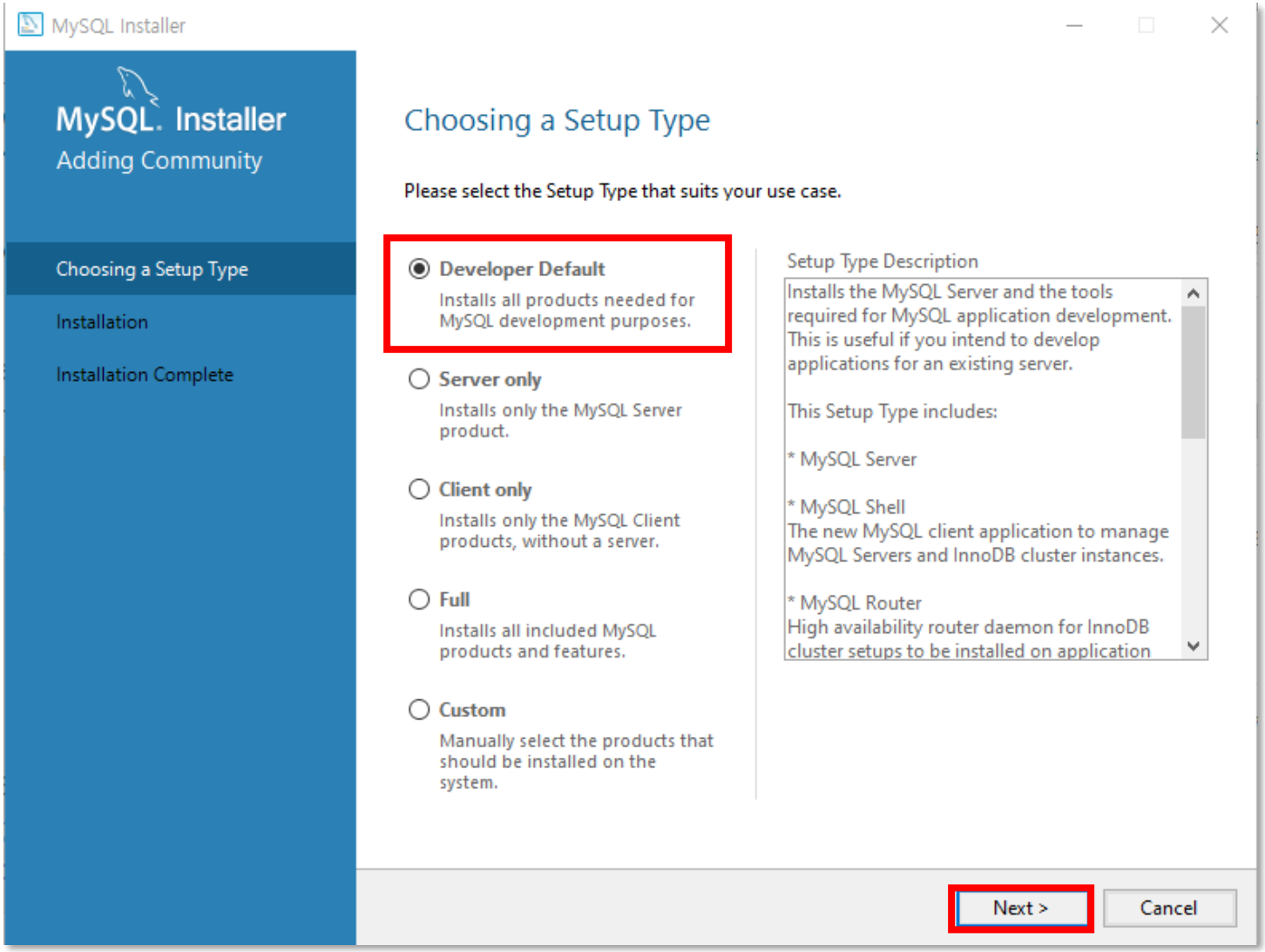


Install MySQL

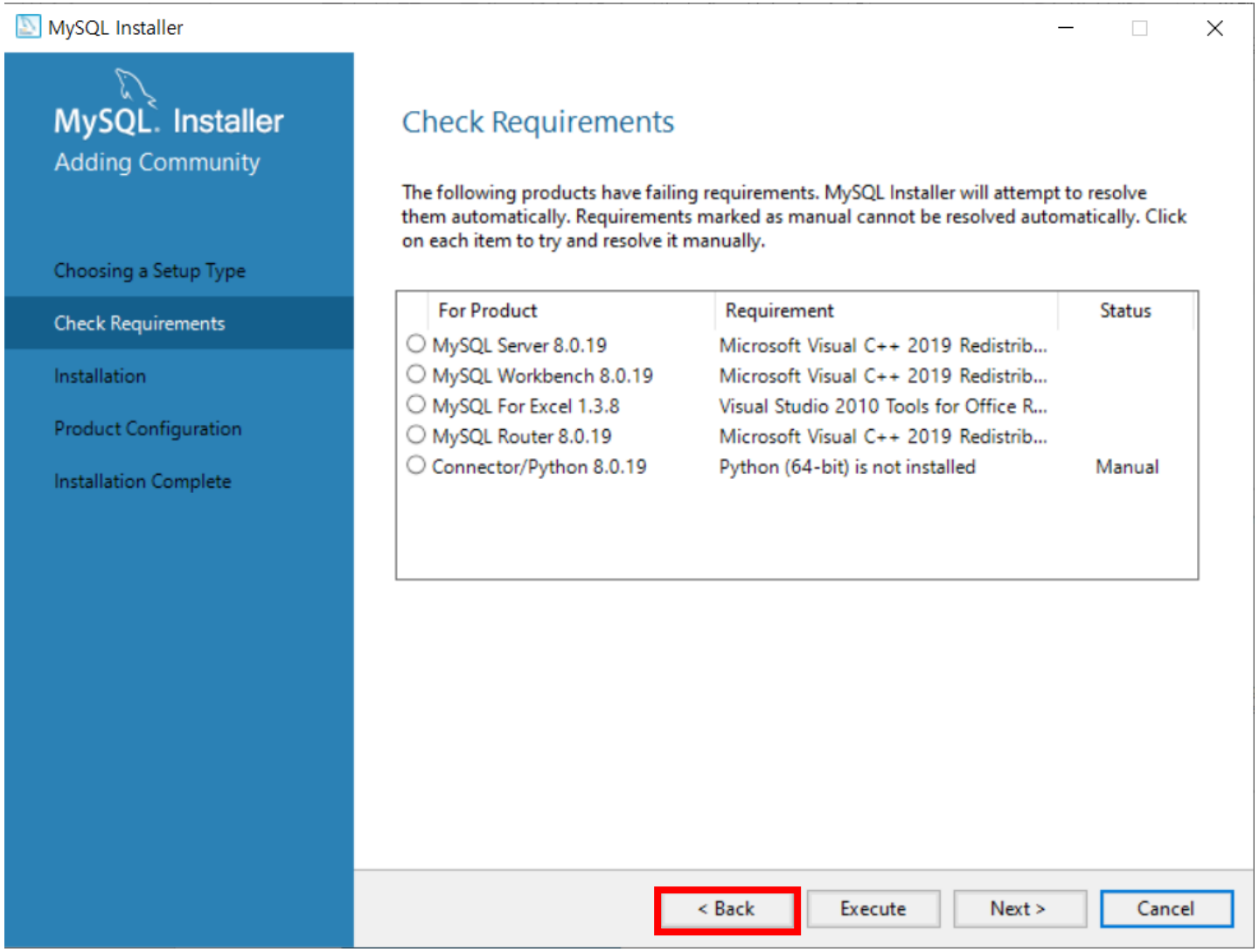
- Run the downloaded file



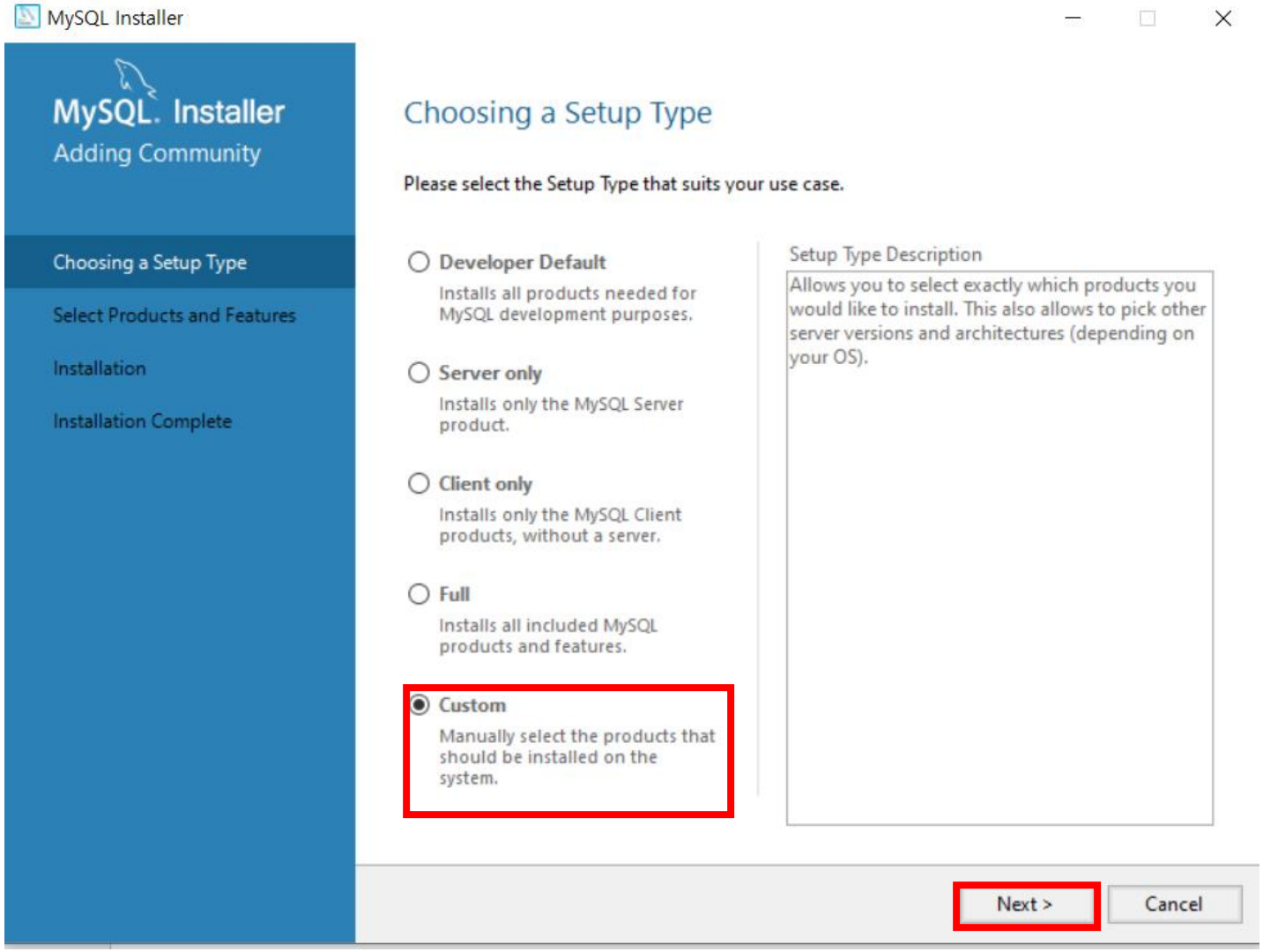
Install MySQL



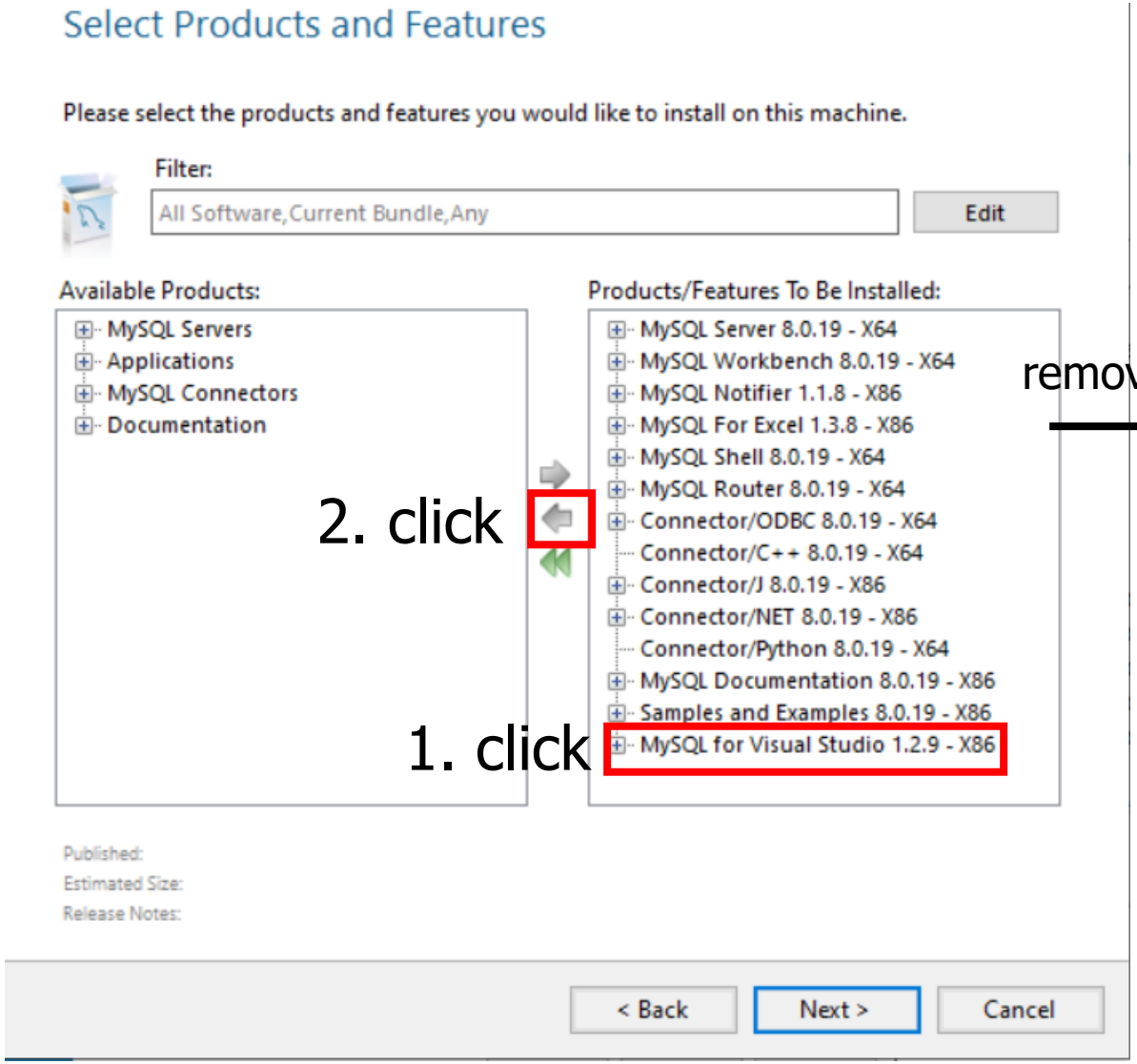
Install MySQL



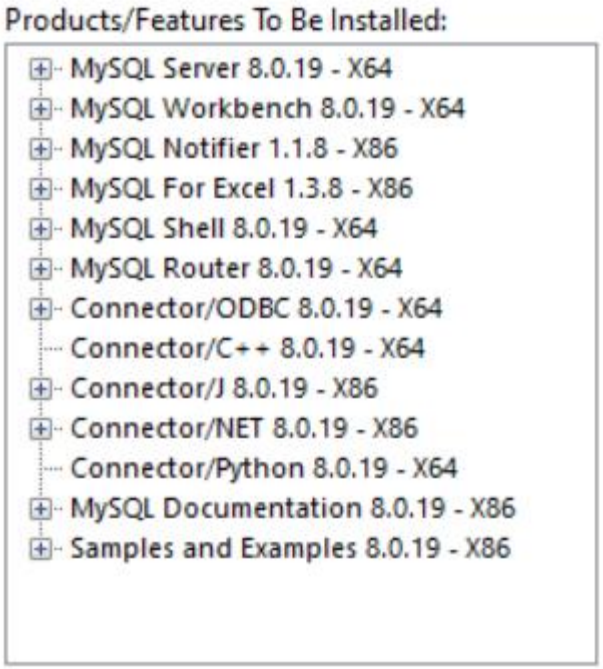
Install MySQL



Install MySQL

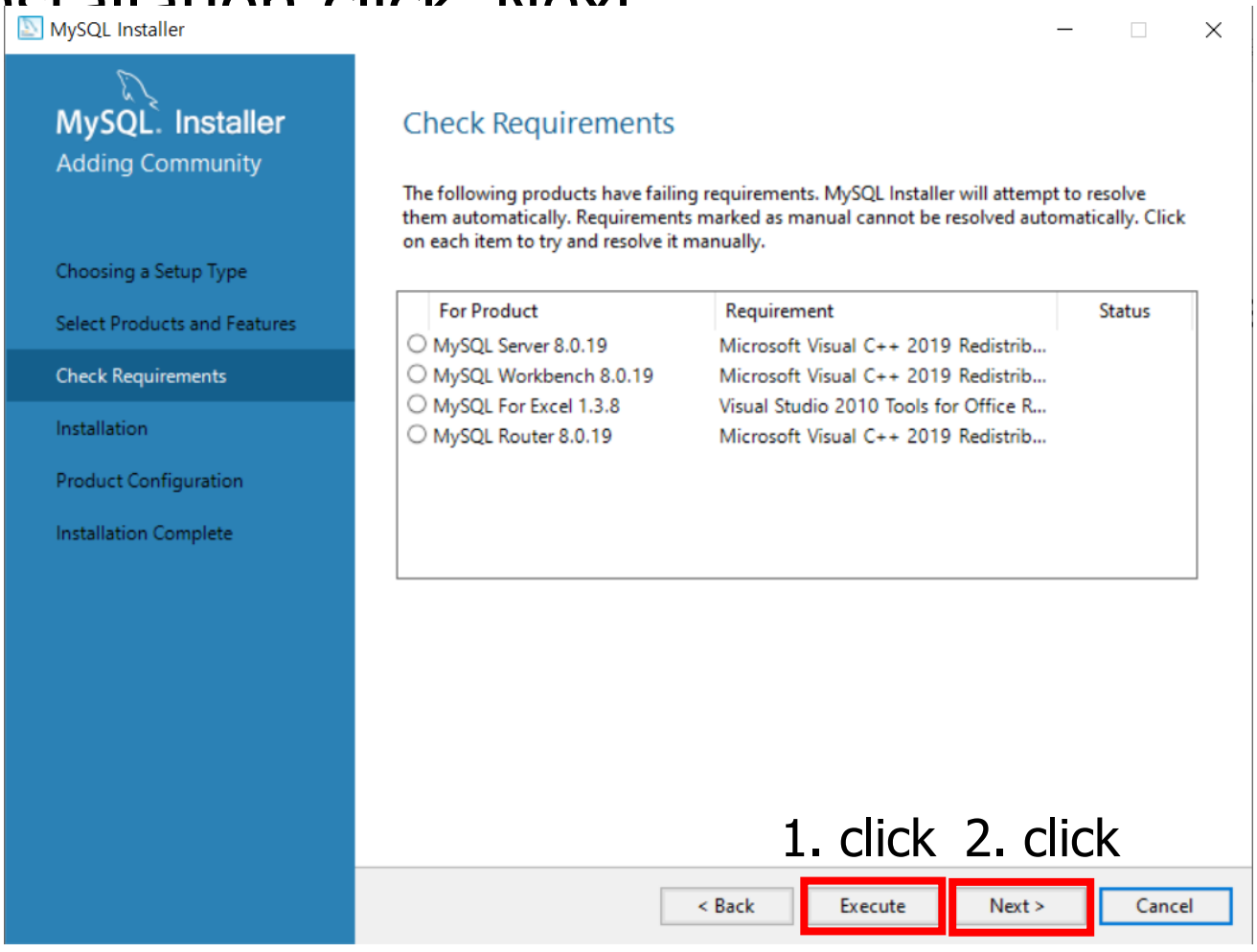


removed

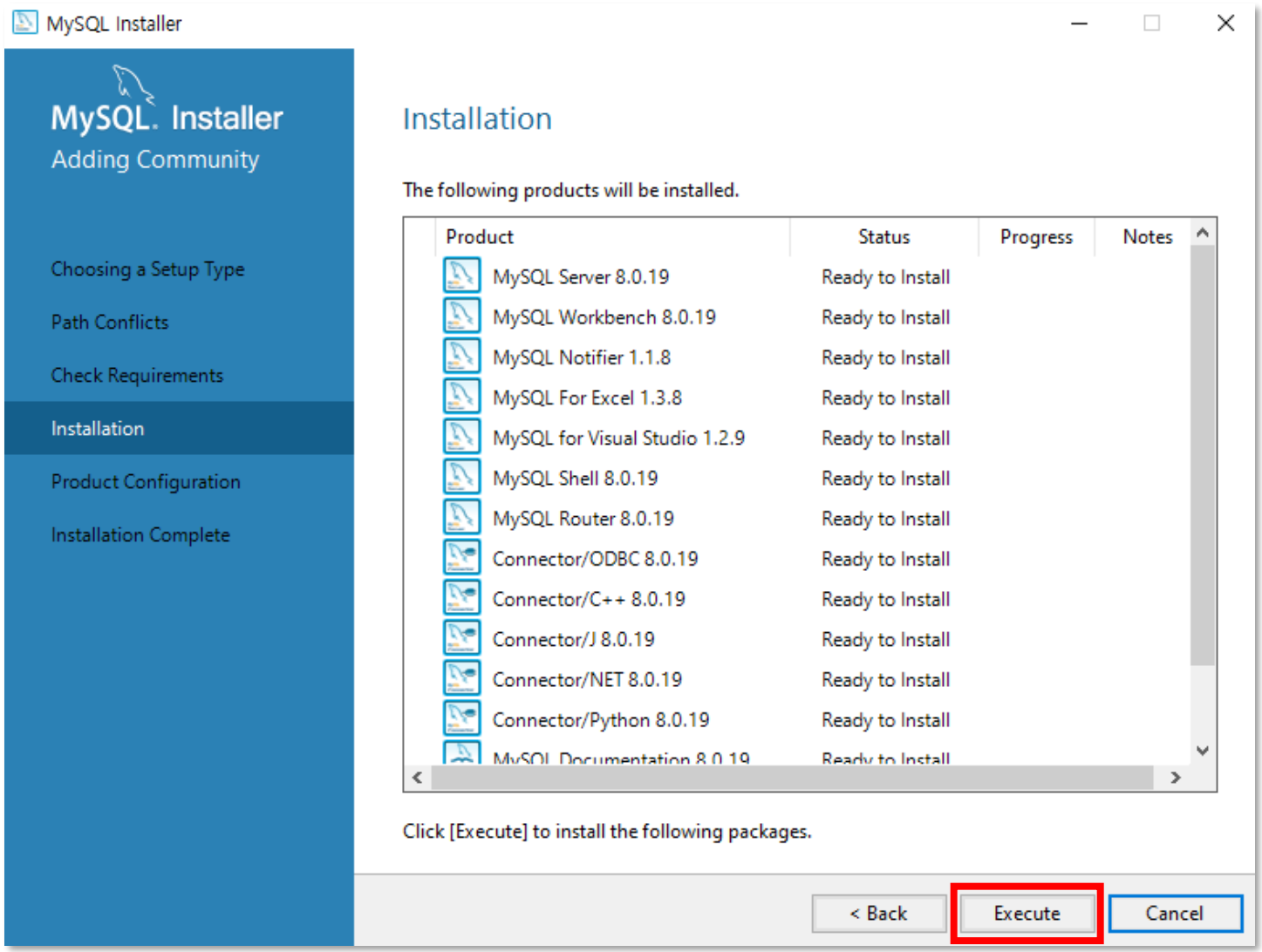


Install MySQL

- Click 'Execute' to install requirements and after installation click 'Next'

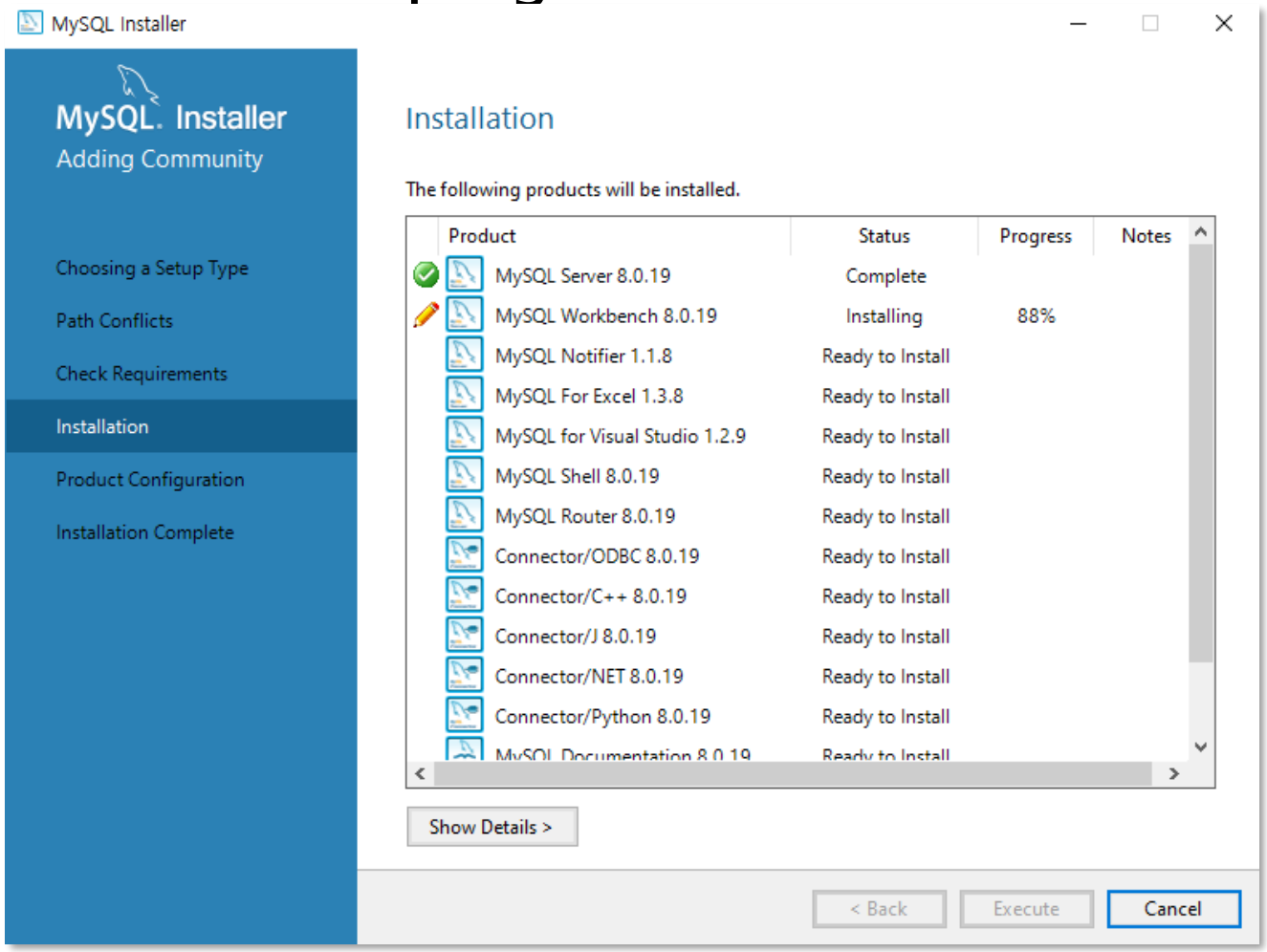


Install MySQL



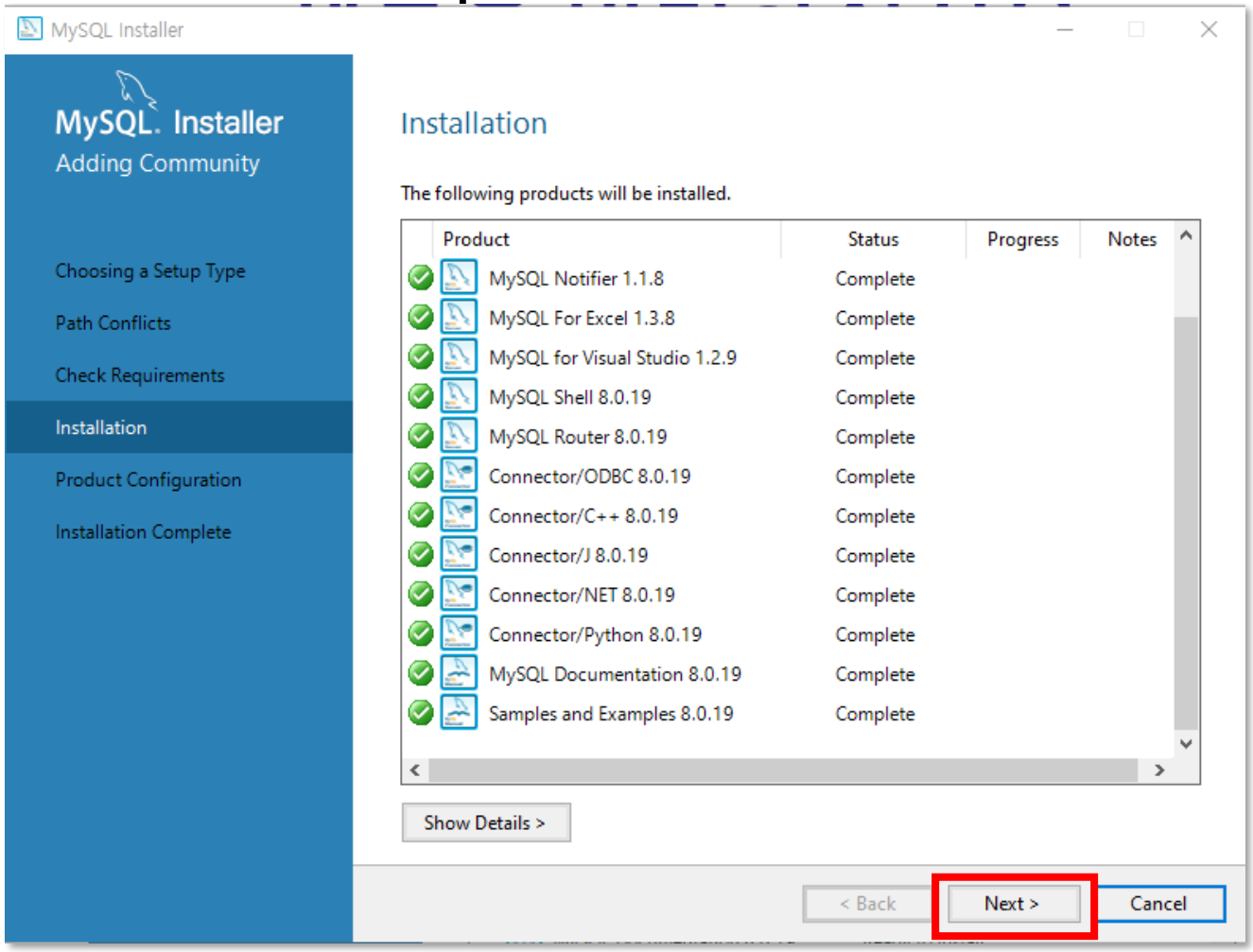
Install MySQL

- Installation is in progress

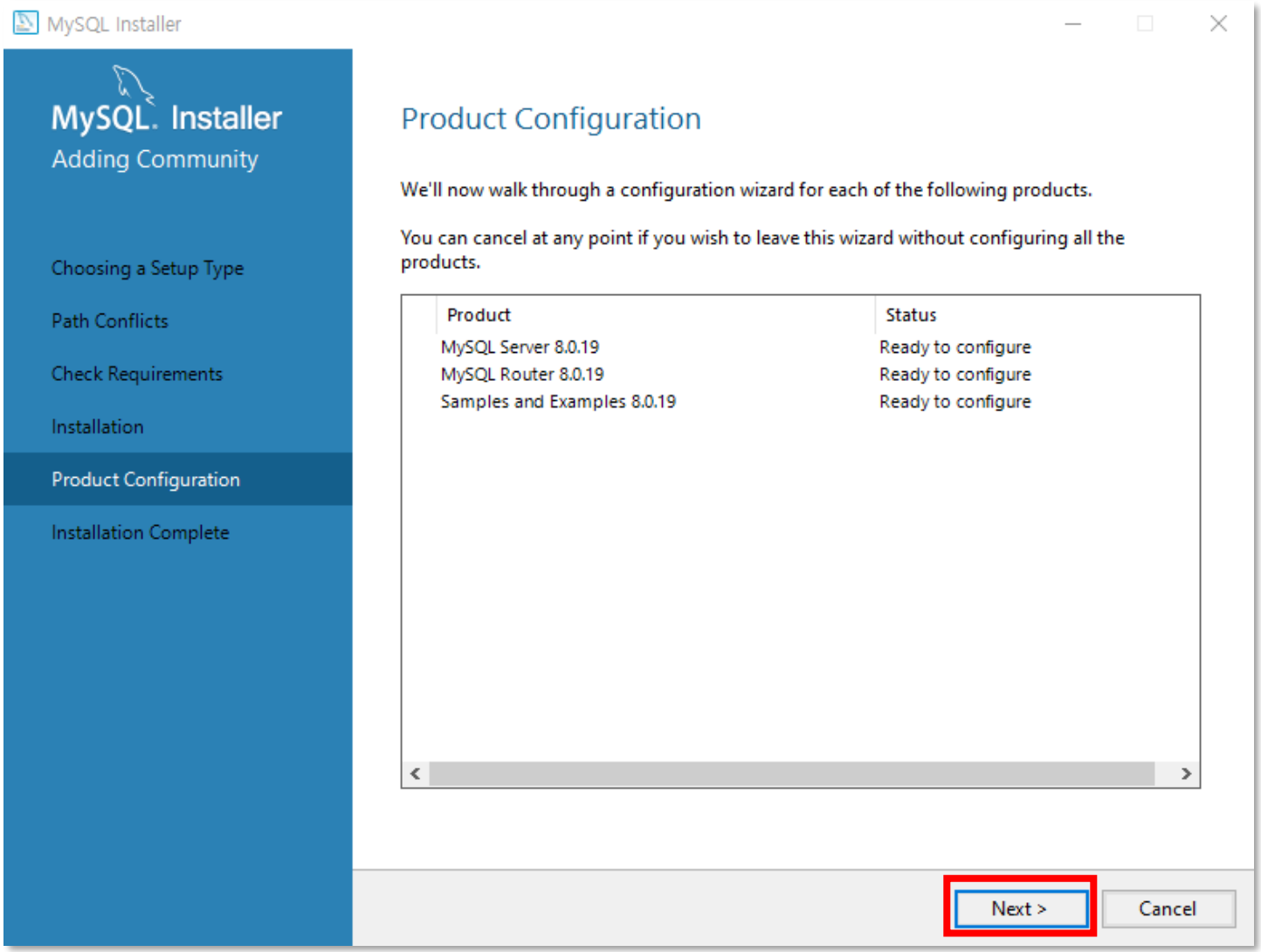


Install MySQL

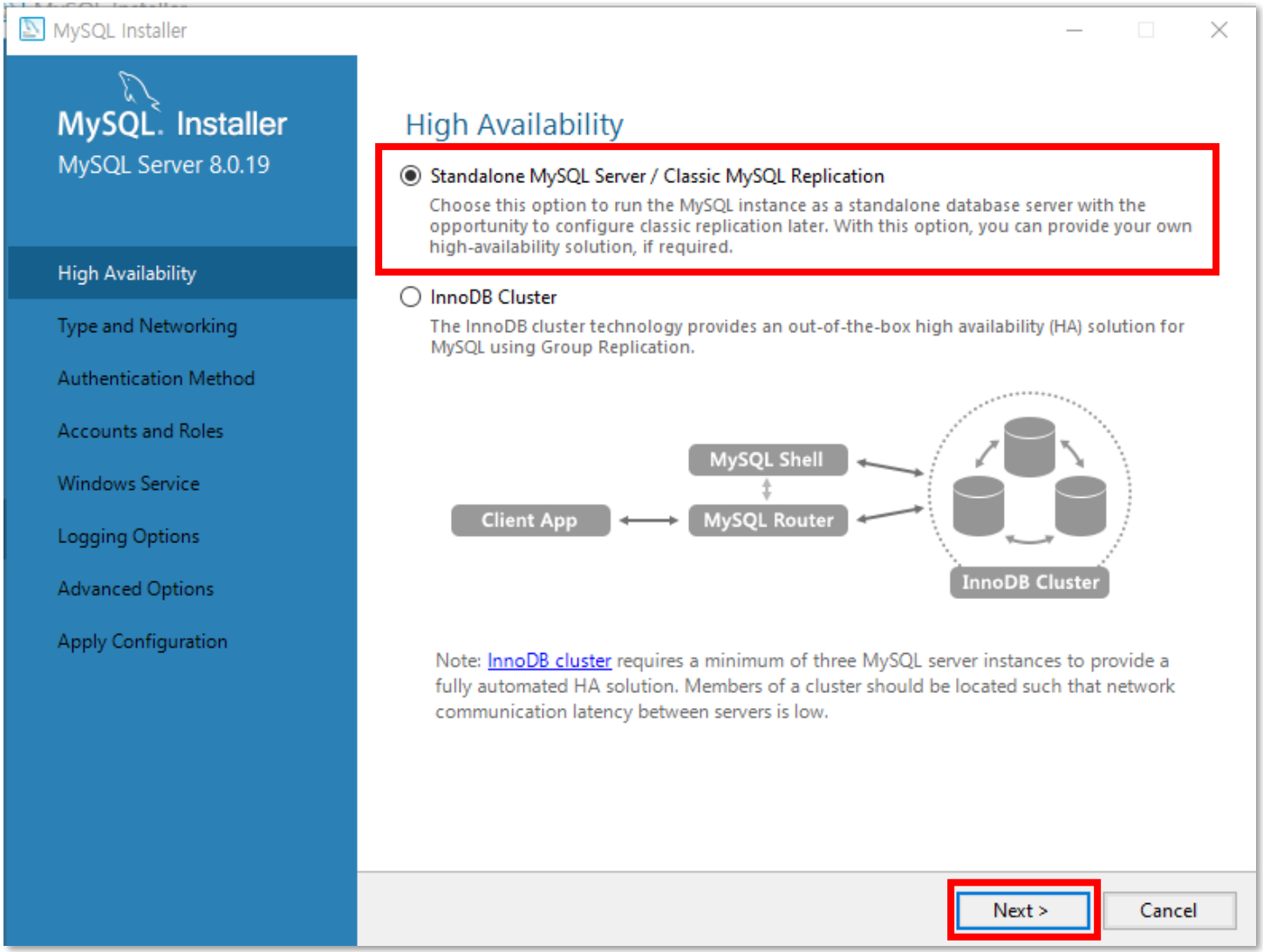
- Installation is complete



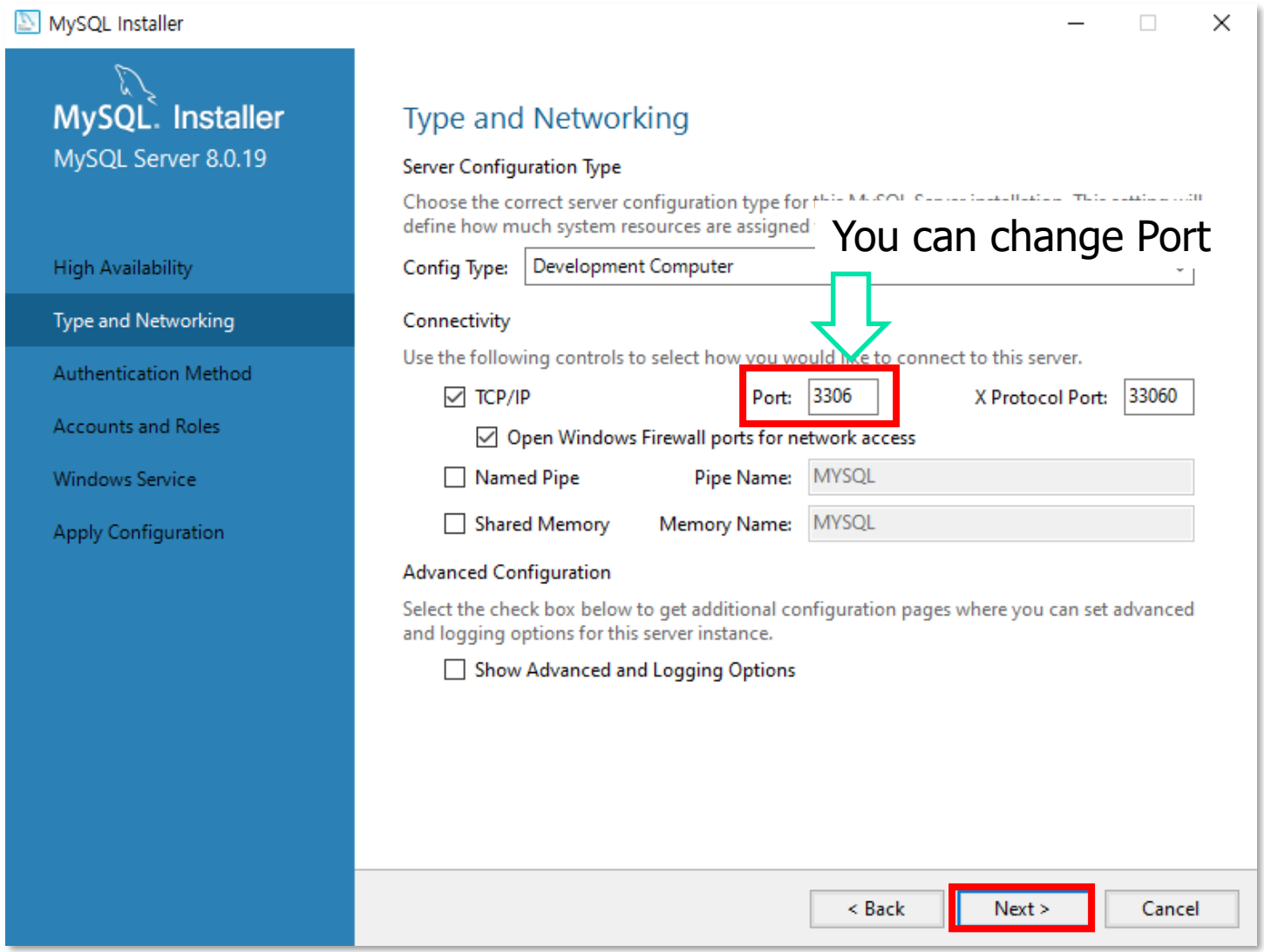
Install MySQL



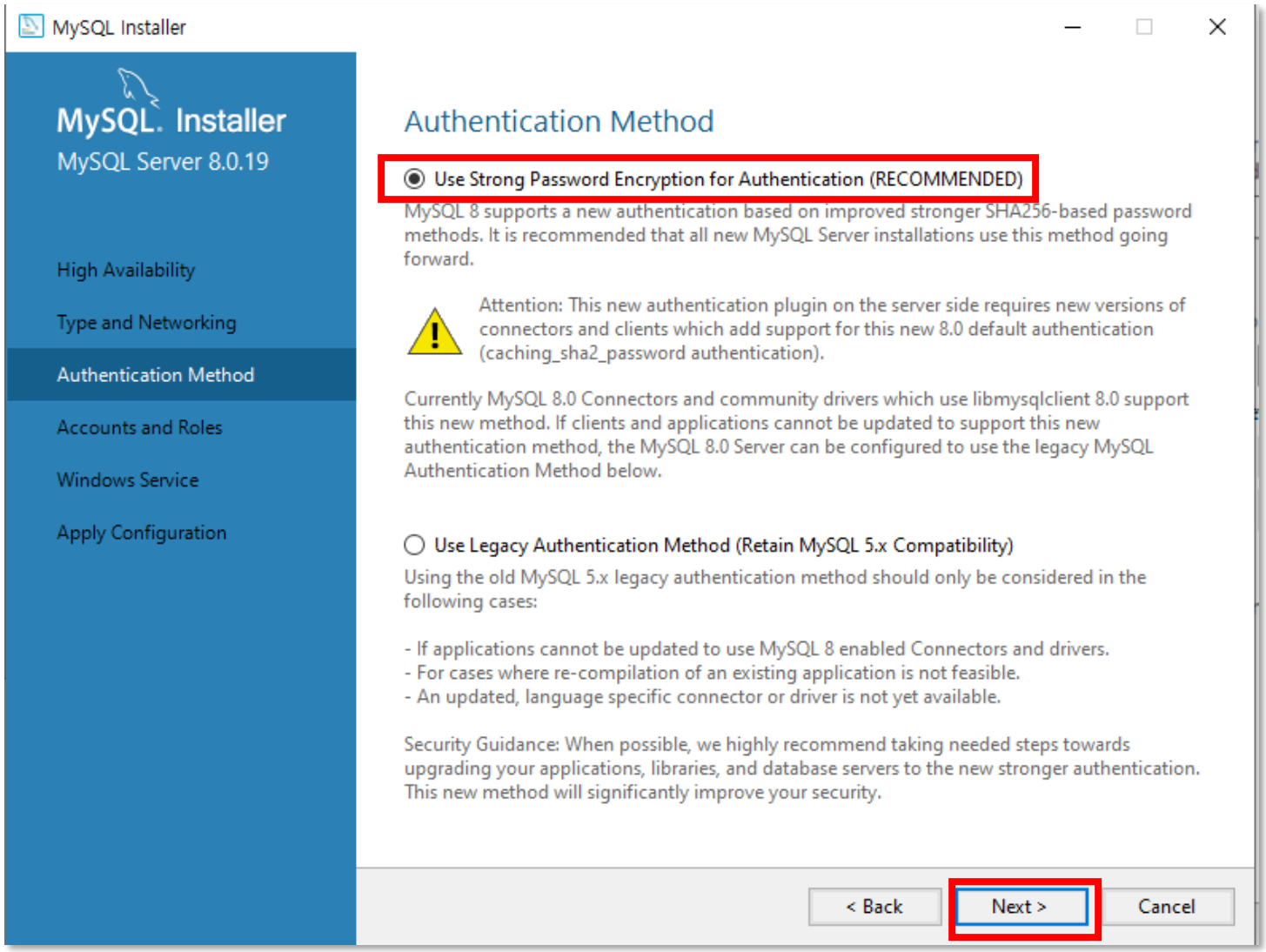
Install MySQL



Install MySQL

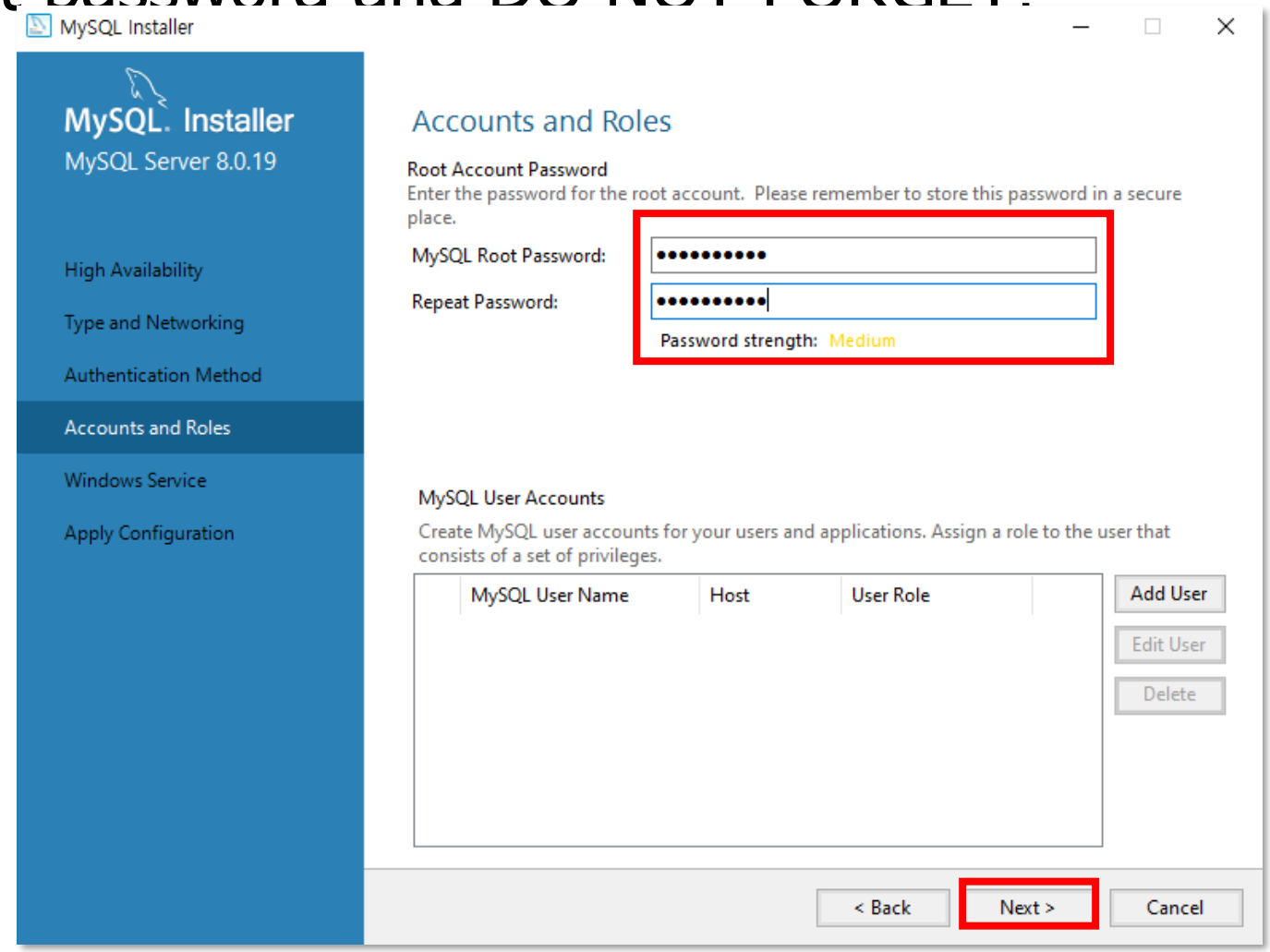


Install MySQL

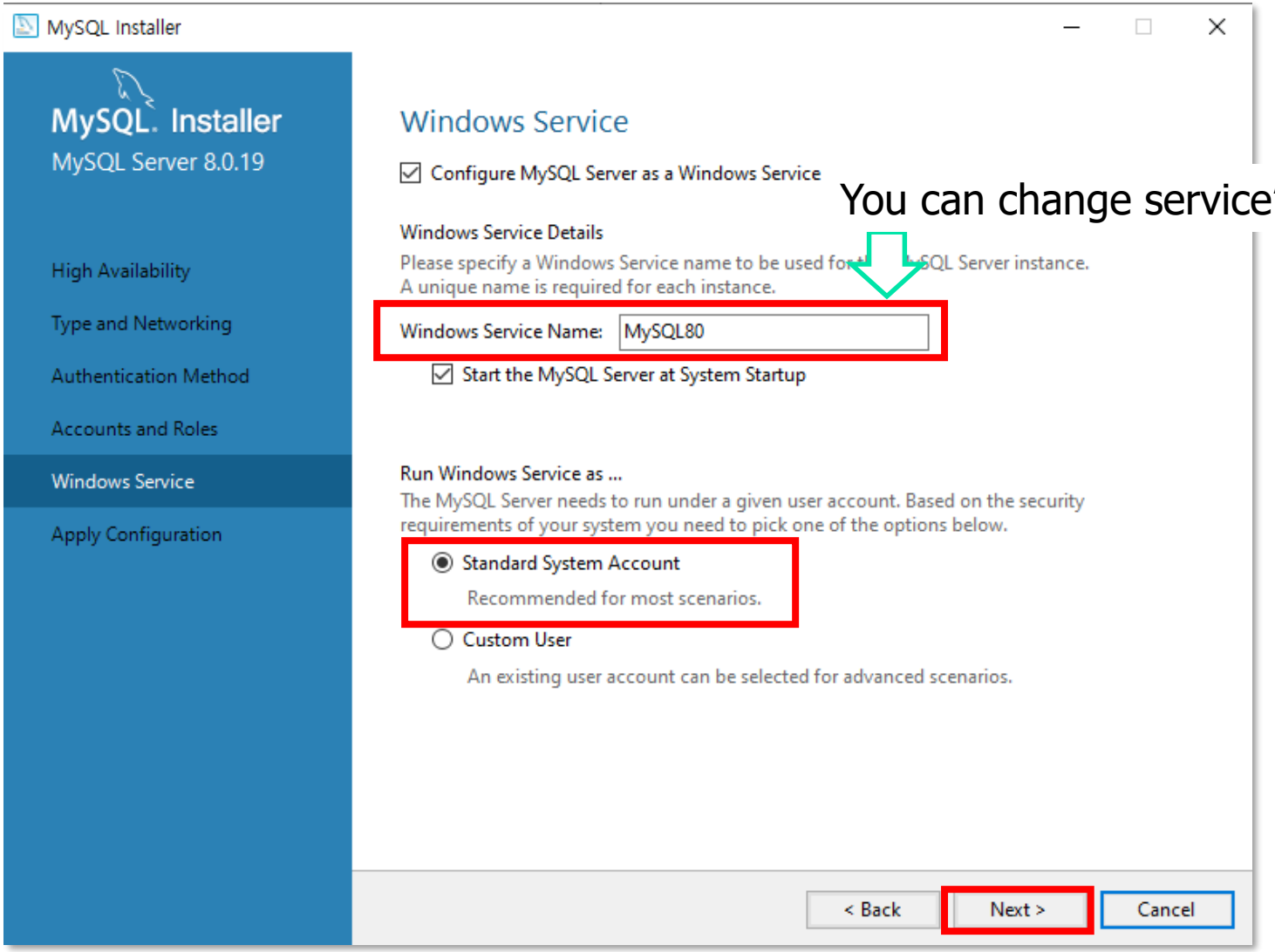


Install MySQL

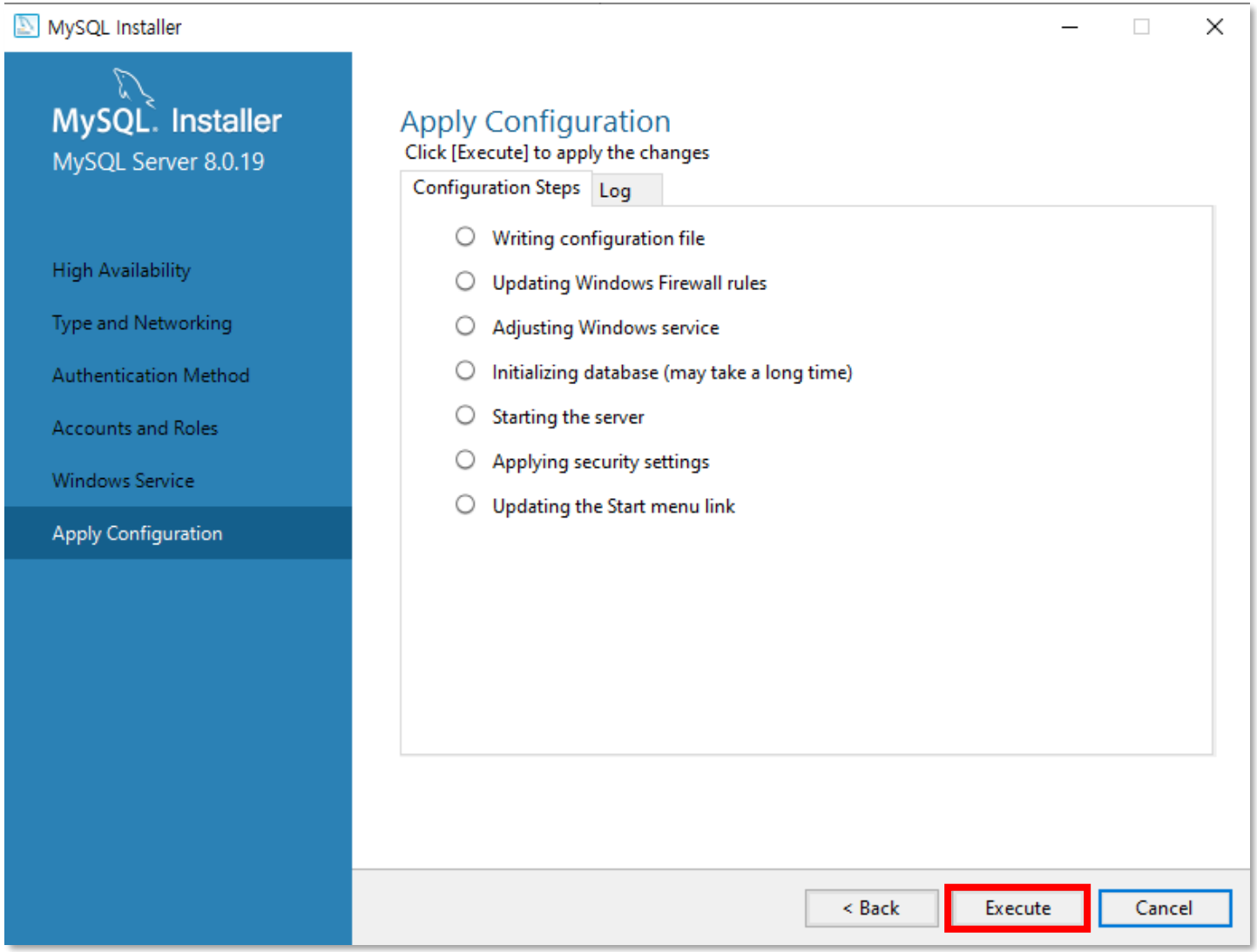
- Set password and DO NOT FORGET!



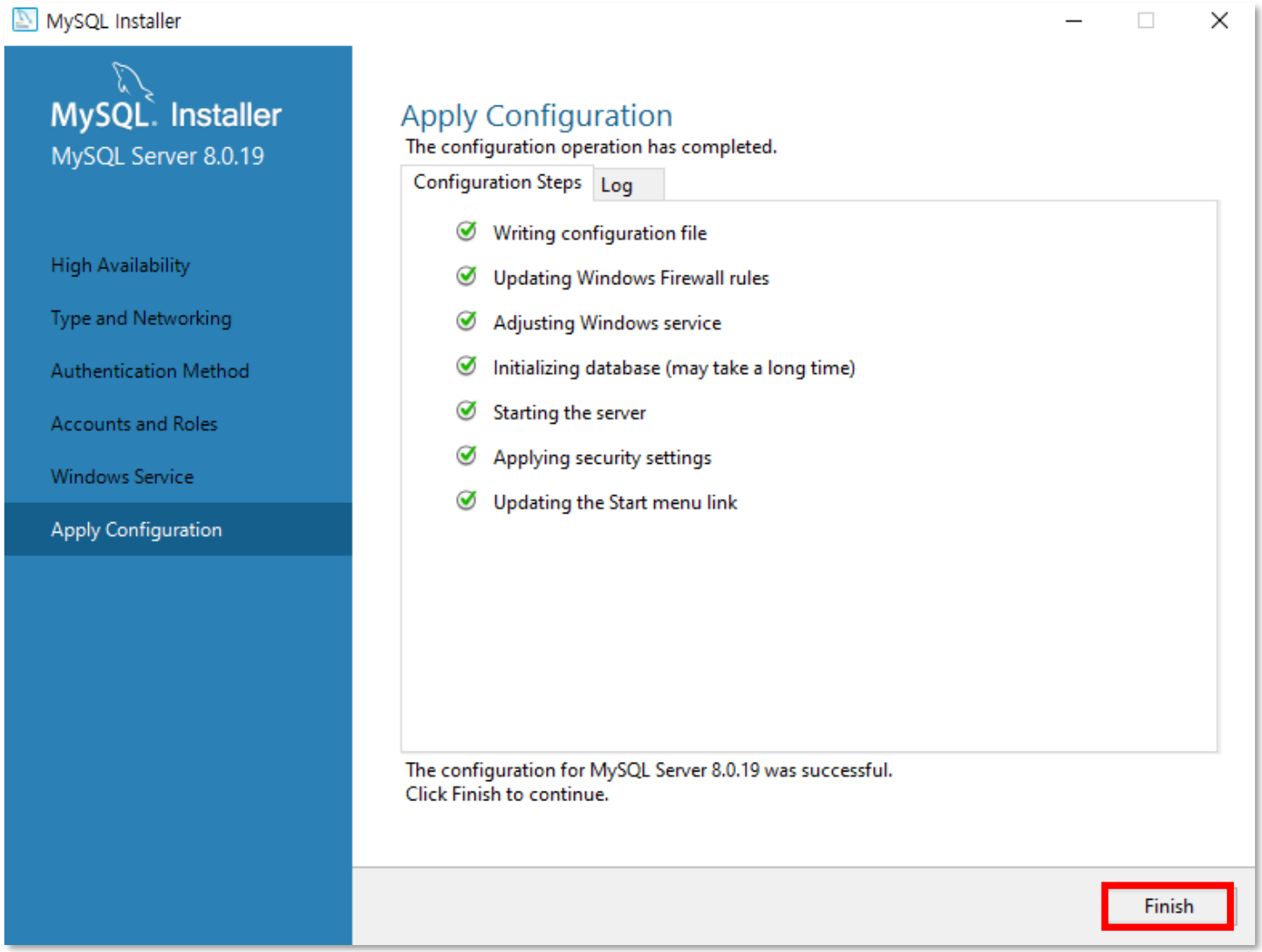
Install MySQL



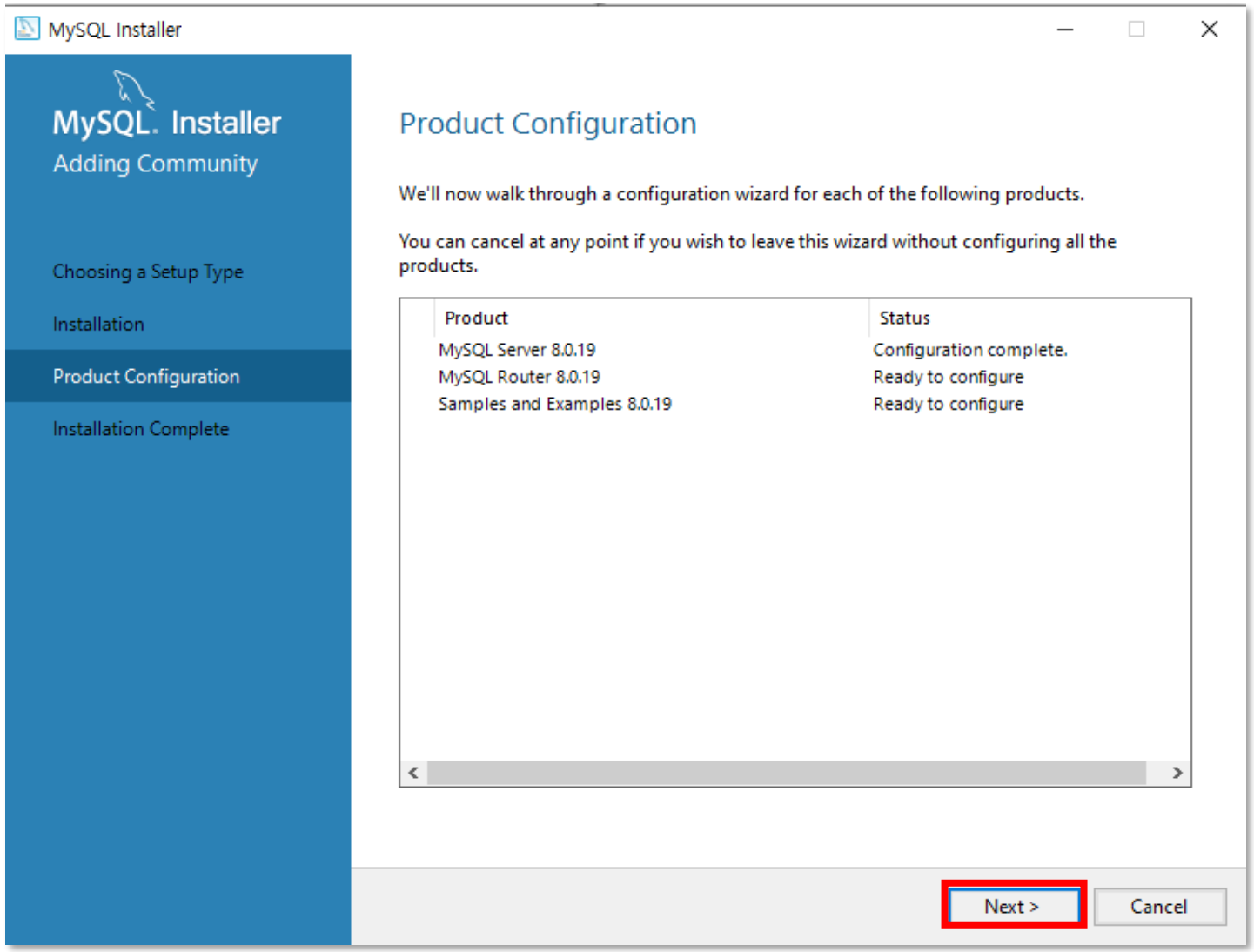
Install MySQL



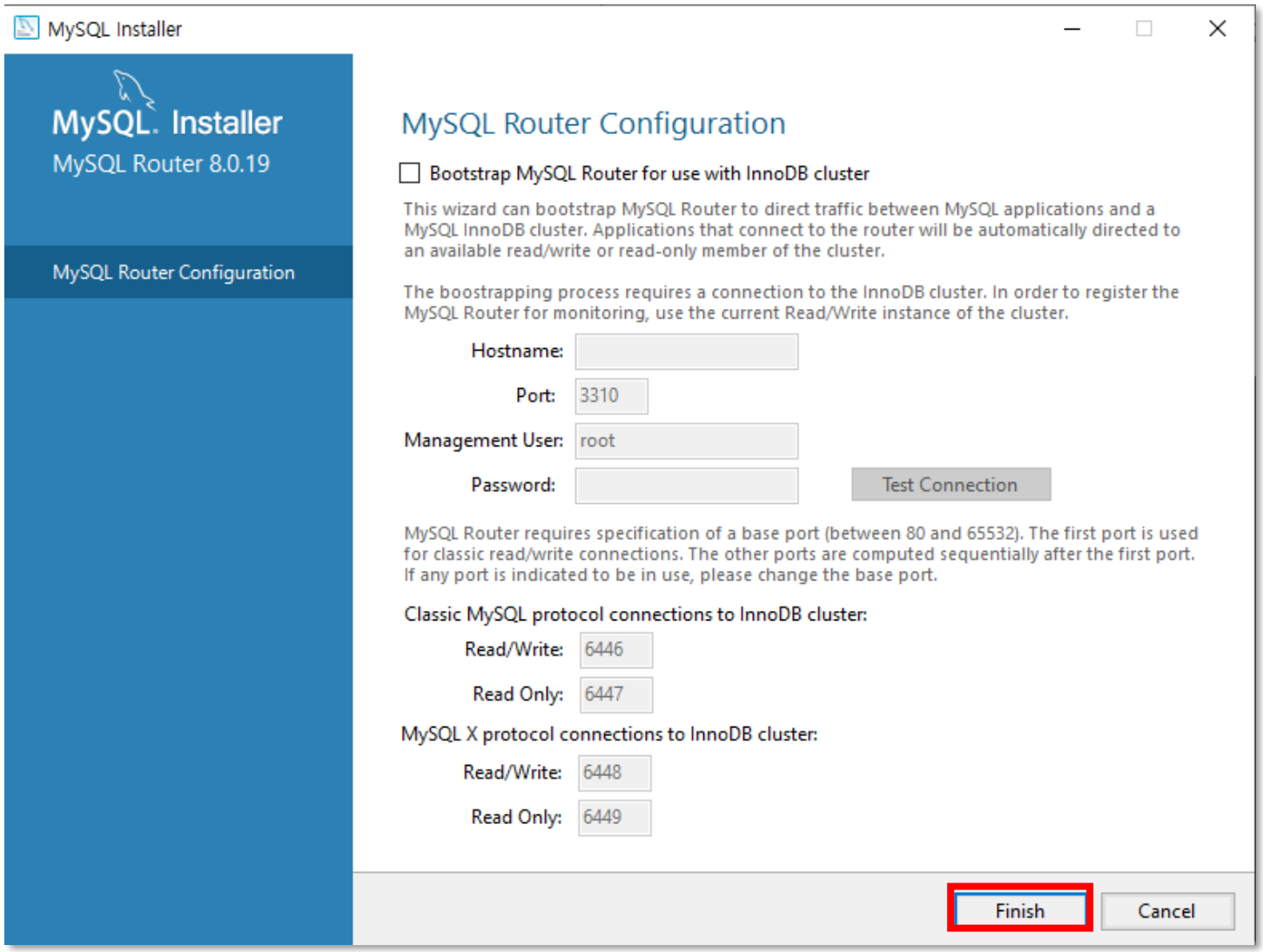
Install MySQL



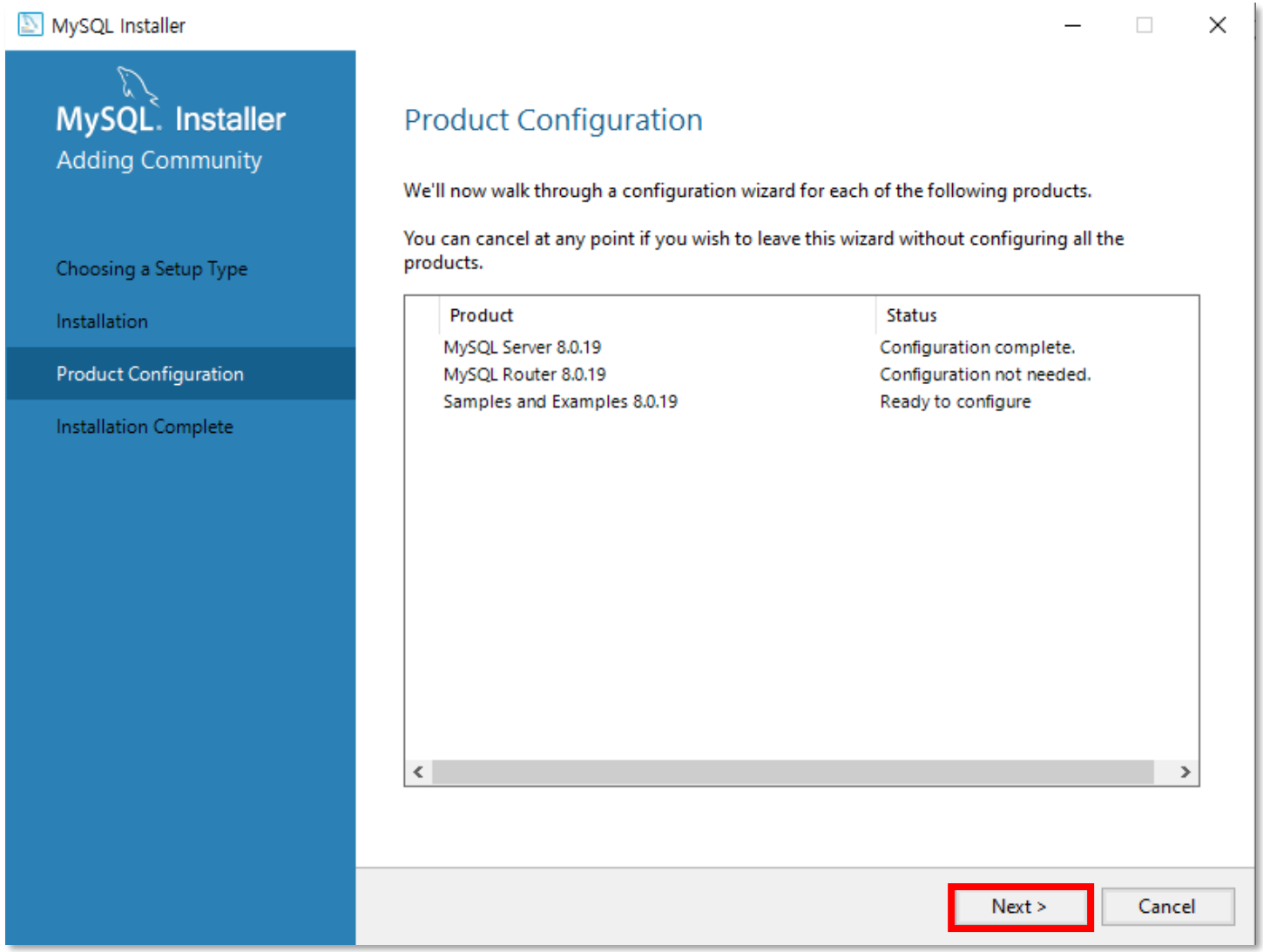
Install MySQL



Install MySQL

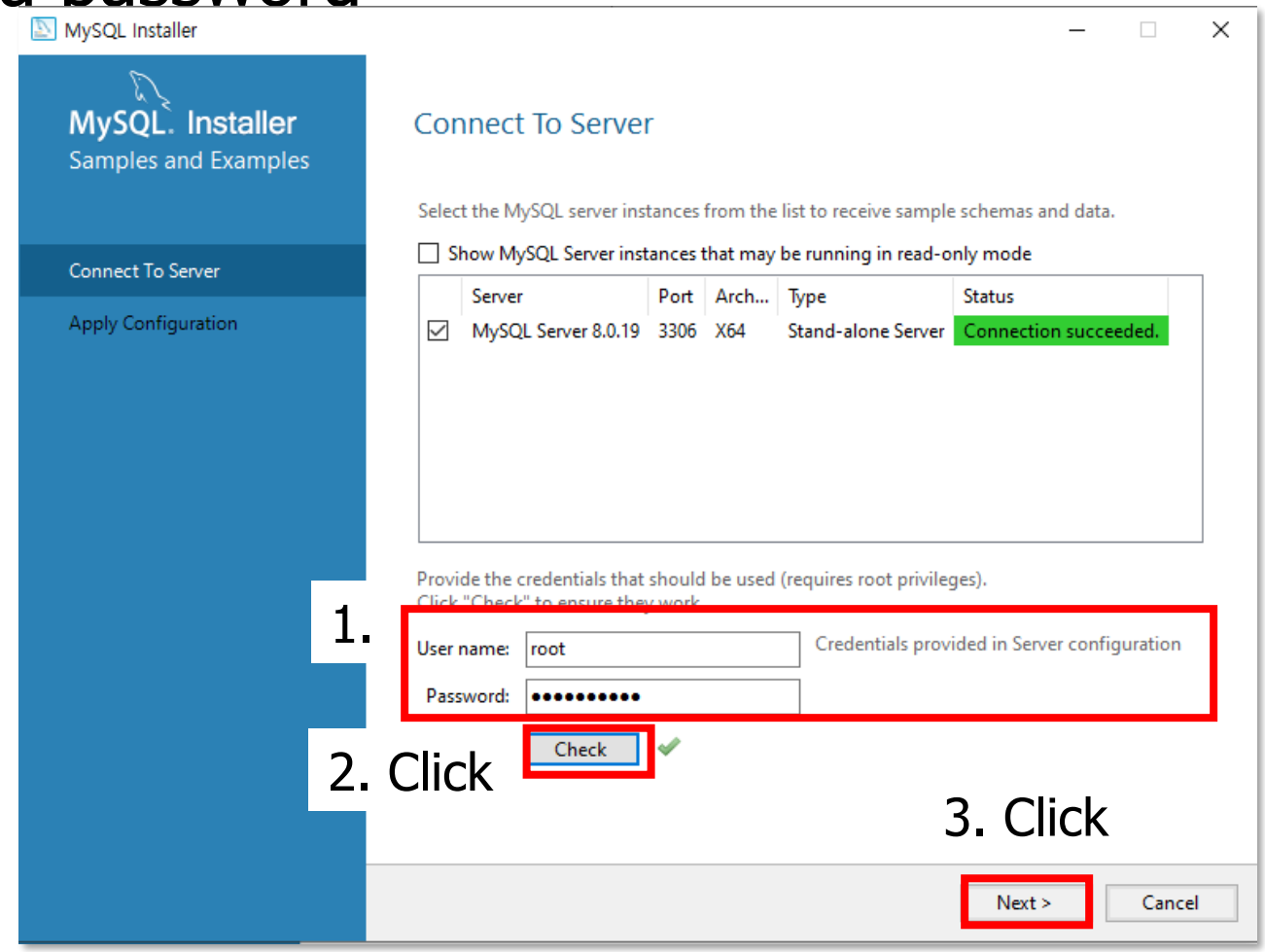


Install MySQL

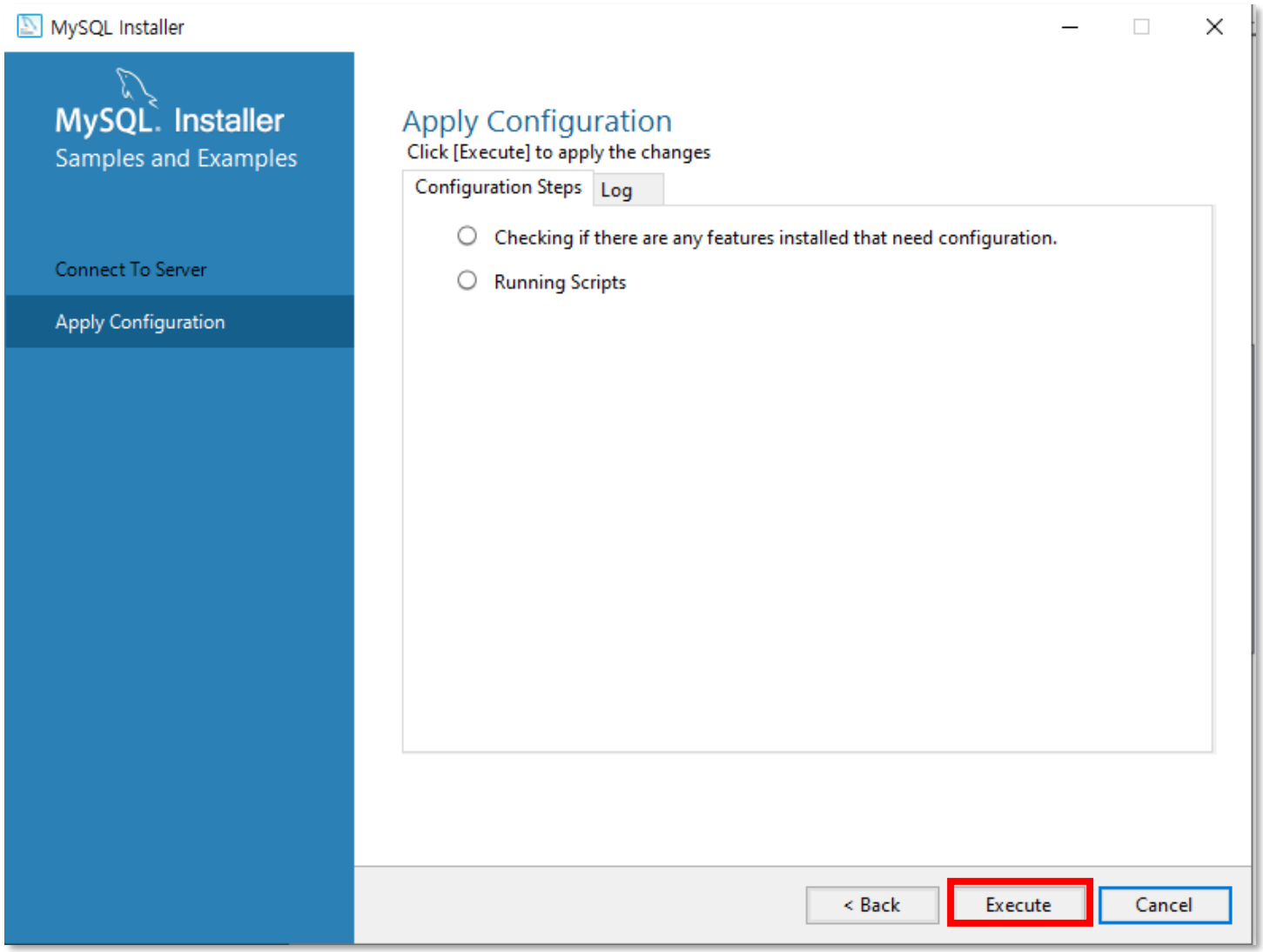


Install MySQL

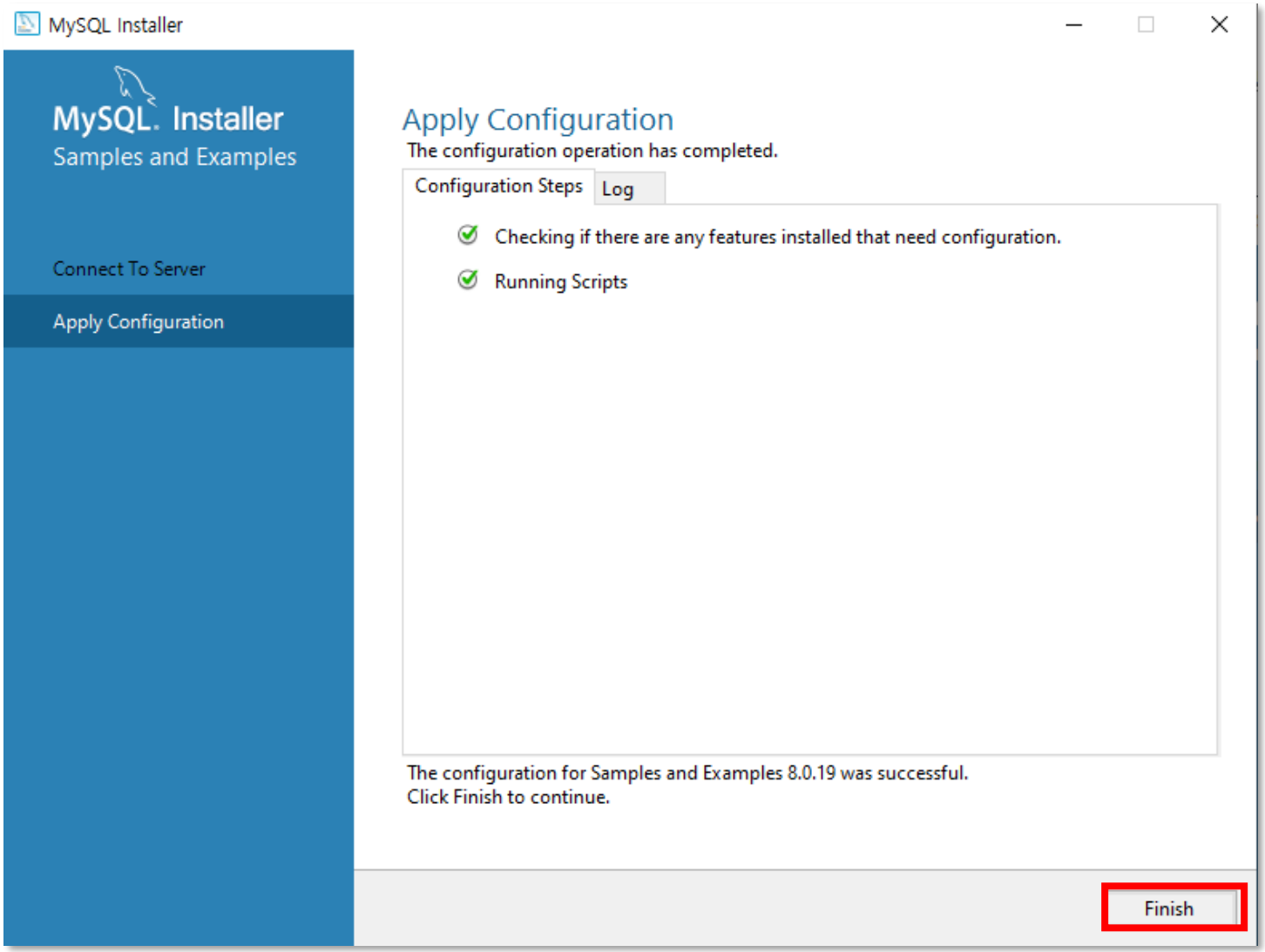
- Put a password



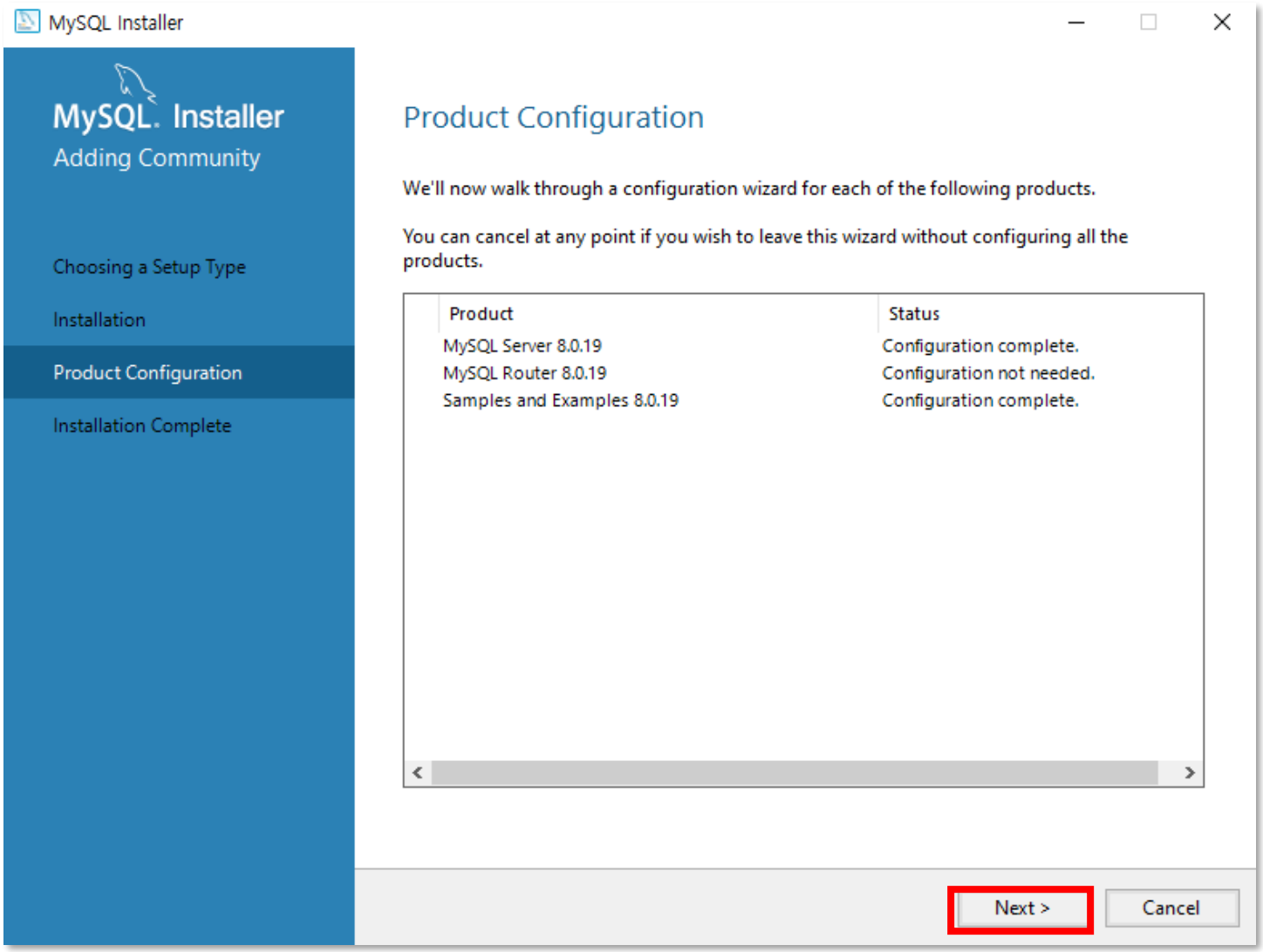
Install MySQL



Install MySQL

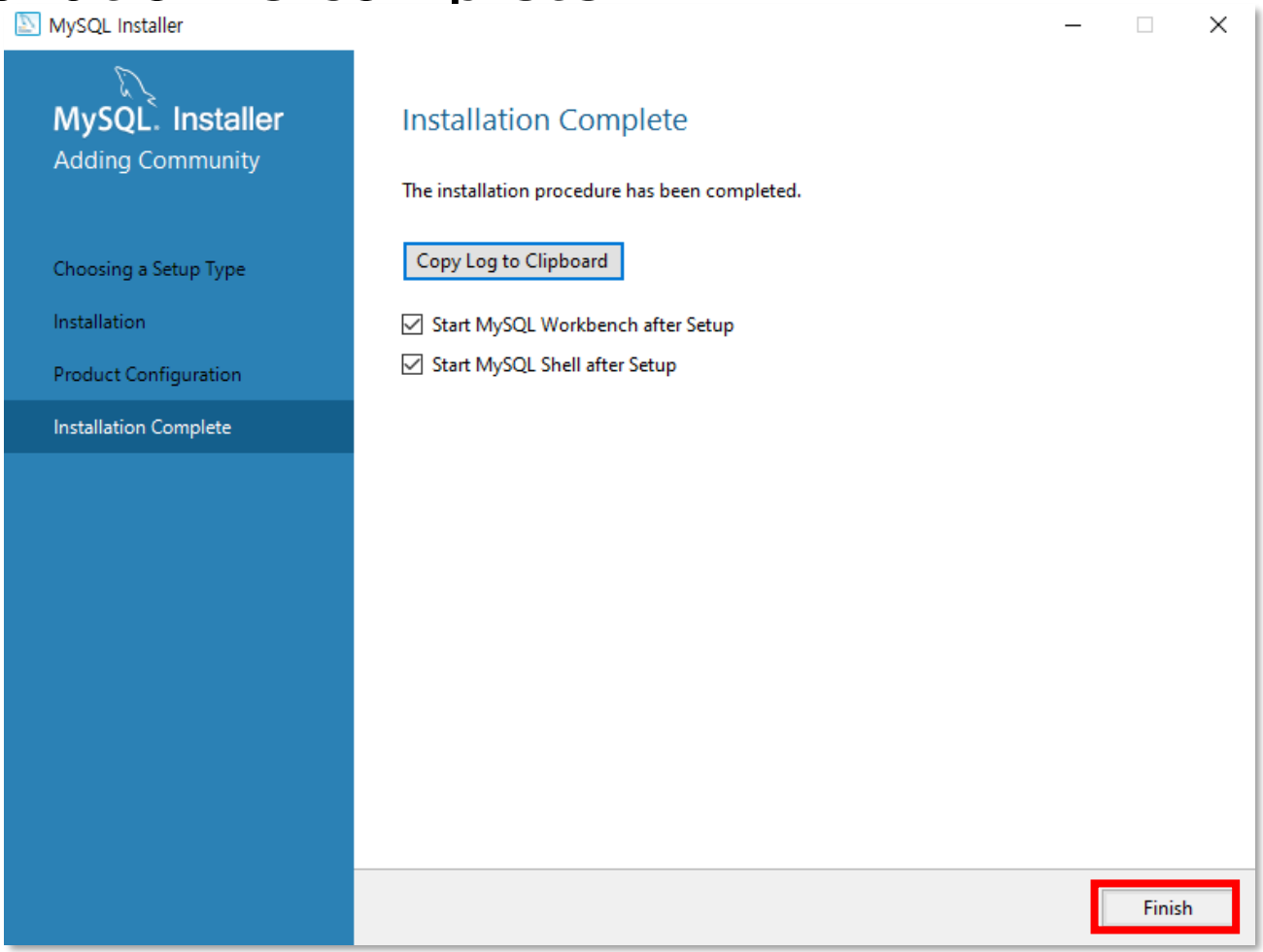


Install MySQL



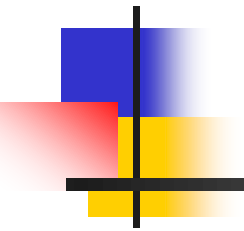
Install MySQL

- Installation is complete



References

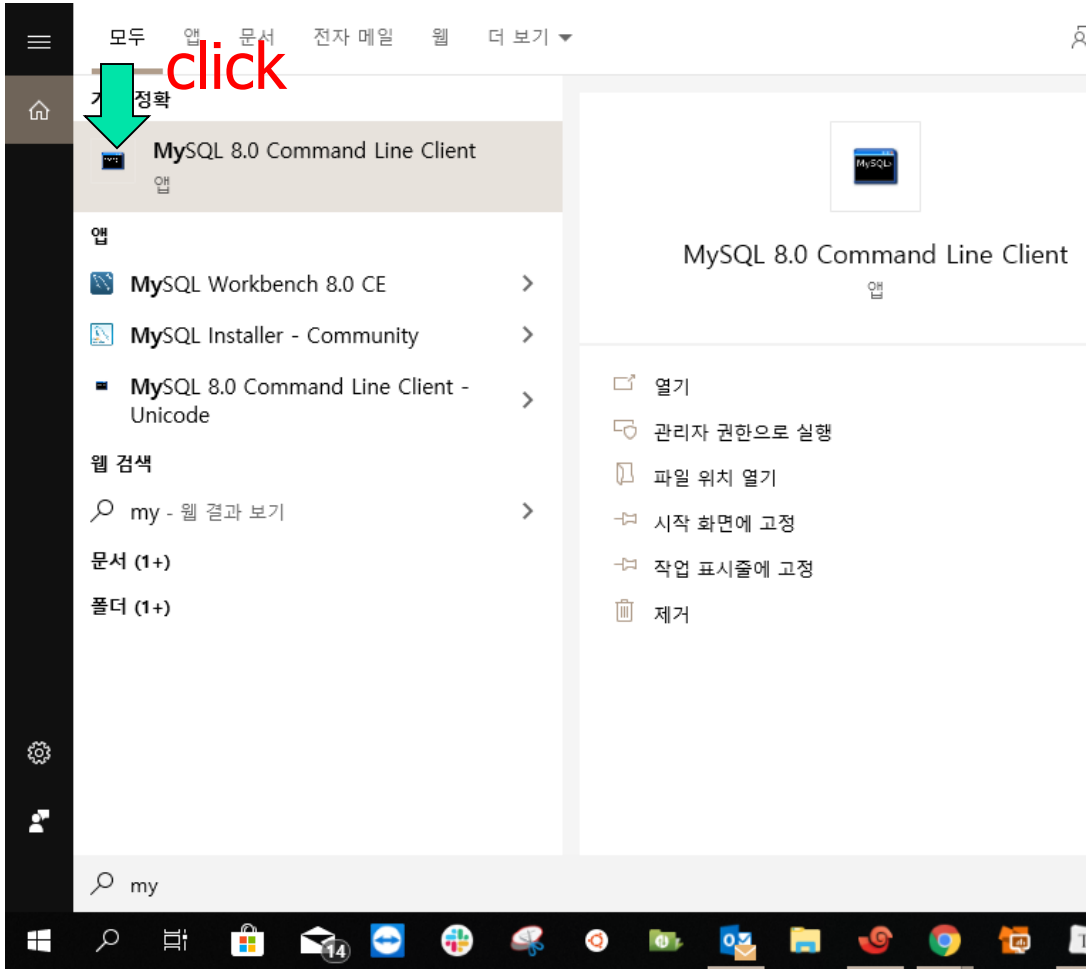
- https://www.mysql.com/



Downloading a Sample DB

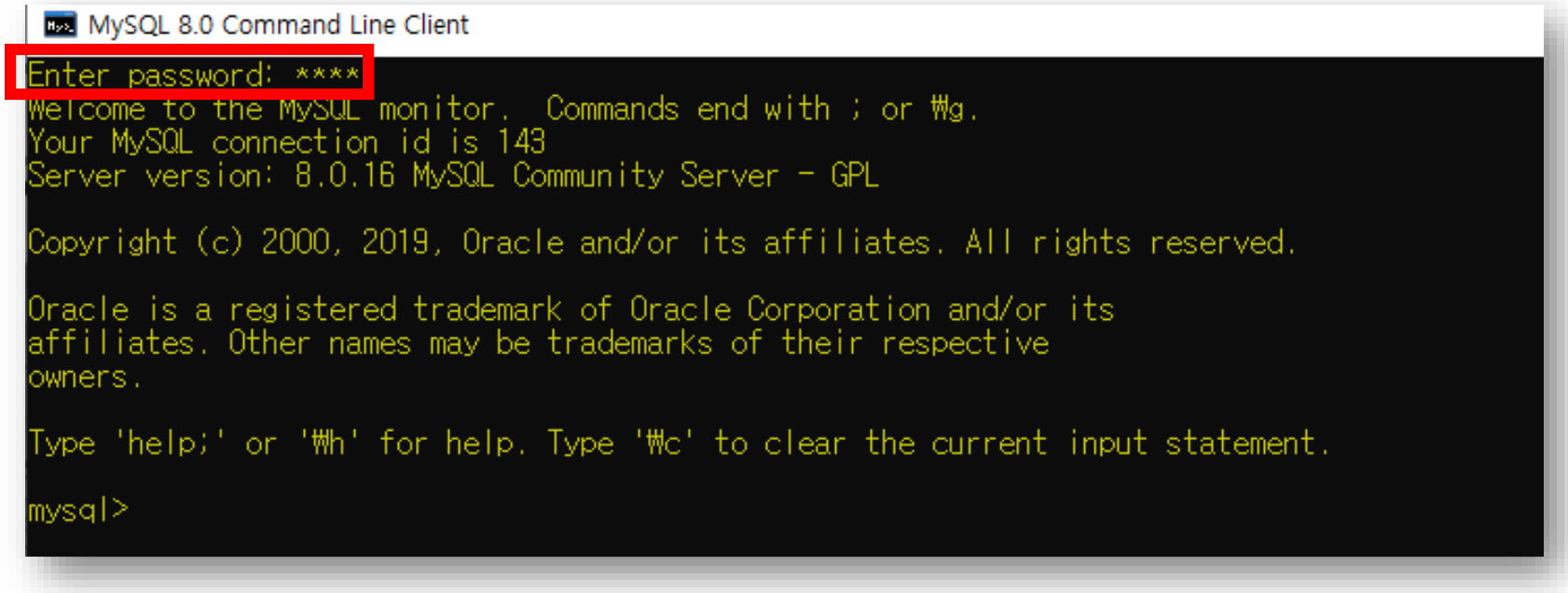
Connecting: Command Line Client

- How to use Command Line Client?



Connecting: Command Line Client

- Enter password

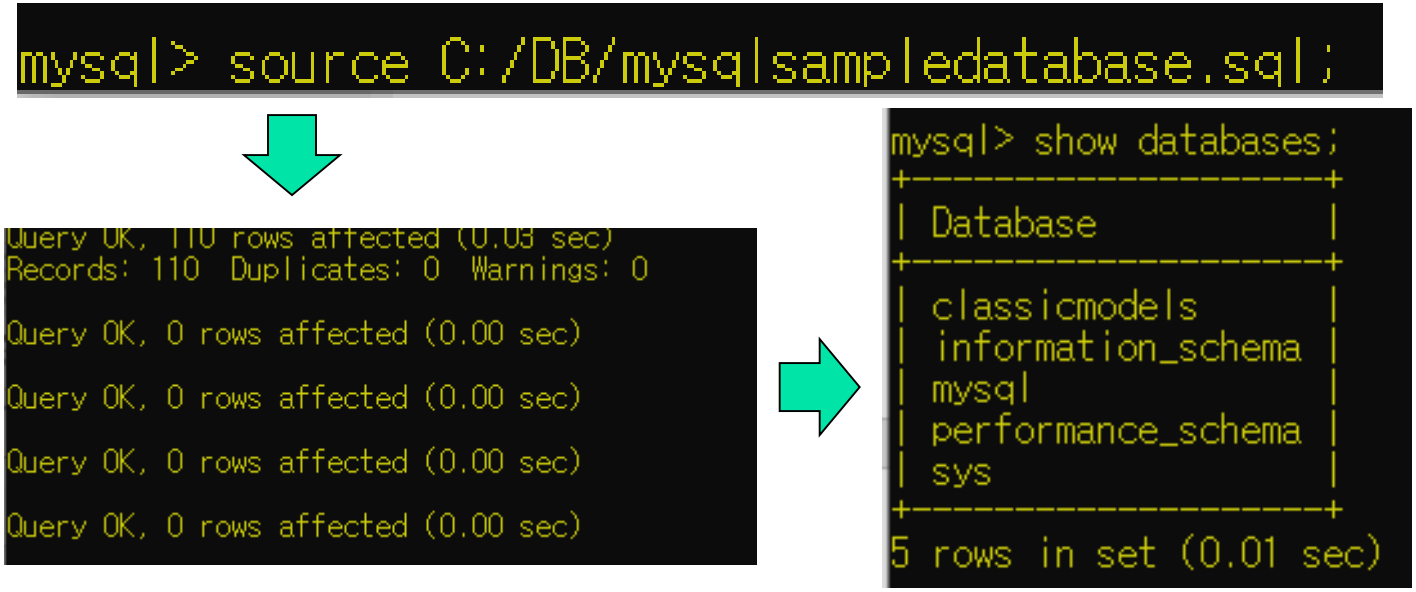


Download a Database

- Download mysqlsampledatabase.sql from http://kdd.snu.ac.kr/python/

Load Sample Database

- We will load a sample database 'classicmodels'
- In the Command Line Client
  - source <the path of the mysqlsampledatabase.sql>



Data Mining – Connect a MySQL database with Weka

Kyuseok Shim
Seoul National University
http://kdd.snu.ac.kr/~shim

Get a JDBC driver - Option 1

- Go to 'C:□Program Files (x86)□MySQL□Connector J 8.0' and copy 'mysql-connector-java-<version>.jar' to 'C:□Program Files□Weka-<version>'

Get a JDBC driver – Option 2

- Download from the web
  - https://dev.mysql.com/downloads/

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench

- MySQL Installer for Windows
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Notifier

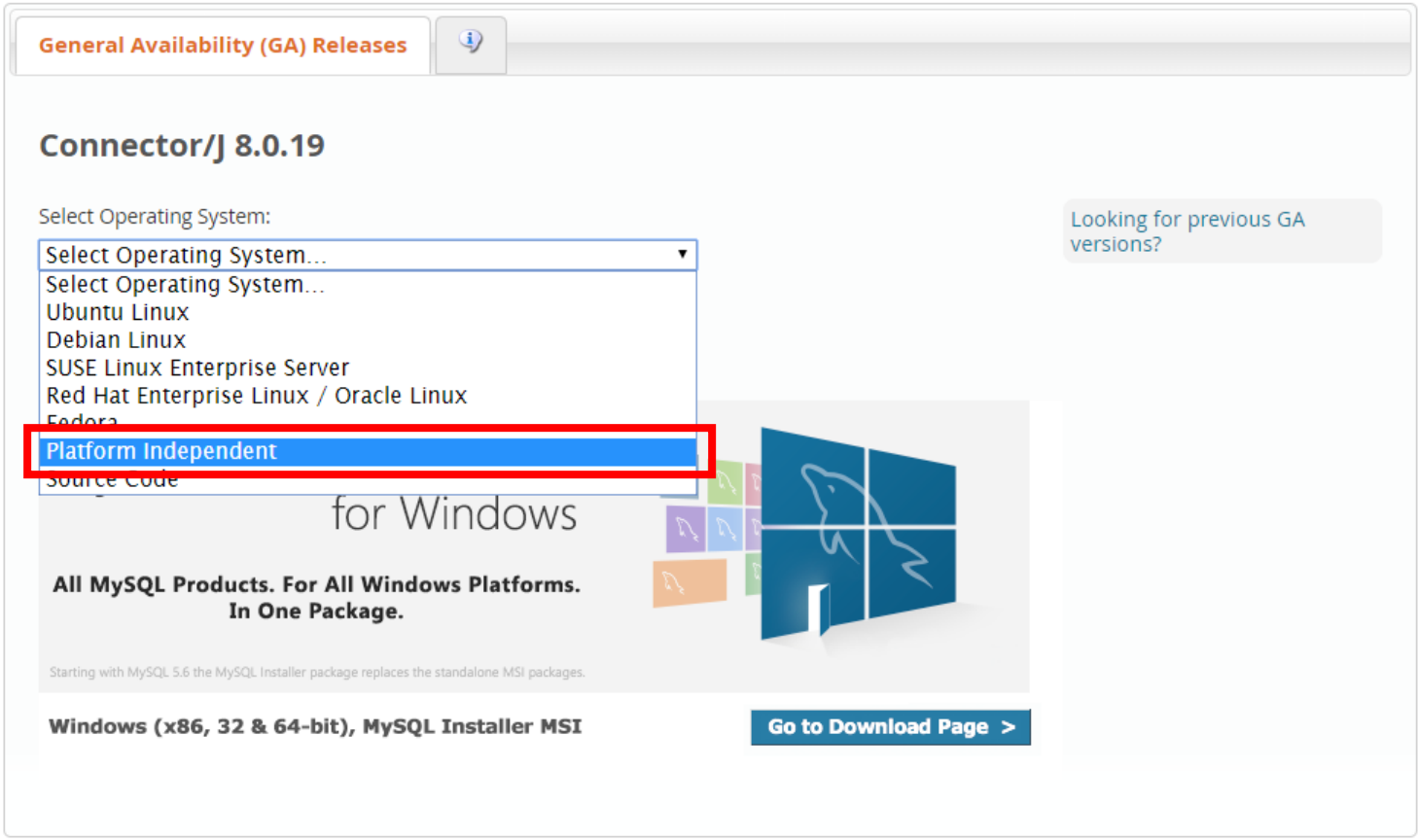
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

Get a JDBC driver – Option 2

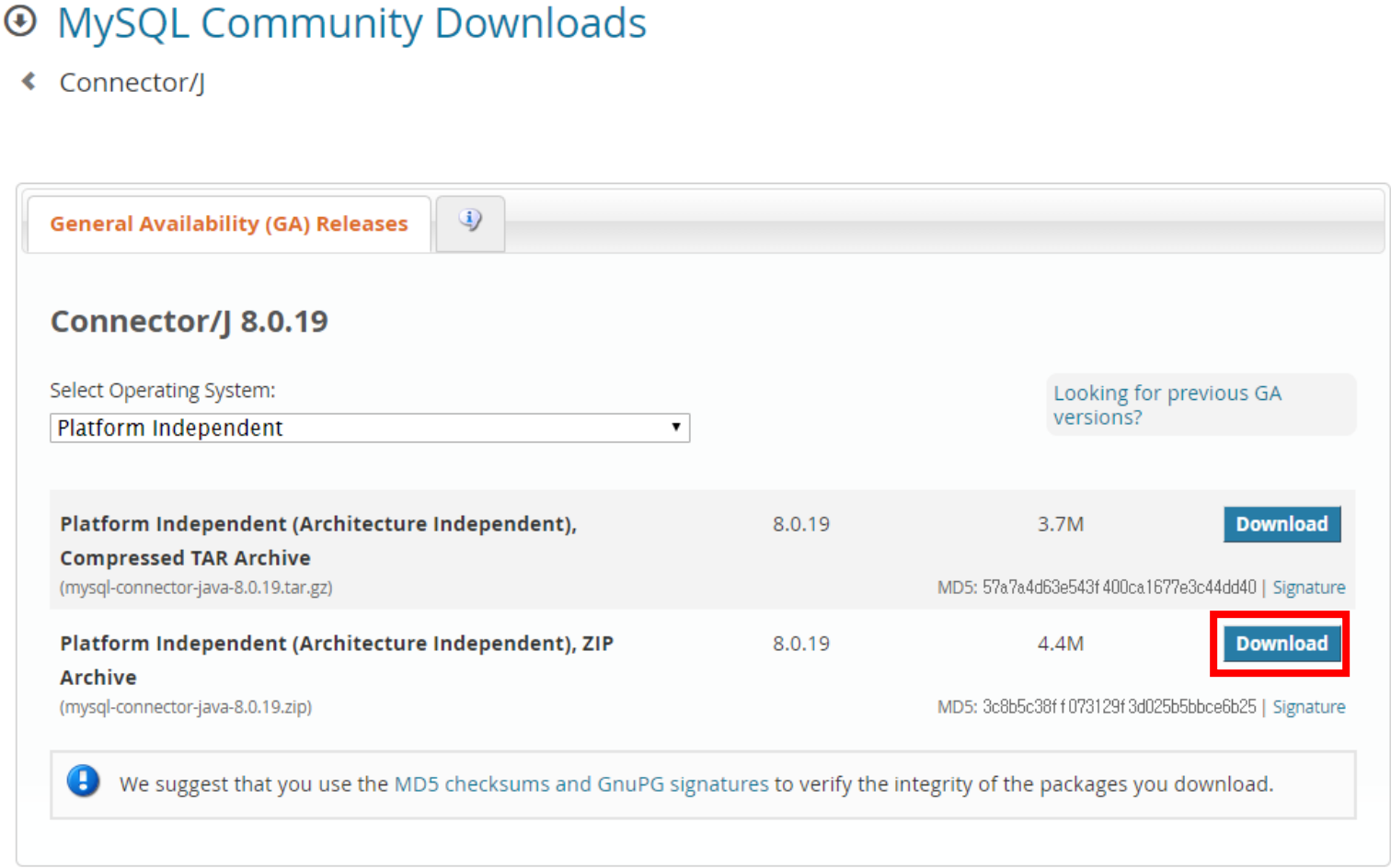
MySQL Community Downloads

Connector/J



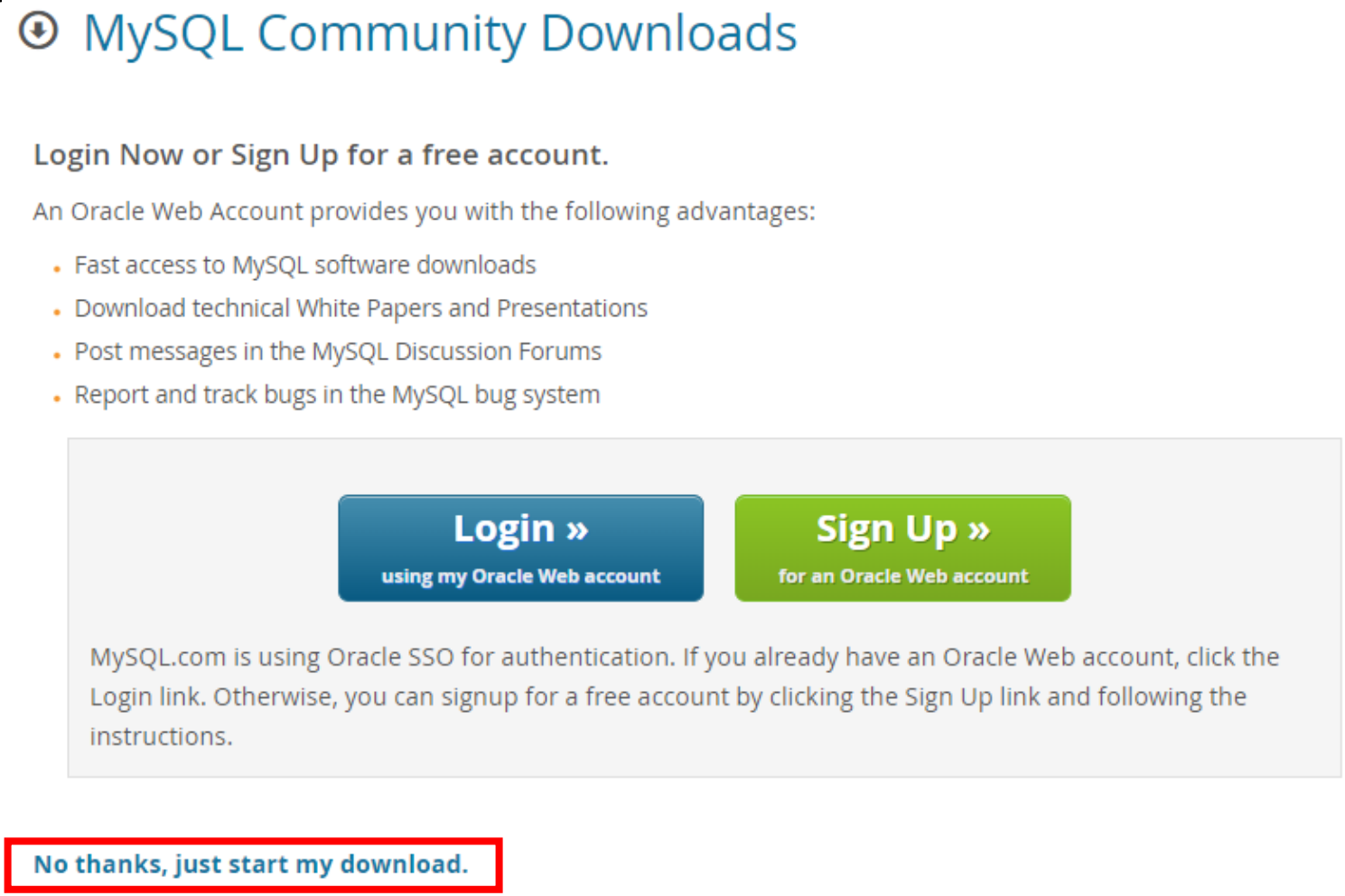
Get a JDBC driver – Option 2

- Download zip file



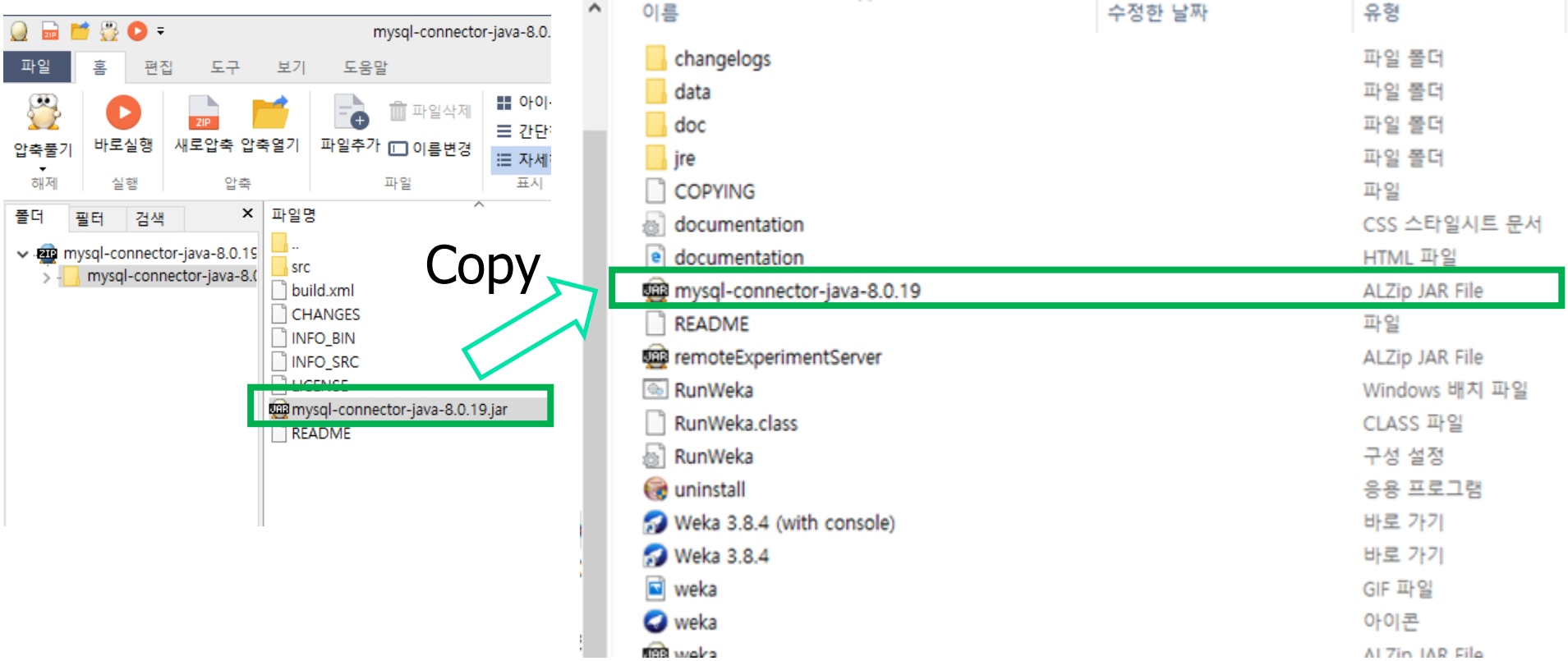
Get a JDBC driver – Option 2

- Download zip file



Get a JDBC driver – Option 2

- Unzip downloaded file and copy ‘mysql-connector-java-<version>.jar’ to ‘C:□Program Files□Weka-<version>’

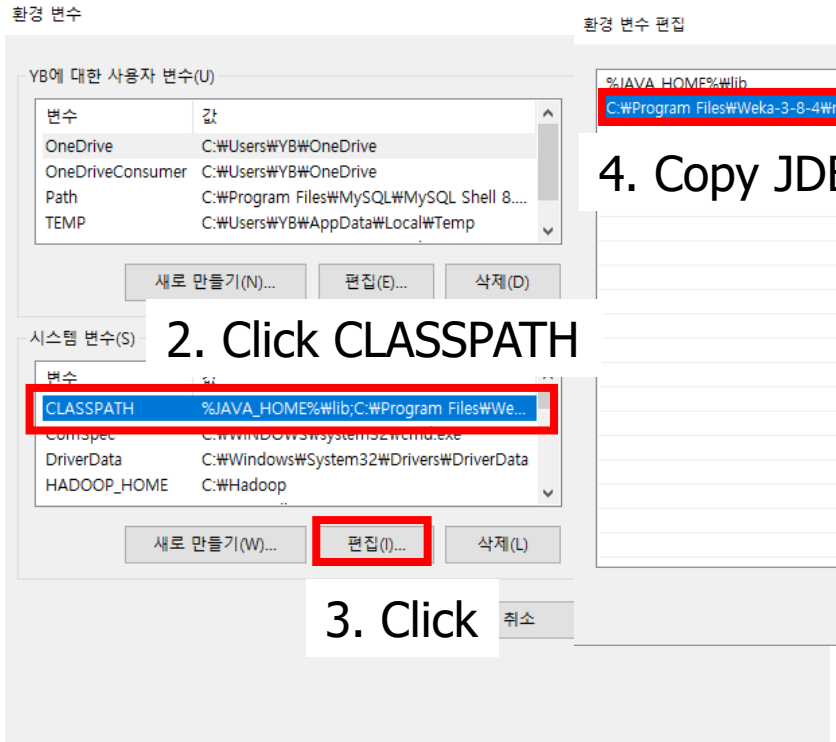


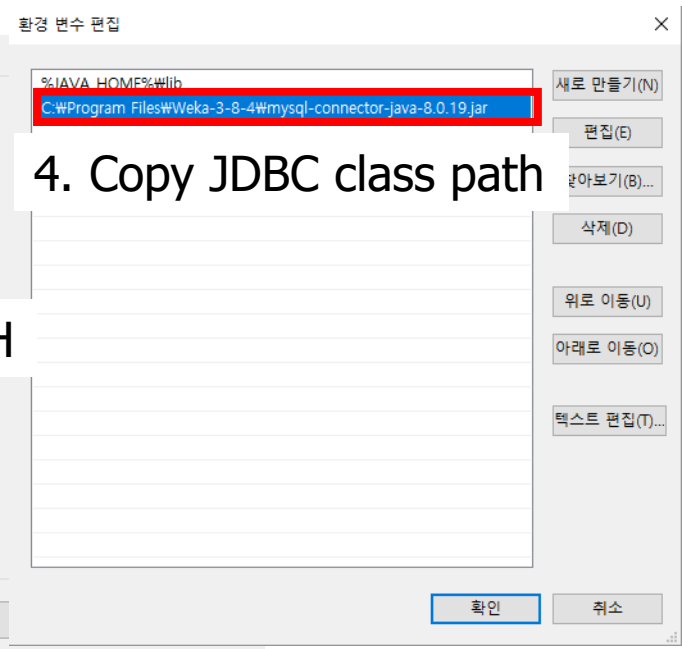
Set environmental variable

- Go to 'Control Panel→System and Security→System'

Set environmental variable

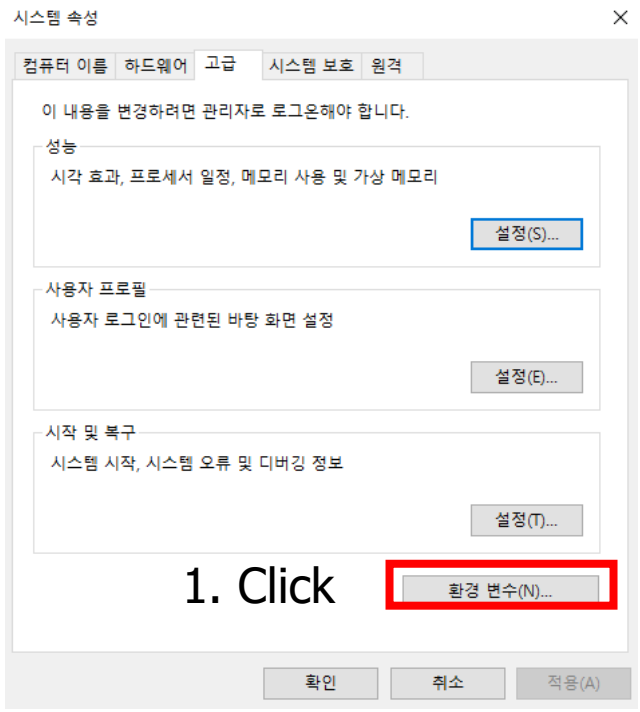
- If CLASSPATH already exists in the system variable, copy JDBC class path to CLASSPATH
  - JDBC class path is the path where you place 'mysql-connector-java-<version>.jar' file
    - E.g., 'C:□Program Files□Weka-3-8-4□mysql-connector-java-8.0.19.jar'

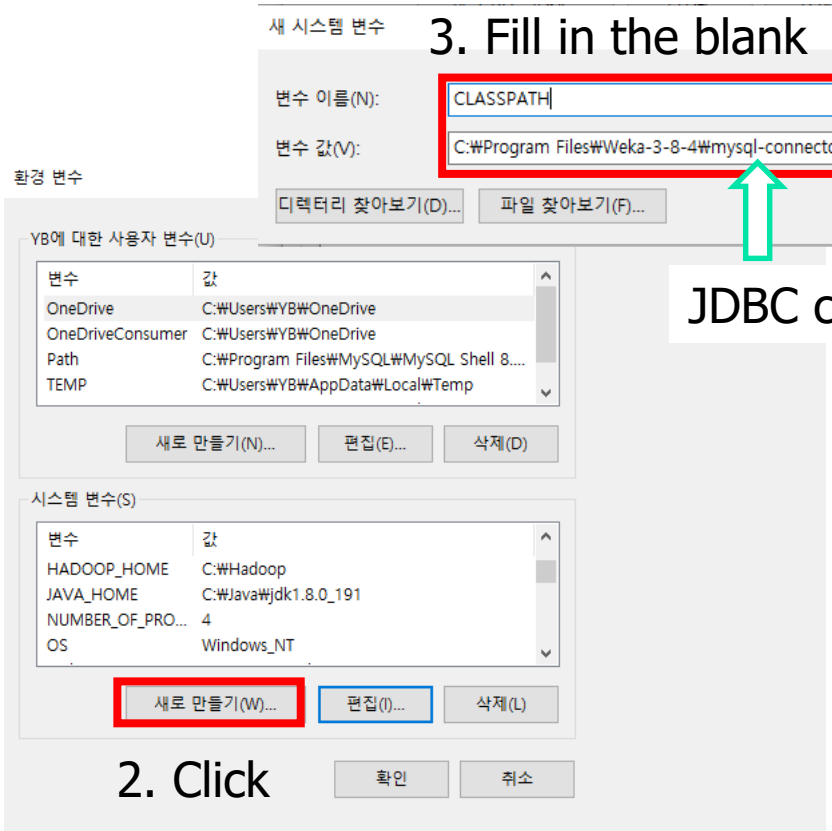




Set environmental variable

- If not, create CLASSPATH and fill in the blank
  - JDBC class path is the path where you place 'mysql-connector-java-<version>.jar' file
    - E.g., 'C:□Program Files□Weka-3-8-4□mysql-connector-java-8.0.19.jar'

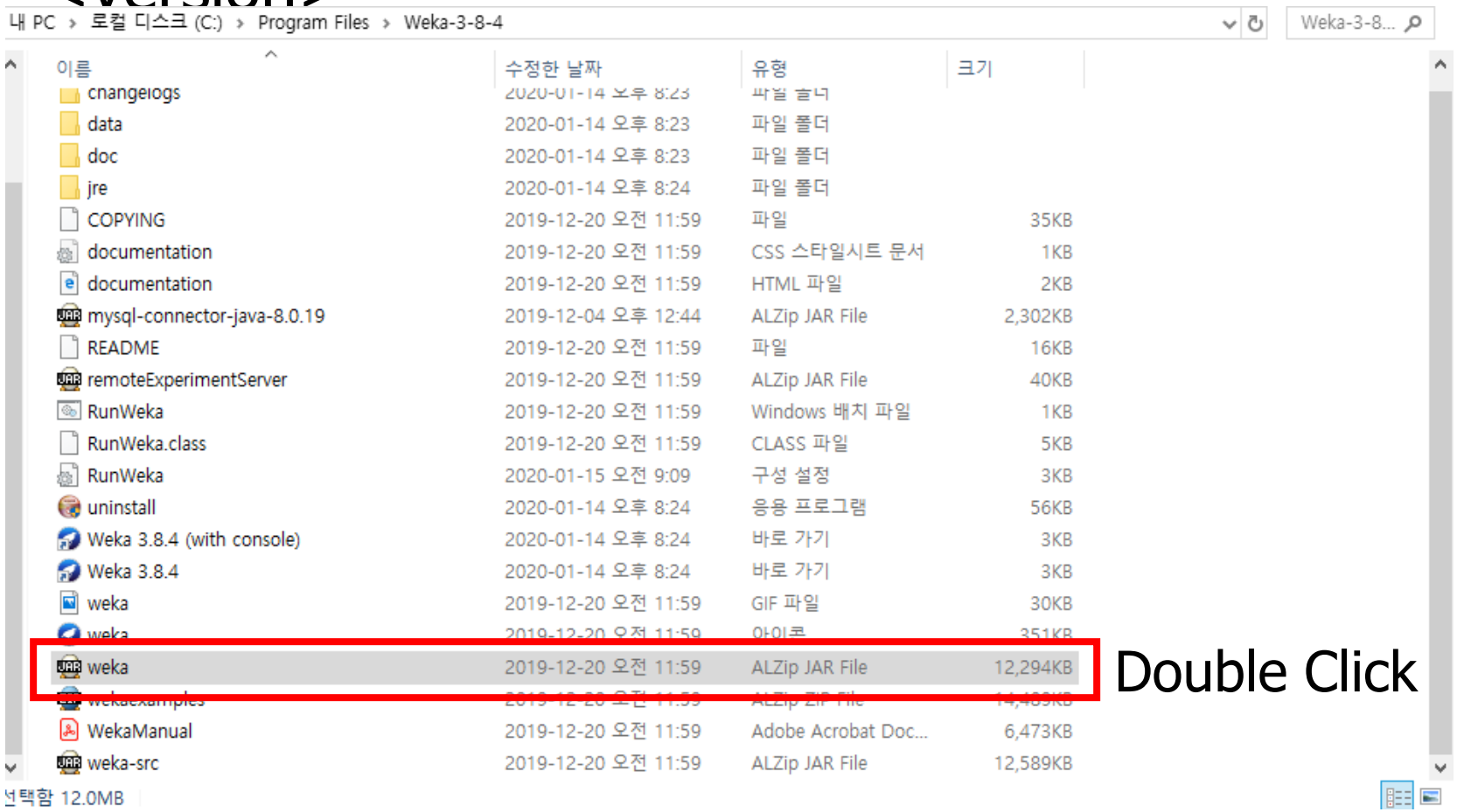




JDBC class path

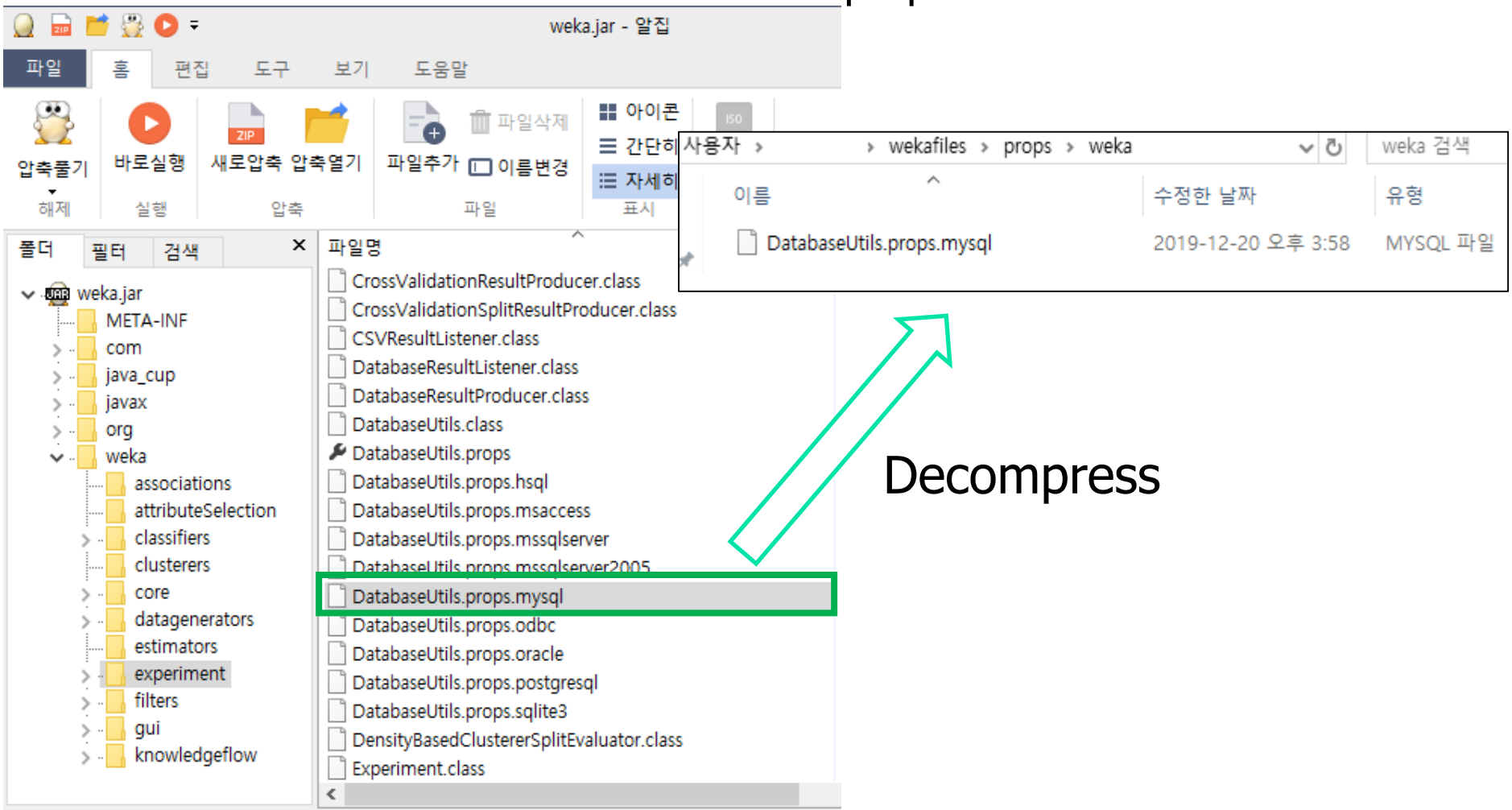
Set DatabaseUtils.props

- Open weka.jar in 'C:□Program Files□Weka-<version>'



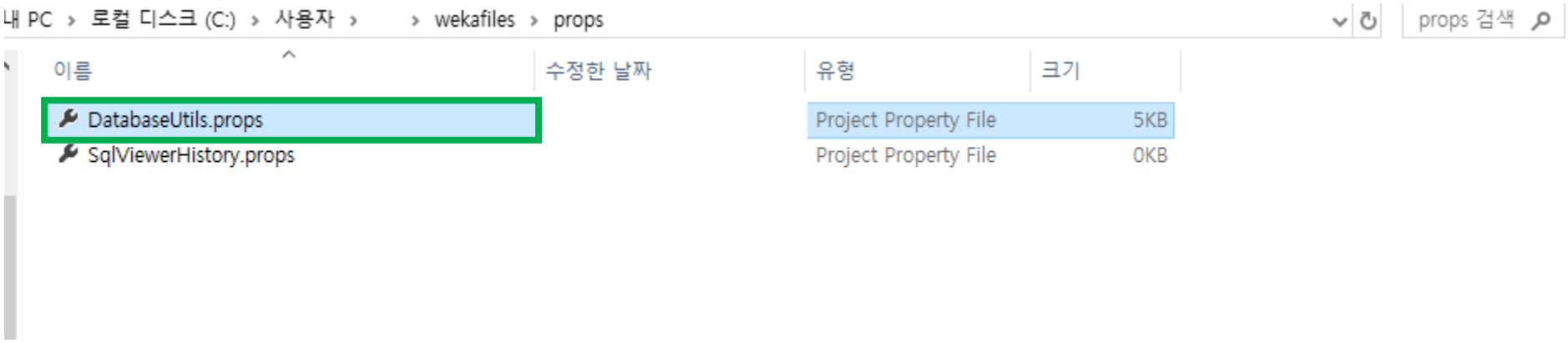
Set DatabaseUtils.props

- Decompress 'DatabaseUtils.props.mysql' file in 'weka□experiment' to 'C:□Users□<username>□wekafiles□props'



Set DatabaseUtils.props

- Rename ‘DatabaseUtils.props.mysql’ to ‘DatabaseUtils.props’



Set DatabaseUtils.props

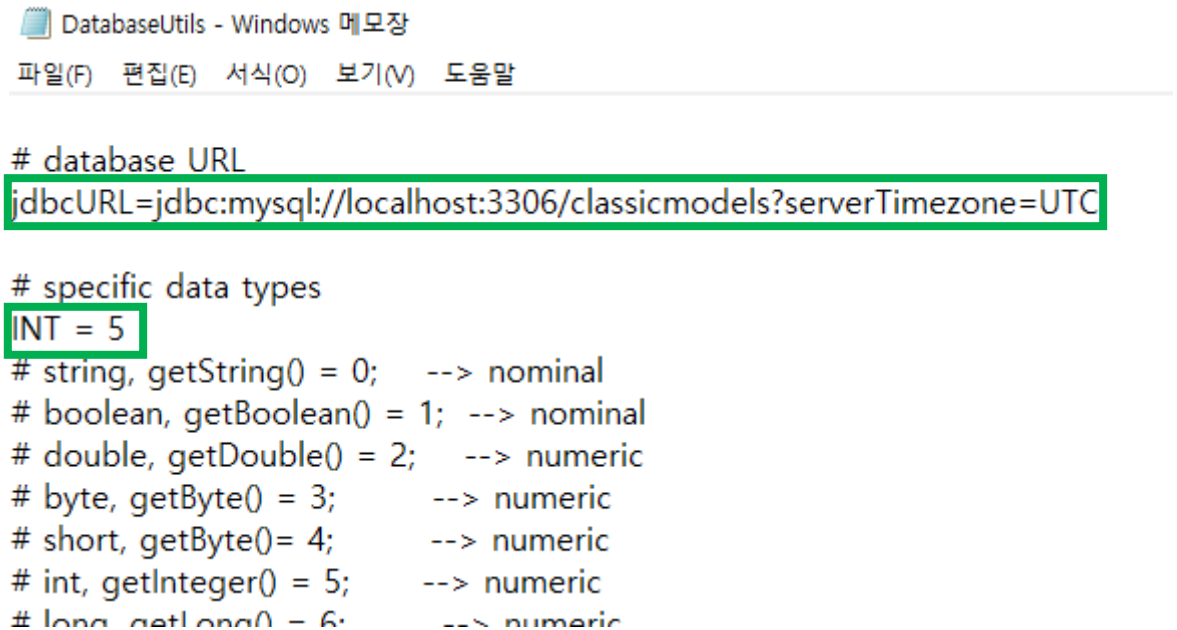
- Open 'DatabaseUtils.props' and change below
  - # database URLjdbcURL=jdbc:mysql://server_name:3306/database_name?server_timezone
    - E.g., jdbc:mysql://localhost:3306/classicmodels?serverTimezone=UTC
  - Type 'INT=5' under '# specific data types'

Update port if you changed

MySQL's server timezone

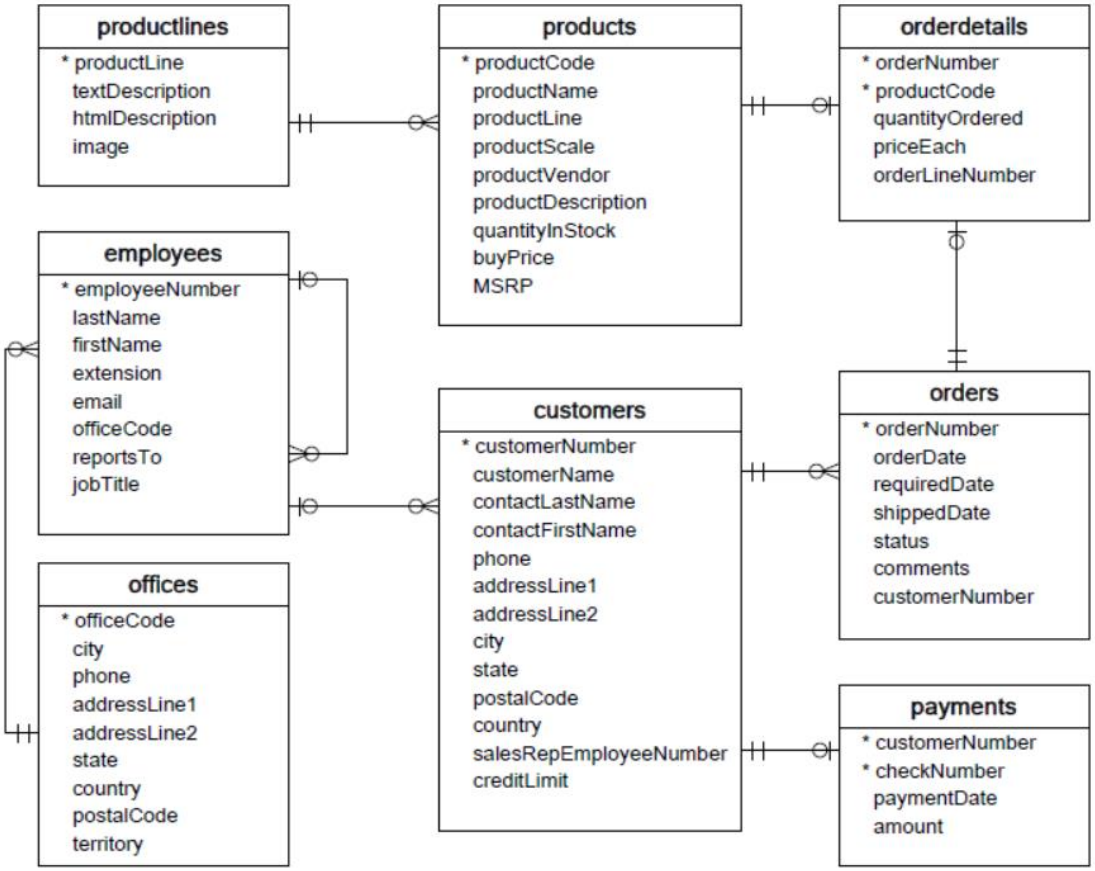
Database name you want to connect

Coordinated Universal Time



Database

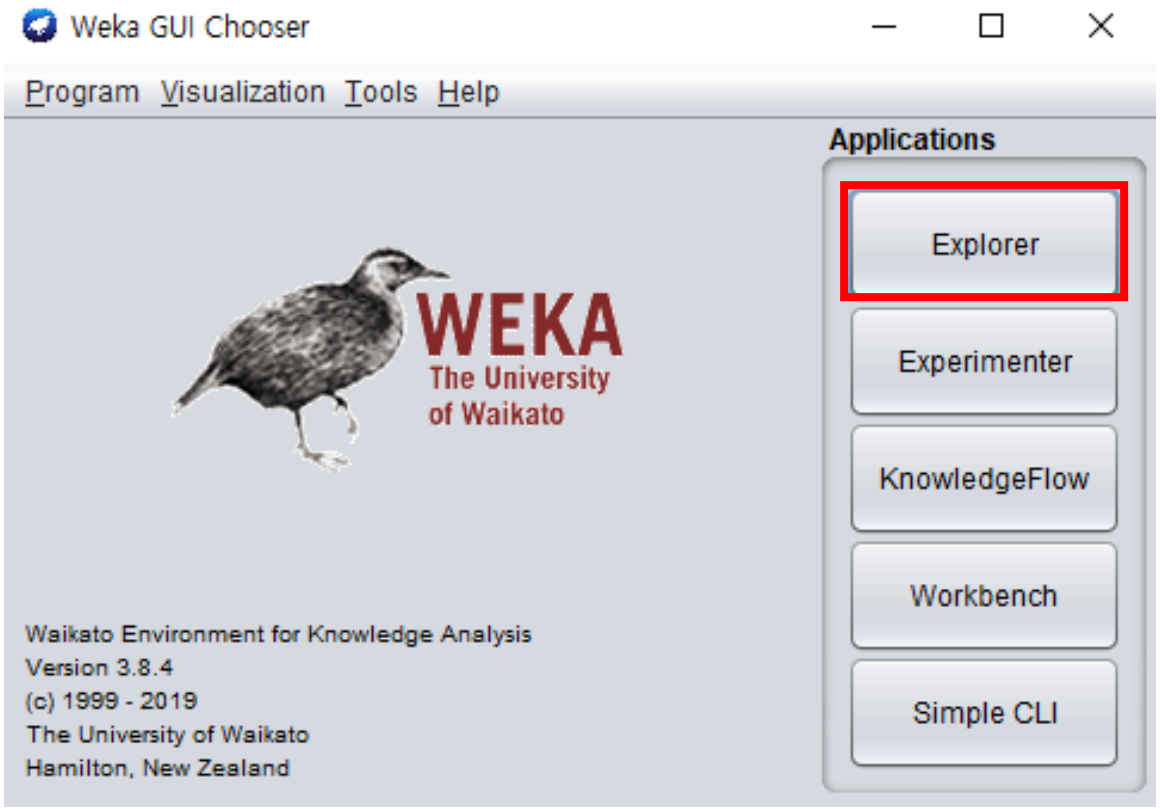
- Downloaded database's schema
  - Source: https://www.mysqltutorial.org/mysql-sample-database.aspx



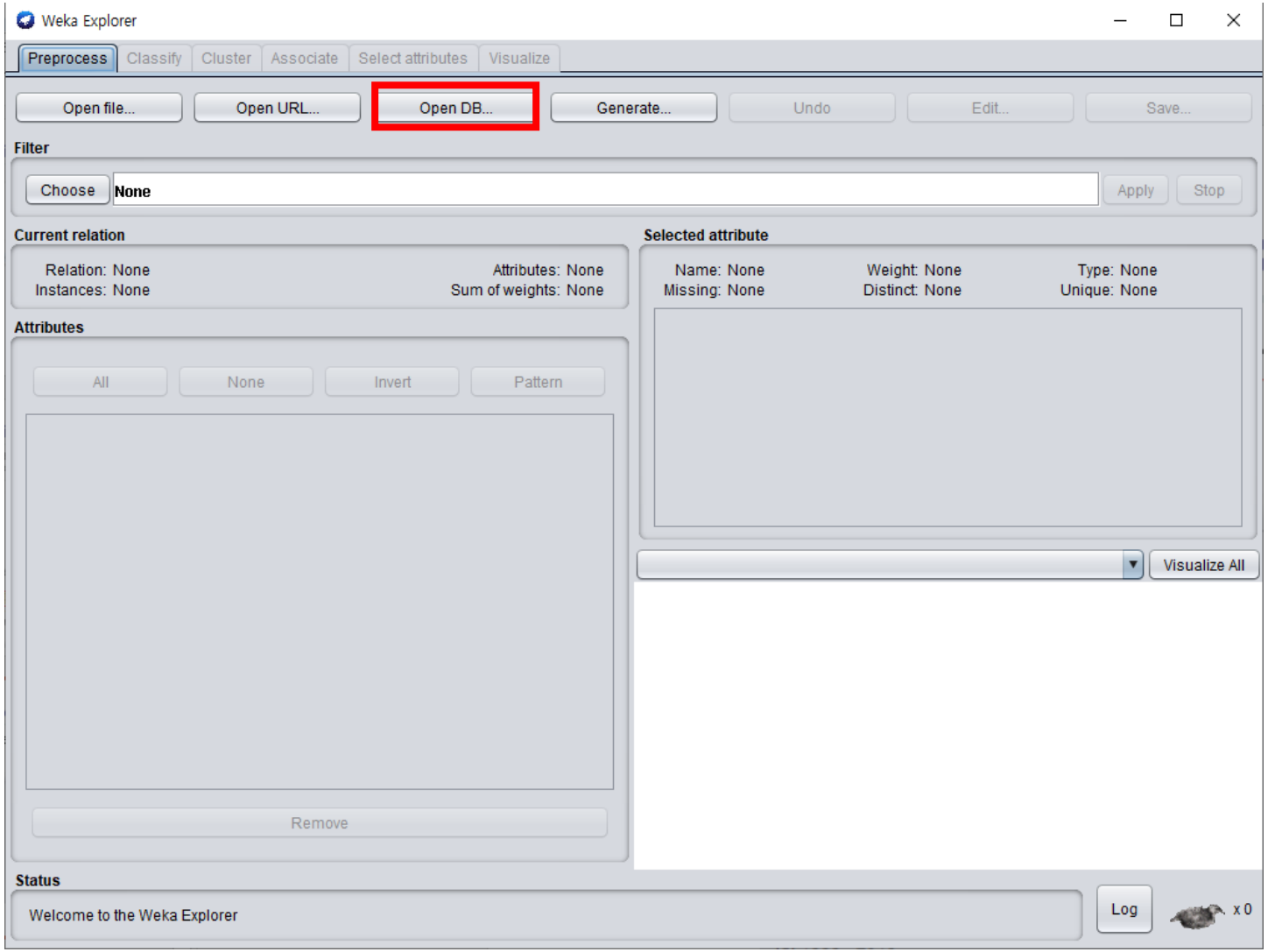
DB: classicmodels

Connet Weka with MySQL

- Open Weka and select 'Explorer'

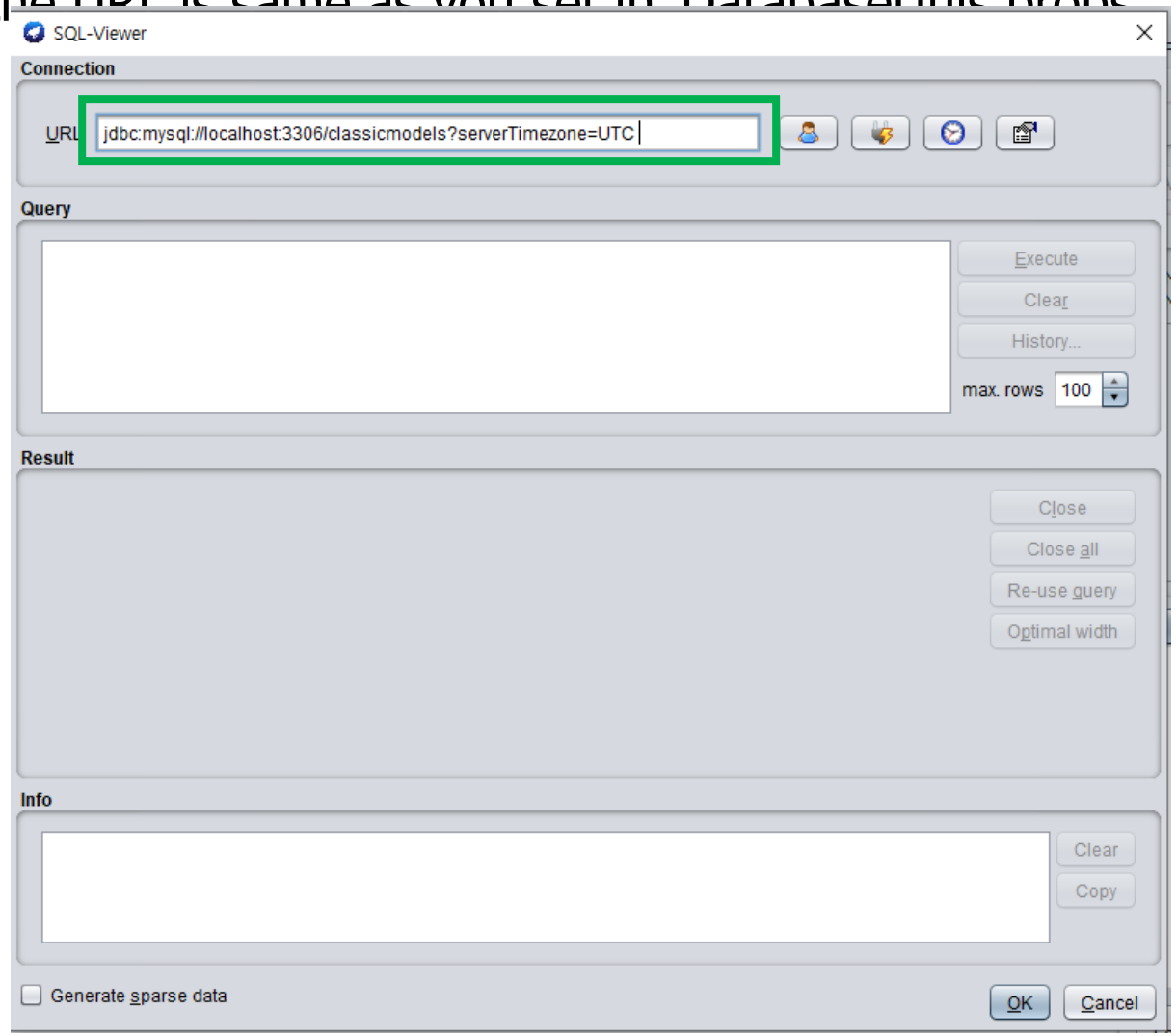


Connet Weka with MySQL



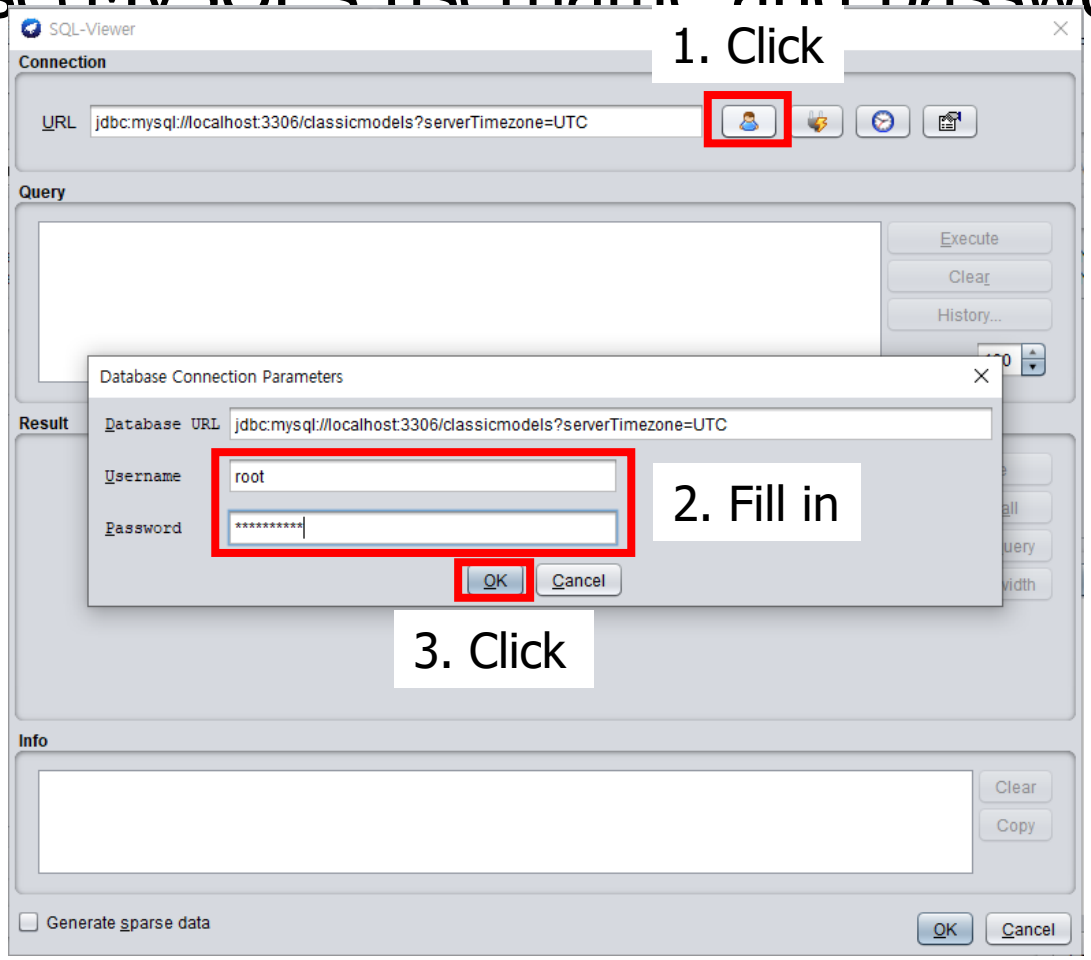
Connet Weka with MySQL

- Check the URL is same as you set in 'DatabaseUtils.props'

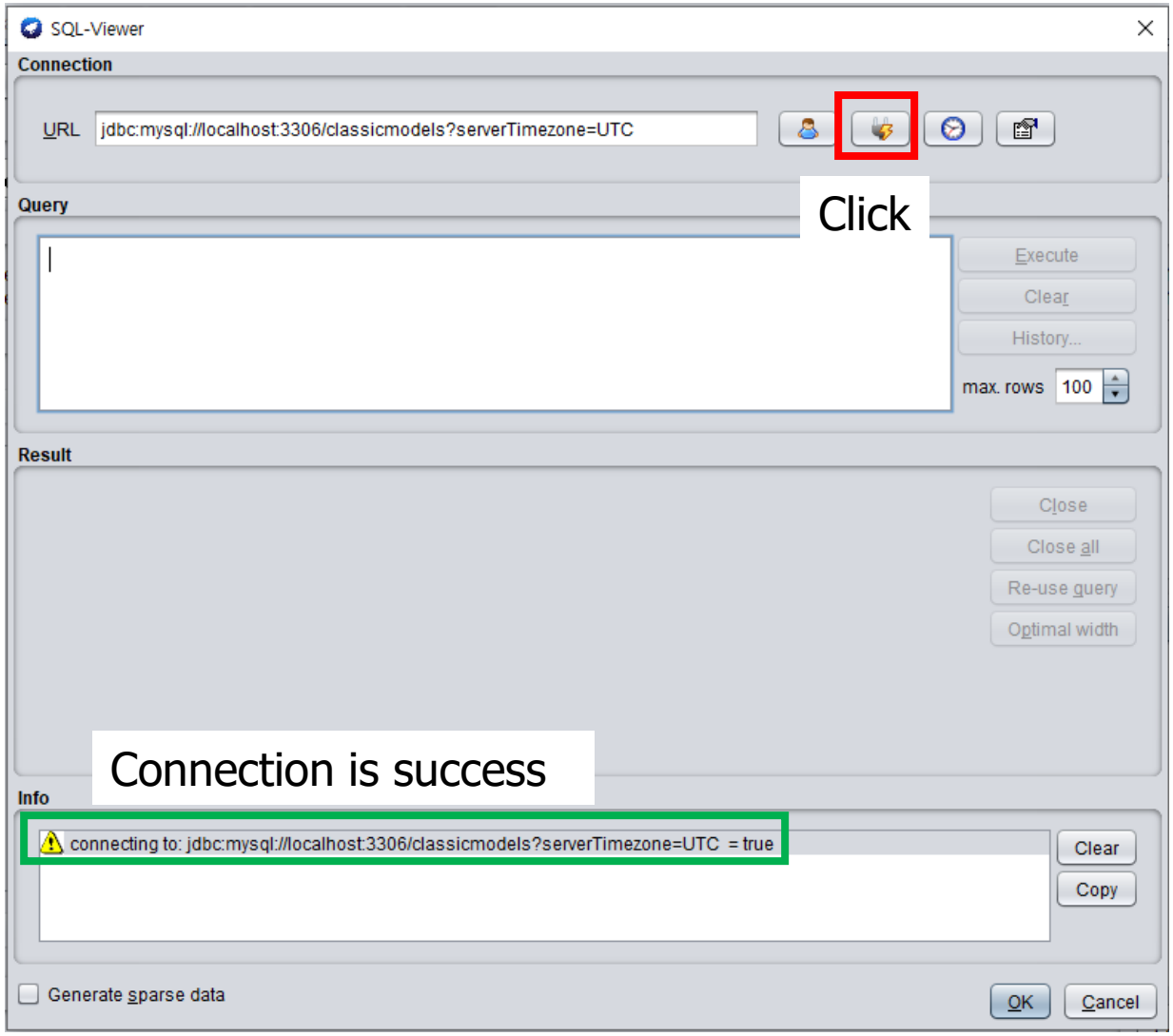


Connet Weka with MySQL

- Put a Username and a Password of database(MySQL's username and password)

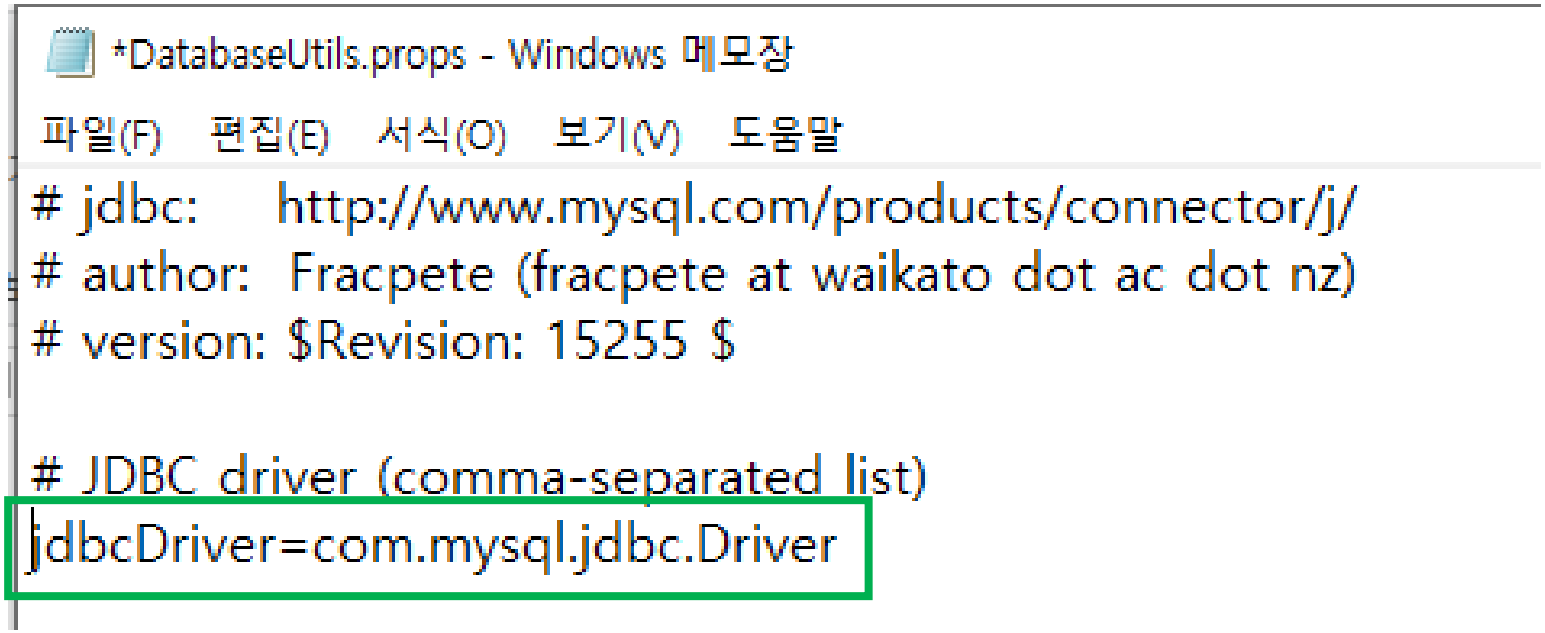


Connet Weka with MySQL



Connet Weka with MySQL

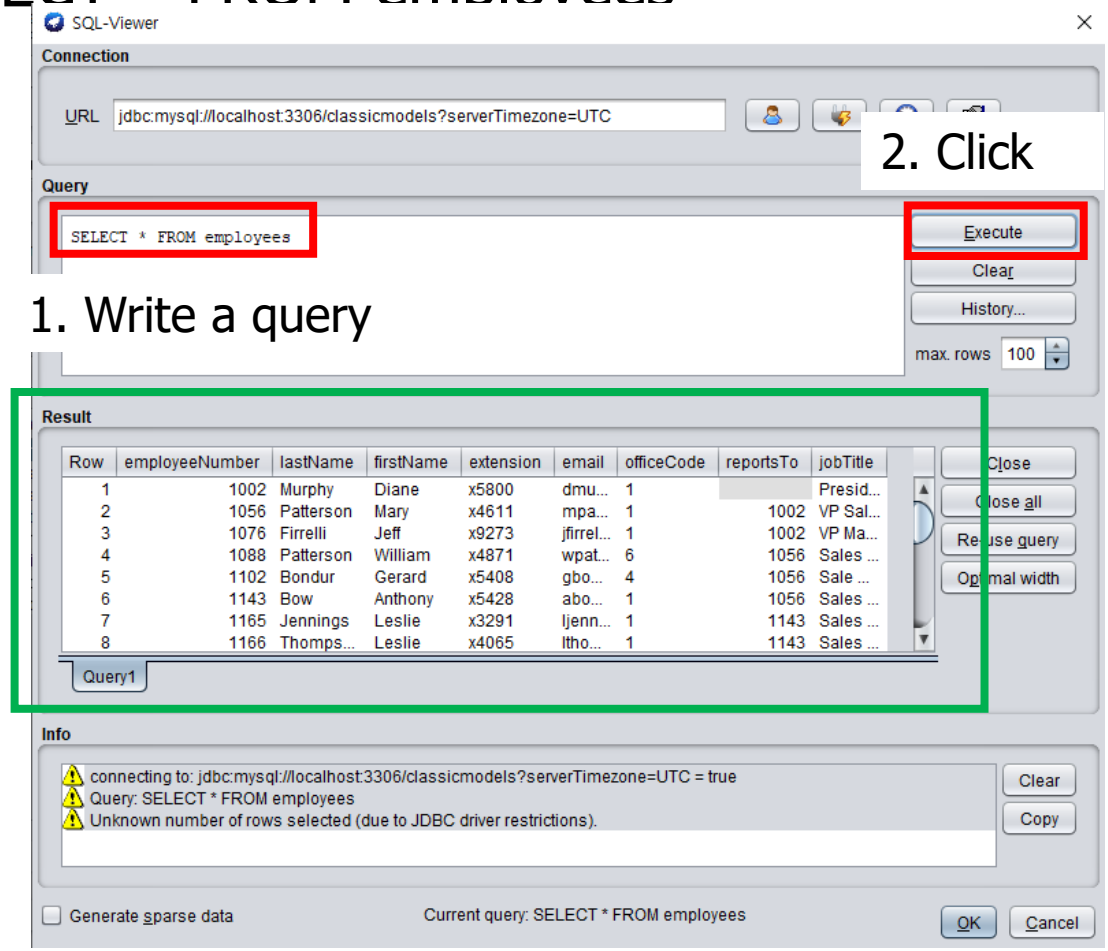
- If 'No suitable driver found …' error occurs, update jdbcDriver in 'DatabaseUtils.props' file and try again
  - # JDBC driver (comma-separated list)
  - jdbcDriver=com.mysql.jdbc.Driver



SELECT

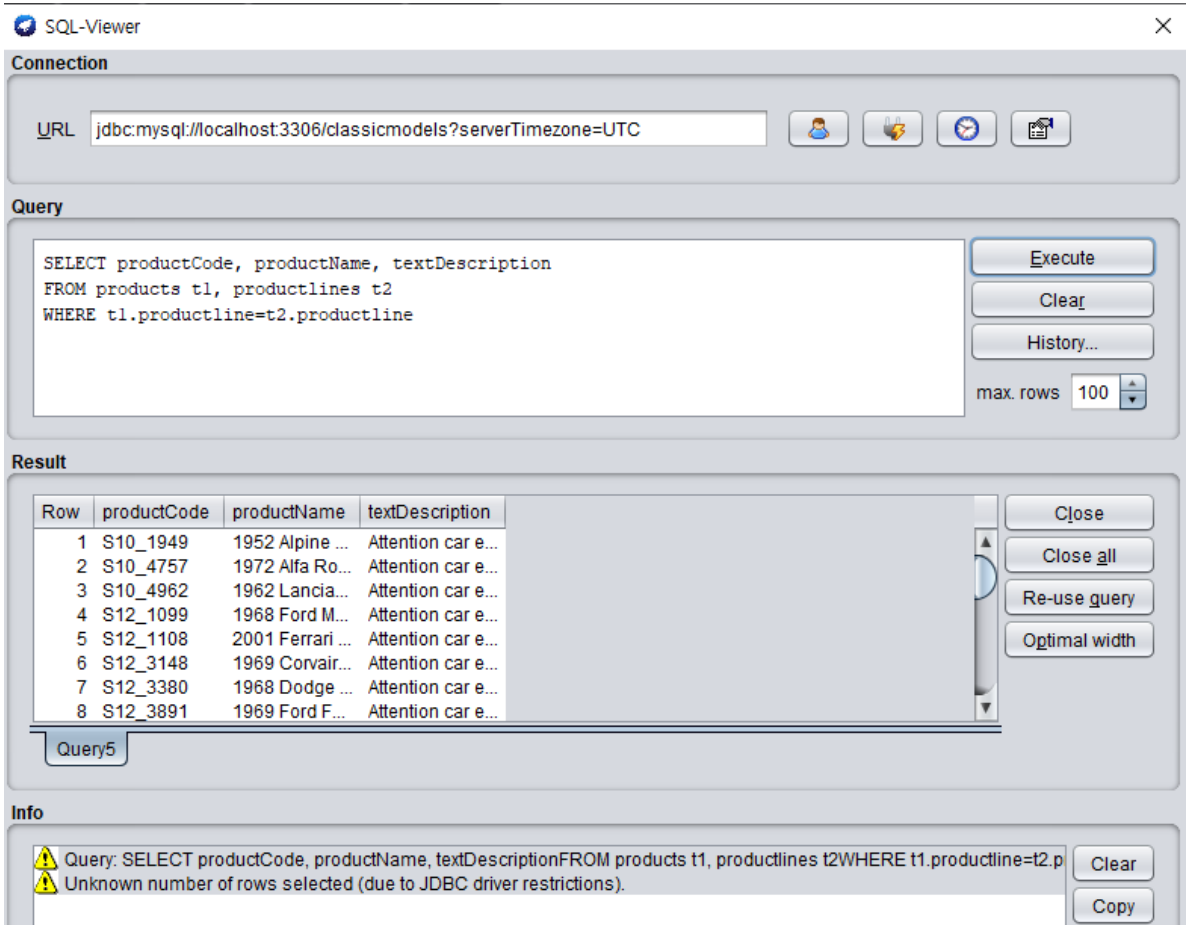
- Write a query in 'Query' and click 'Execute'
  - SELECT * FROM employees

Result



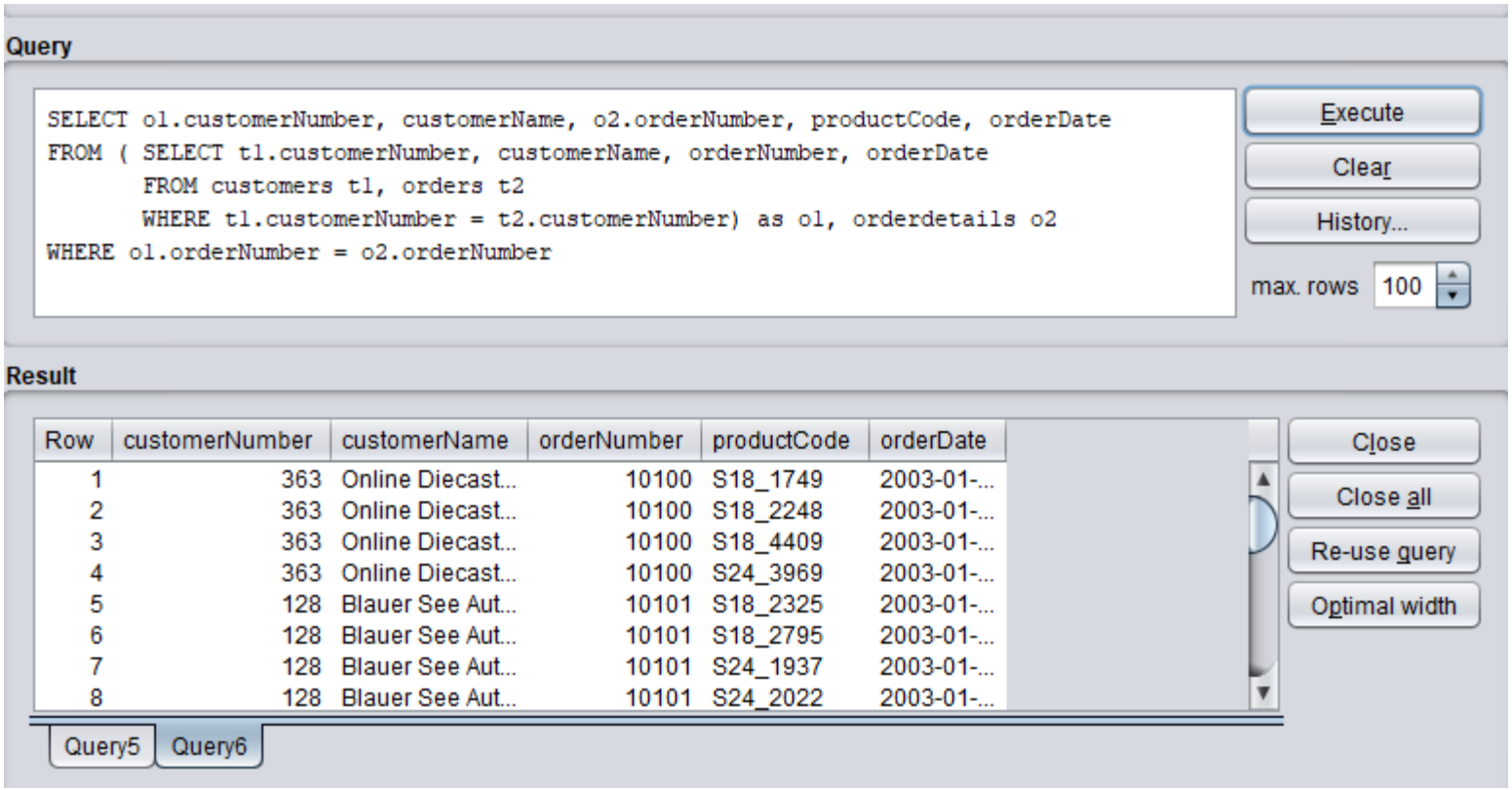
JOIN

```
SELECT productCode, productName, textDescription
FROM products t1, productlines t2
WHERE t1.productline=t2.productline
```



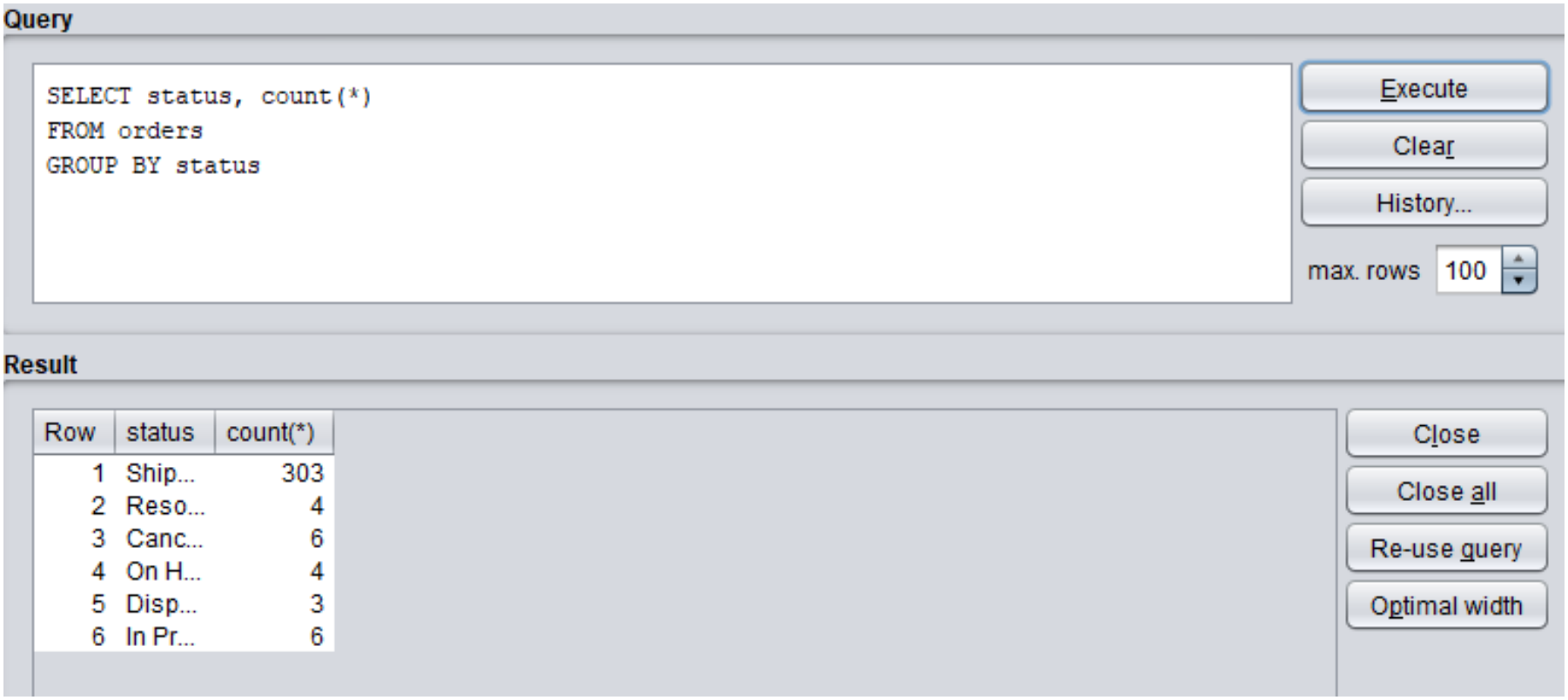
NESTED JOIN

```
SELECT o1.customerNumber, customerName, o2.orderNumber, productCode, orderDate
FROM ( SELECT t1.customerNumber, customerName, orderNumber, orderDate
       FROM customers t1, orders t2
       WHERE t1.customerNumber = t2.customerNumber) as o1, orderdetails o2
WHERE o1.orderNumber = o2.orderNumber
```



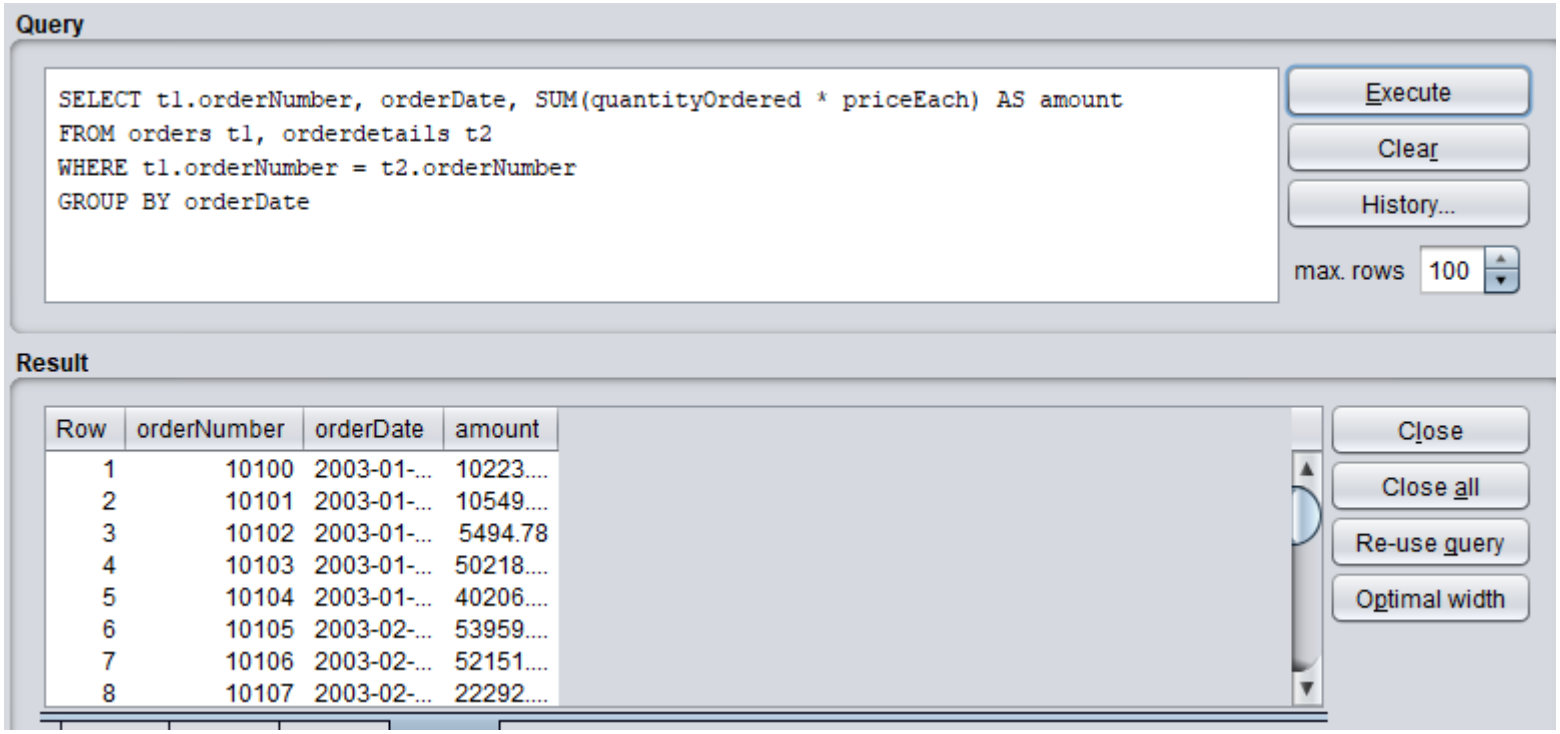
GROUP BY

```
SELECT status, count(*)
FROM orders
GROUP BY status
```



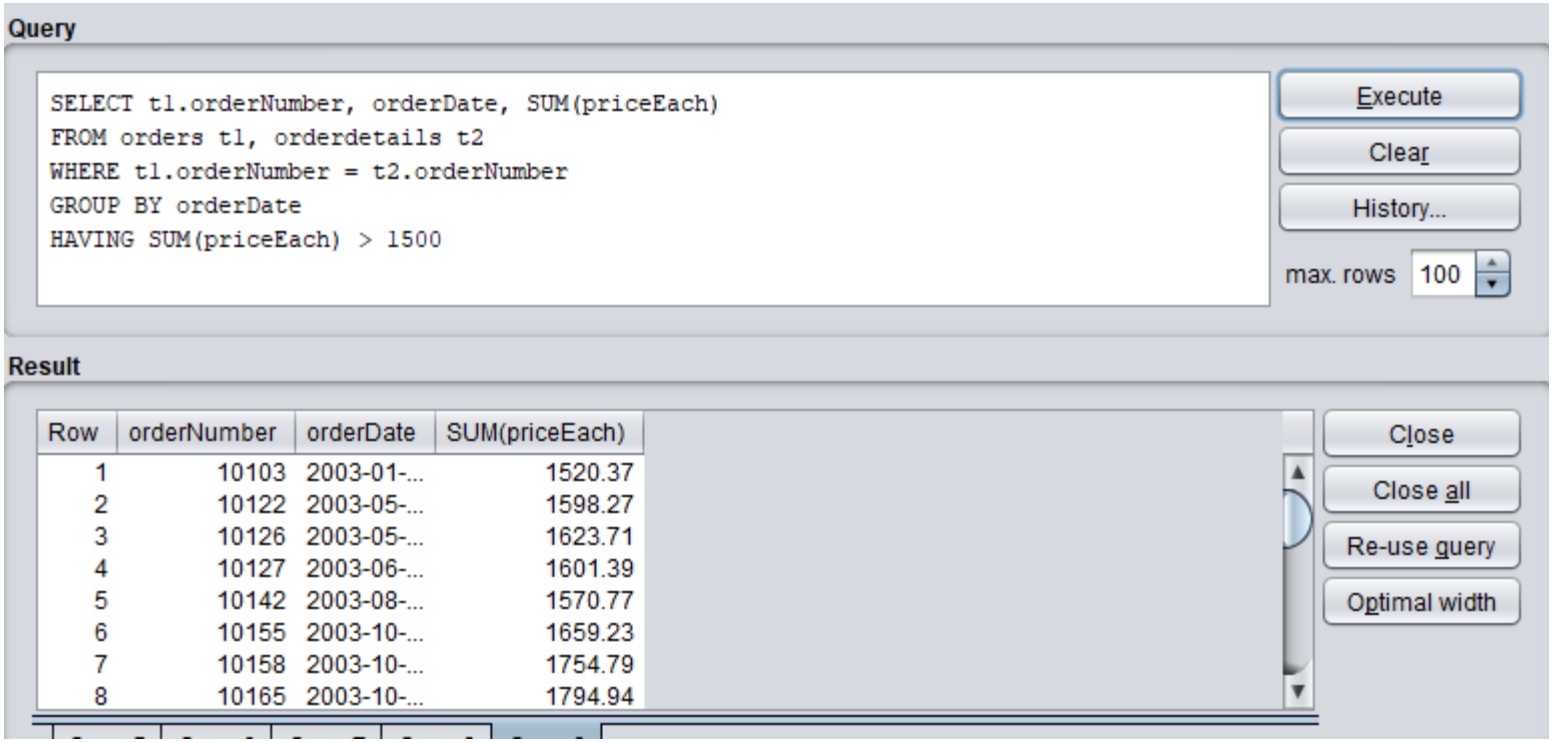
GROUP BY

```
SELECT t1.orderNumber, orderDate, SUM(quantityOrdered * priceEach) AS amount
FROM orders t1, orderdetails t2
WHERE t1.orderNumber = t2.orderNumber
GROUP BY orderDate
```



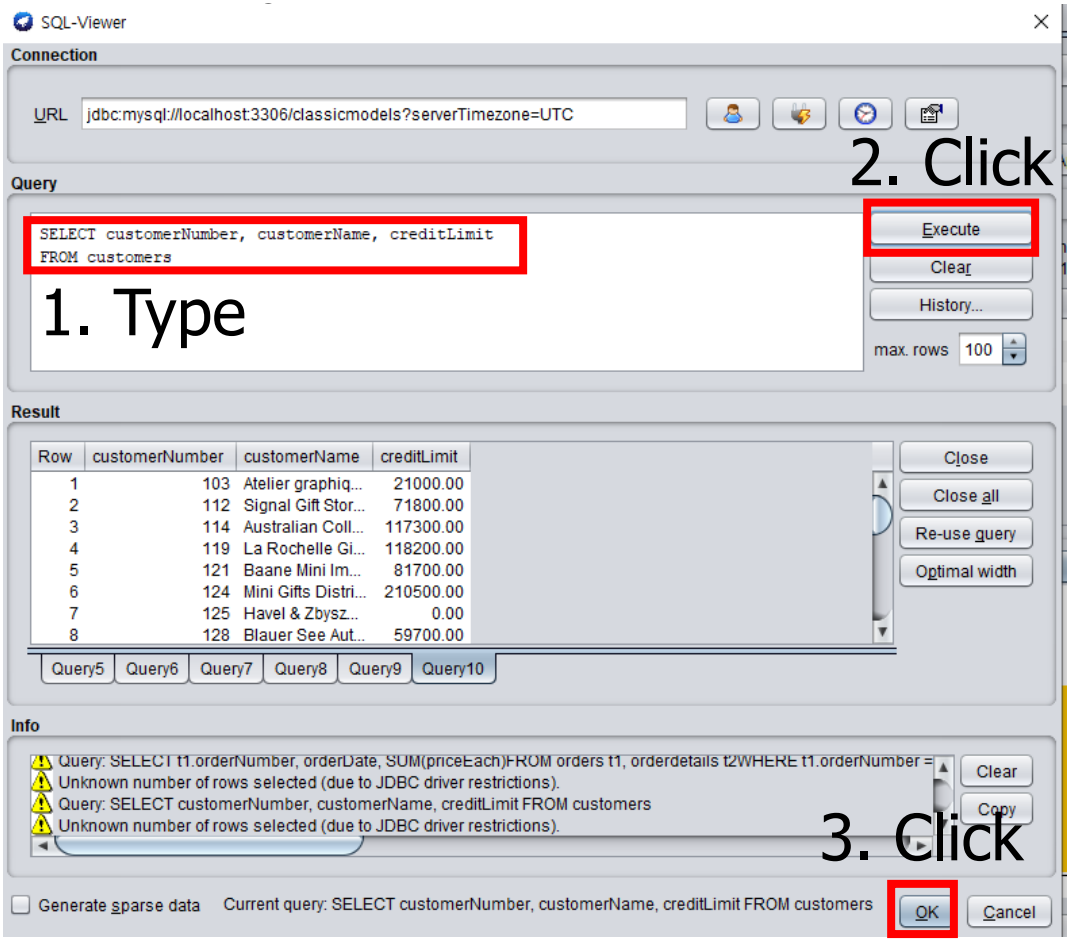
HAVING

```
SELECT t1.orderNumber, orderDate, SUM(priceEach)
FROM orders t1, orderdetails t2
WHERE t1.orderNumber = t2.orderNumber
GROUP BY orderDate
HAVING SUM(priceEach) > 1500
```



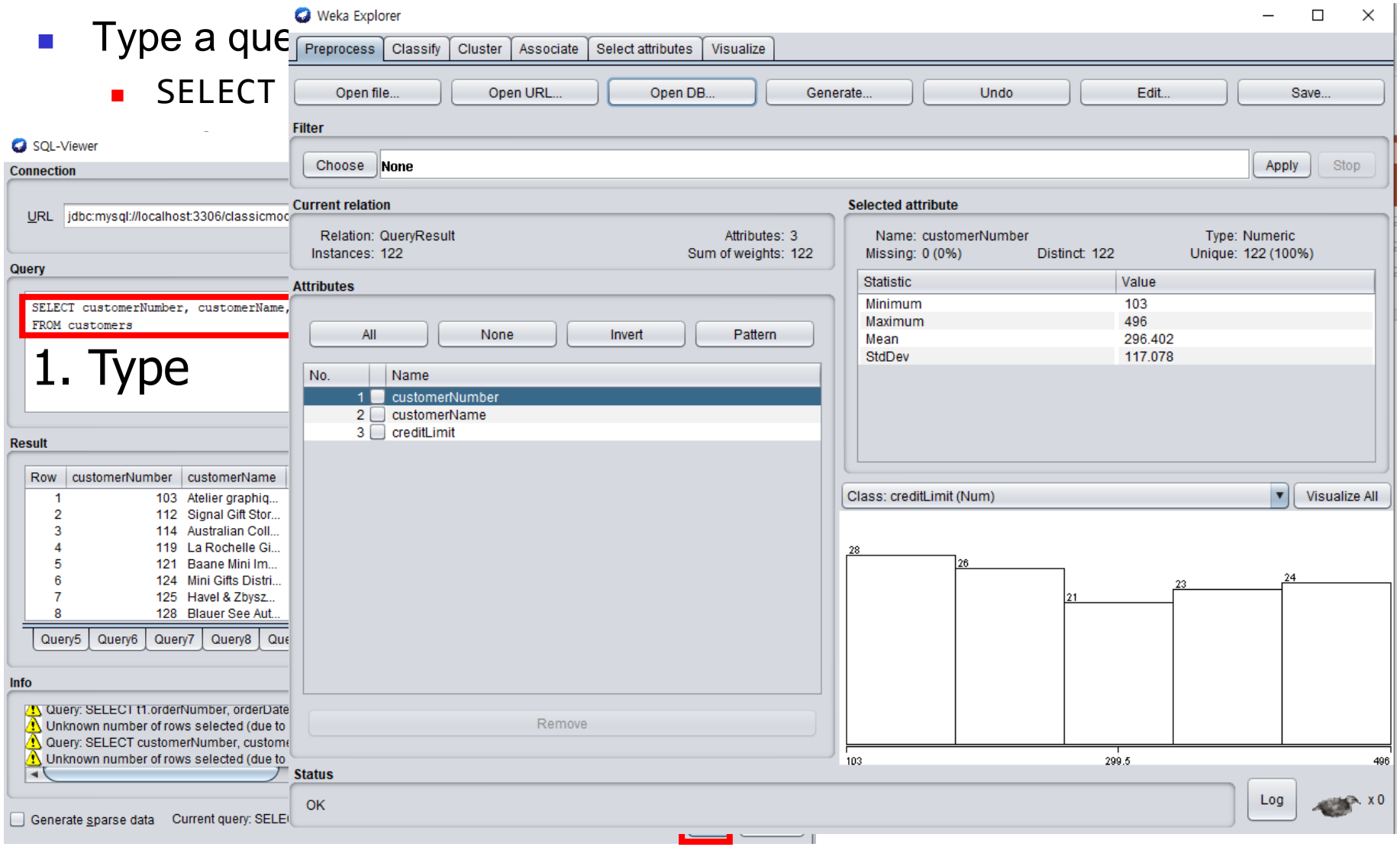
Load table from database

- Type a query, execute and click ‘OK’
  - `SELECT customerNumber, customerName, creditLimit`

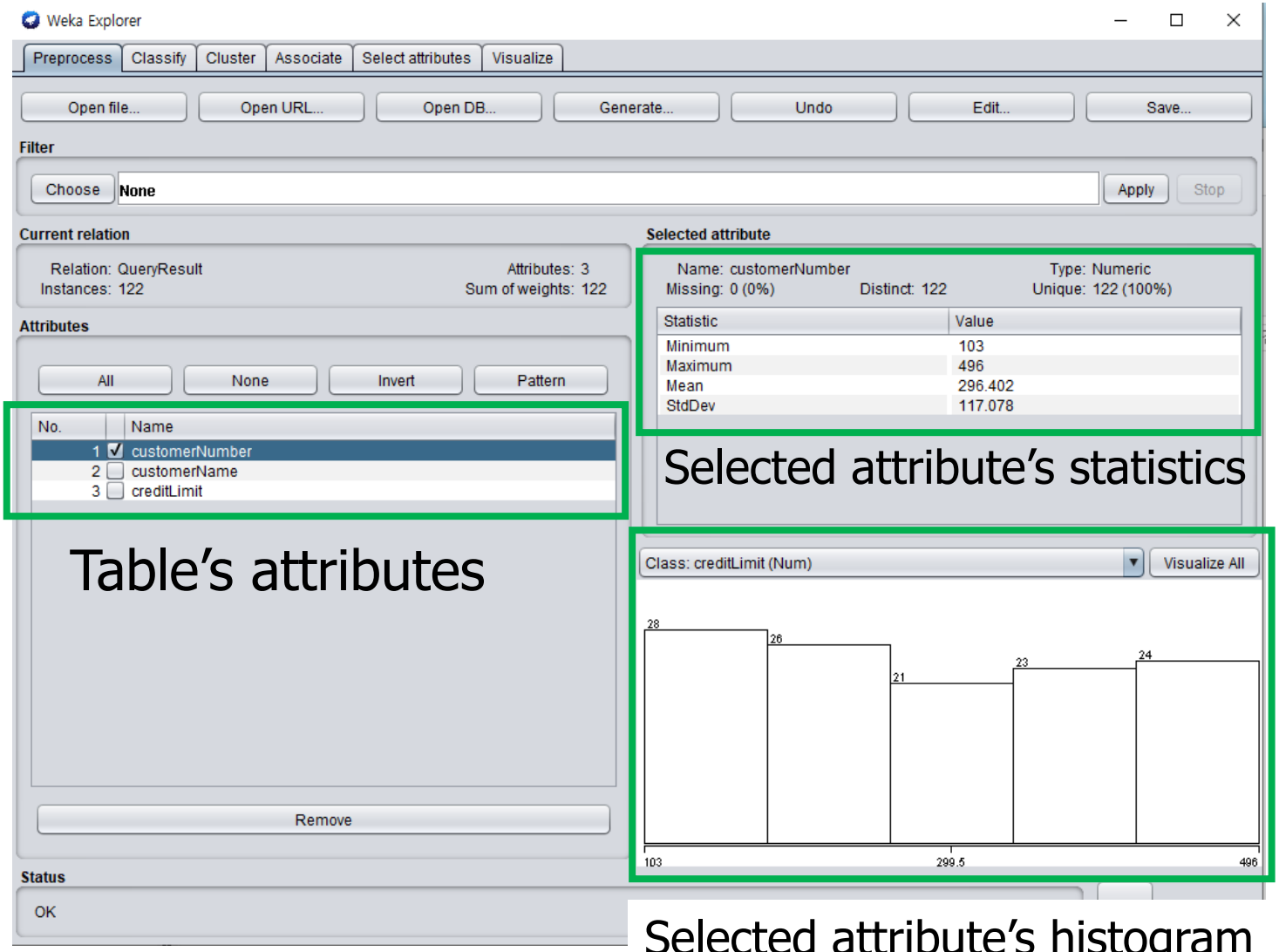


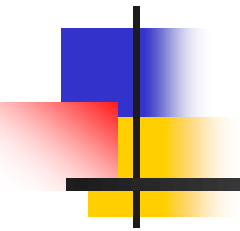
Load table from database

- Type a que
  - SELECT



Load table from database





Python - MySQL

Connecting: PyMySQL

- Open a jupyter notebook document
- Prerequisite: ‘pymysql’ package
- Installing the package by the following code

```
1 !pip install pymysql
```

```
Requirement already satisfied: pymysql in c:₩programdata₩anaconda3₩l
```

Connecting: PyMySQL

- Import packages and connect to the database

```
1  import pymysql.cursors
2  import pandas as pd
3  import sys
```

```
1  IP = "localhost"
2  ID = "root"
3  PW = [redacted]
4  con = pymysql.connect(host = IP, user = ID, password = PW, charset = 'utf8mb4')
```

Connecting: PyMySQL

- Show databases

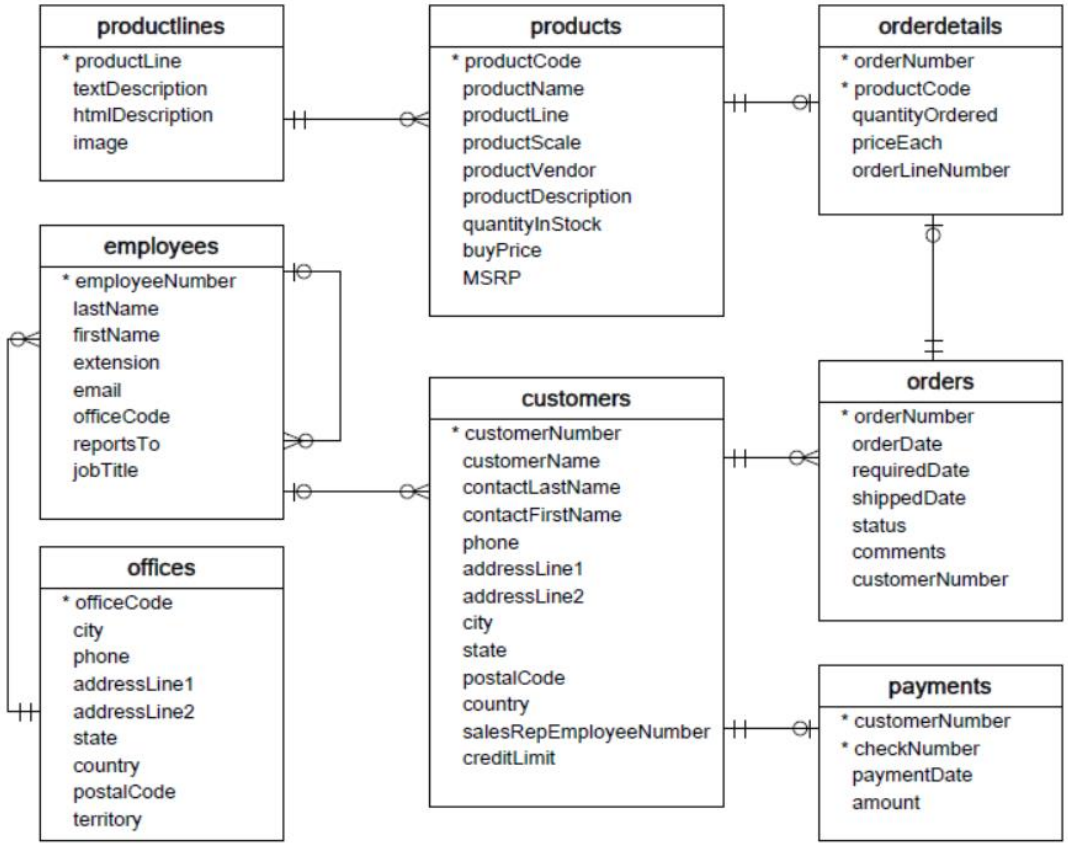
Get the results in the form of pandas.DataFrame object

```
1  pd.read_sql("show databases", con)
```

	Database
0	classicmodels
1	information_schema
2	mysql
3	performance_schema
4	sys

Sample Database Overview

- Source: https://www.mysqltutorial.org/mysql-sample-database.aspx



In [8]:

```
with con.cursor() as cursor:
    sql = "USE classicmodels"
    cursor.execute(sql)
pd.read_sql("SHOW TABLES", con)
```

Out[8]:

	Tables_in_classicmodels
0	customers
1	employees
2	offices
3	orderdetails
4	orders
5	payments
6	productlines
7	products

DB: classicmodels

SELECT

In [10]:
```
1 pd.read_sql("SELECT * From employees", con)
```

Out[10]:

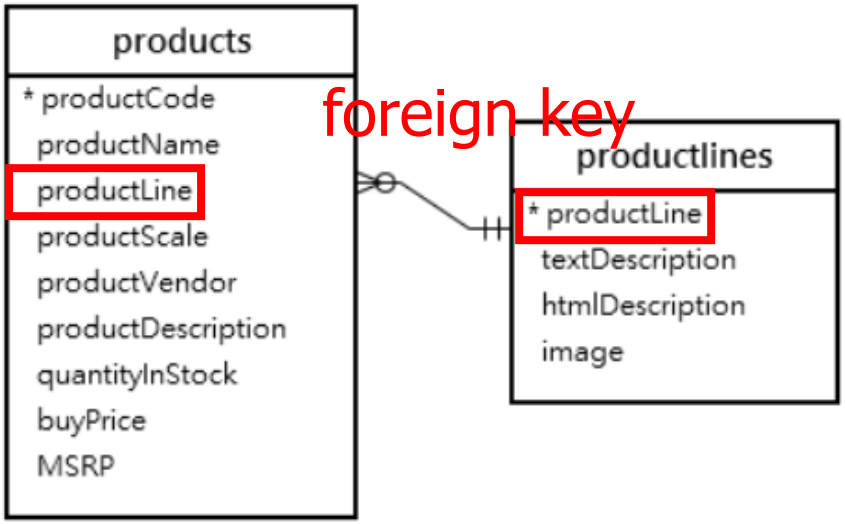
	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
0	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NaN	President
1	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002.0	VP Sales
2	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002.0	VP Marketing
3	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056.0	Sales Manager (APAC)
4	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056.0	Sale Manager (EMEA)
5	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056.0	Sales Manager (NA)
6	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143.0	Sales Rep
7	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143.0	Sales Rep

In [11]:
```
1 pd.read_sql("SELECT lastName, firstname From employees", con)
```

Out[11]:

	lastName	firstname
0	Murphy	Diane
1	Patterson	Mary
2	Firrelli	Jeff
3	Patterson	William

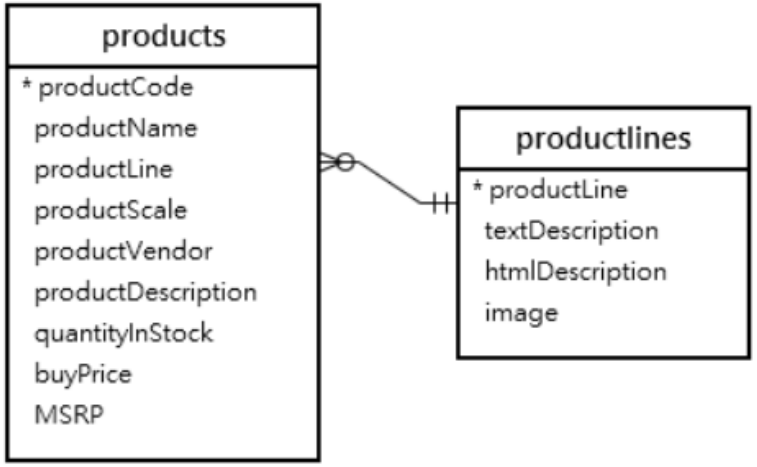
JOIN



- The column productLine in the table products is called the foreign key column.
- Suppose you want to get:
  - The productCode and productName from the products table.
  - The textDescription of product lines from the productlines table.

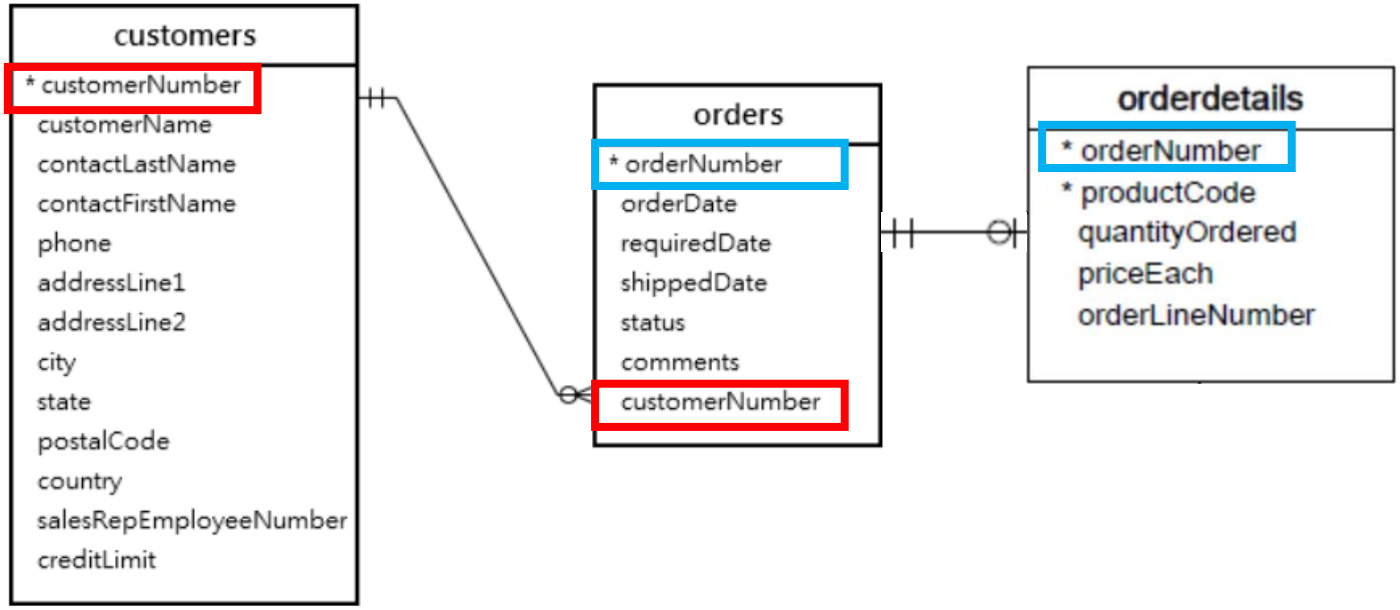
JOIN

```
1 sql = """
2 SELECT productCode, productName, textDescription
3 FROM products t1, productlines t2
4 WHERE t1.productline = t2.productline;"""
5 pd.read_sql(sql, con)
```



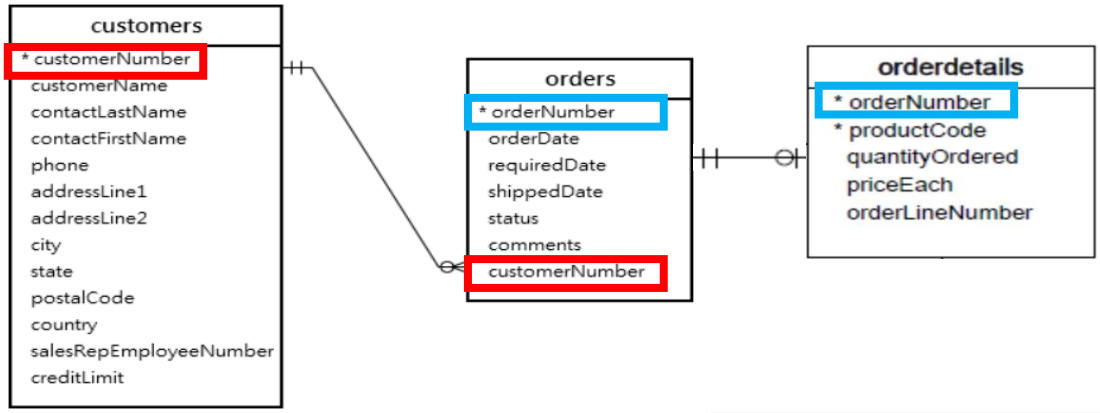
	productCode	productName	textDescription
0	S10_1949	1952 Alpine Renault 1300	Attention car enthusiasts: Make your wildest c...
1	S10_4757	1972 Alfa Romeo GTA	Attention car enthusiasts: Make your wildest c...
2	S10_4962	1962 LanciaA Delta 16V	Attention car enthusiasts: Make your wildest c...
3	S12_1099	1968 Ford Mustang	Attention car enthusiasts: Make your wildest c...
4	S12_1108	2001 Ferrari Enzo	Attention car enthusiasts: Make your wildest c...
5	S12_3148	1969 Corvair Monza	Attention car enthusiasts: Make your wildest c...
6	S12_3380	1968 Dodge Charger	Attention car enthusiasts: Make your wildest c...

NESTED JOIN



- Each customer can have zero or more orders while each order must belong to one customer.

NESTED JOIN




```
sql = """
SELECT o1.customerNumber, customerName, o2.orderNumber, productCode, orderDate
FROM ( SELECT t1.customerNumber, customerName, orderNumber, orderDate
       FROM customers t1, orders t2
       WHERE t1.customerNumber = t2.customerNumber) as o1, orderdetails o2
WHERE o1.orderNumber = o2.orderNumber;"""
pd.read_sql(sql, con)
```

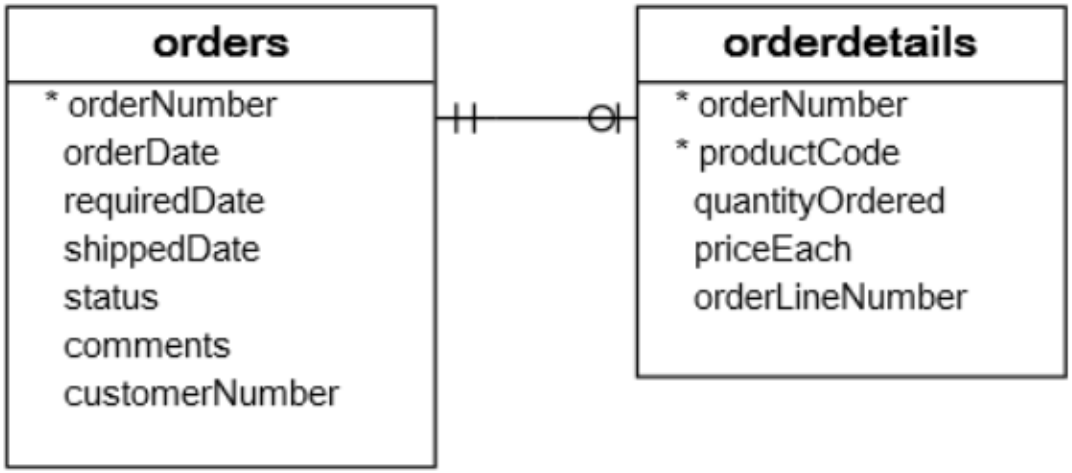
	customerNumber	customerName	orderNumber	productCode	orderDate
0	363	Online Diecast Creations Co.	10100	S18_1749	2003-01-06
1	363	Online Diecast Creations Co.	10100	S18_2248	2003-01-06
2	363	Online Diecast Creations Co.	10100	S18_4409	2003-01-06

GROUP BY

- Group values of the order's status

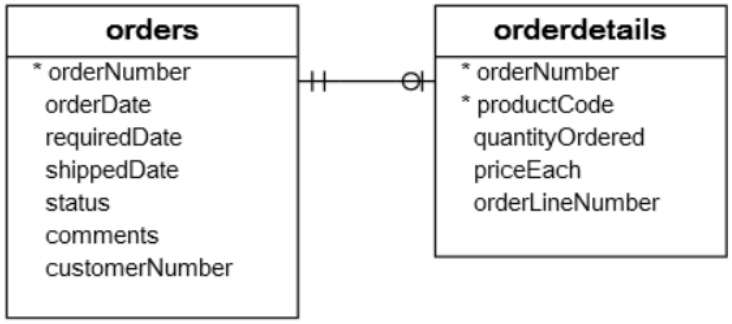
```
sql = """
SELECT status, count(*)
FROM orders
GROUP BY status;"""
pd.read_sql(sql, con)
```

	status	count(*)
0	Shipped	303
1	Resolved	4
2	Cancelled	6
3	On Hold	4
4	Disputed	3
5	In Process	6



GROUP BY

- Get the total amount of all orders by orderDate

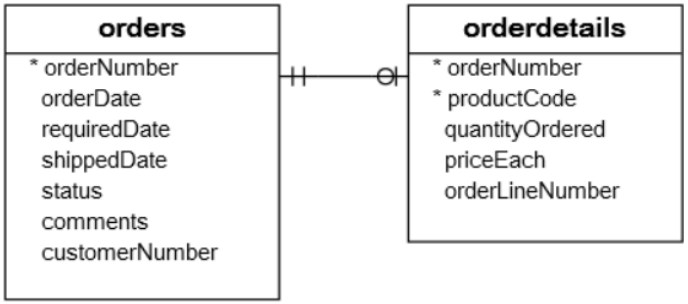



```
sql = """
SELECT t1.orderNumber, orderDate, SUM(quantityOrdered * priceEach) AS amount
FROM orders t1, orderdetails t2
WHERE t1.orderNumber = t2.orderNumber
GROUP BY orderDate; """
pd.read_sql(sql, con)
```

	orderNumber	orderDate	amount
0	10100	2003-01-06	10223.83
1	10101	2003-01-09	10549.01
2	10102	2003-01-10	5494.78
3	10103	2003-01-29	50218.95
4	10104	2003-01-31	40206.20

HAVING

- To filter the groups returned by GROUP BY clause, you use a HAVING clause.




```
sql = """
SELECT t1.orderNumber, orderDate, SUM(priceEach)
FROM orders t1, orderdetails t2
WHERE t1.orderNumber = t2.orderNumber
GROUP BY orderDate
HAVING SUM(priceEach) > 1500; """
pd.read_sql(sql, con)
```

	orderNumber	orderDate	SUM(priceEach)
0	10103	2003-01-29	1520.37
1	10122	2003-05-08	1598.27
2	10126	2003-05-28	1623.71
3	10127	2003-06-03	1601.39

Usage In PyMySQL

- You can get the python array which corresponds to the selected records

```
1  df = pd.read_sql("SELECT customerNumber, customerName, creditLimit FROM customers", con)
2  print('df.values.shape = ', df.values.shape)
3  df.values
```

```
df.values.shape =  (122, 3)

array([[103, 'Atelier graphique', 21000.0],
       [112, 'Signal Gift Stores', 71800.0],
       [114, 'Australian Collectors, Co.', 117300.0],
       [119, 'La Rochelle Gifts', 118200.0],
       [121, 'Baane Mini Imports', 81700.0],
       [124, 'Mini Gifts Distributors Ltd.', 210500.0],
       [125, 'Havel & Zbyszek Co', 0.0],
       [128, 'Blauer See Auto, Co.', 59700.0],
       [129, 'Mini Wheels Co.', 64600.0],
       [131, 'Land of Toys Inc.', 114900.0],
       ...
       [489, 'Double Decker Gift Stores, Ltd', 43300.0],
       [495, 'Diecast Collectables', 85100.0],
       [496, "Kelly's Gift Shop", 110000.0]], dtype=object)
```

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts 
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

Data Warehouse and OLAP

- Generalize and consolidate data in multidimensional space.
- Construction involves
  - Data cleaning
  - Data integration
  - Data transformation
- Construction is an important preprocessing step for data mining.
- Provide online analytical processing (OLAP) tools for the interactive analysis of multidimensional data of varied granularities
  - Facilitates effective data generalization and data mining.
- Many other data mining functions can be integrated with OLAP operations to enhance interactive mining of knowledge at multiple levels of abstraction.
- Thus, data warehousing and OLAP form an essential step in the knowledge discovery process.

What Is a Data Warehouse?

- Provides architectures and tools to systematically organize, understand, and use their data to make strategic decisions.
- Valuable tools in today's competitive, fast-evolving world.
- In the last several years, many firms have spent millions of dollars in building enterprise-wide data warehouses.
- Data warehousing is the latest must-have marketing weapon—a way to retain customers by learning more about their needs.

What Is a Data Warehouse?

- Loosely speaking, a data repository that is maintained separately from an organization's operational databases.
  - Data warehouse systems allow for integration of a variety of application systems.
  - They support information processing by providing a solid platform of consolidated historic data for analysis.
- A subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process.
  - The four keywords—subject-oriented, integrated, time-variant, and nonvolatile—distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - Relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element"

Data Warehouse—Nonvolatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - Initial loading of data and *access of data*

Data Warehouse

- In sum, a data warehouse is a semantically consistent data store that serves as a physical implementation of a decision support data model.
- It stores the information an enterprise needs to make strategic decisions.
- A data warehouse is also often viewed as an architecture, constructed by integrating data from multiple heterogeneous sources to support structured and/or ad hoc queries, analytical reporting, and decision making.

Data Warehousing

- The process of constructing and using data warehouses.
- The construction of a data warehouse requires data cleaning, data integration, and data consolidation.
- The utilization of a data warehouse often necessitates a collection of decision support technologies.
- This allows "knowledge workers" (e.g., managers, analysts, and executives) to use the warehouse to quickly and conveniently obtain an overview of the data, and to make sound decisions based on information in the warehouse.

Information from Data Warehouses

- Many organizations use this information to support business decision-making activities, including
  - Increasing customer focus, which includes the analysis of customer buying patterns (such as buying preference, buying time, budget cycles, and appetites for spending)
  - Repositioning products and managing product portfolios by comparing the performance of sales by quarter, by year, and by geographic regions in order to fine-tune production strategies
  - Analyzing operations and looking for sources of profit
  - Managing customer relationships, making environmental corrections, and managing the cost of corporate assets.

Data Warehouses and Heterogeneous Database Integration

- Data warehousing is also very useful from the point of view of heterogeneous database integration.
- Organizations typically collect diverse kinds of data and maintain large databases from multiple, heterogeneous, autonomous, and distributed information sources.
- It is highly desirable, yet challenging, to integrate such data and provide easy and efficient access to it.
- Much effort has been spent in the database industry and research community toward achieving this goal.
- The traditional database approach to heterogeneous database integration is to build wrappers and integrators (or mediators) on top of multiple, heterogeneous databases.

Heterogeneous Database Integration - Query-driven Approach

- When a query is posed to a client site, a metadata dictionary is used to translate the query into queries appropriate for the individual heterogeneous sites involved.
- These queries are then mapped and sent to local query processors.
- The results returned from the different sites are integrated into a global answer set.
- This query-driven approach requires complex information filtering and integration processes, and competes with local sites for processing resources.
- It is inefficient and potentially expensive for frequent queries, especially queries requiring aggregations.

Heterogeneous Database Integration - Update-driven Approach

- Information from multiple, heterogeneous sources is integrated in advance and stored in a warehouse for direct querying and analysis.
- Unlike online transaction processing databases, data warehouses do not contain the most current information.
- However, a data warehouse brings high performance to the integrated heterogeneous database system because data are copied, preprocessed, integrated, annotated, summarized, and restructured into one semantic data store.
- Furthermore, query processing in data warehouses does not interfere with the processing at local sources.
- Moreover, data warehouses can store and integrate historic information and support complex multidimensional queries.
- As a result, data warehousing has become popular in industry.

Differences between Database Systems and Data Warehouses

- The major task of online operational database systems is to perform online transaction and query processing.
  - Called online transaction processing (OLTP) systems.
  - They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.
- Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making.
  - Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users.
  - These systems are known as online analytical processing (OLAP) systems.

Major Distinguishing Features of OLTP and OLAP

- Users and system orientation
  - OLTP system – is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals.
  - OLAP system – is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.
- Data contents
  - OLTP system – manages current data that, typically, are too detailed to be easily used for decision making.
  - OLAP system - manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.
    - These features make the data easier to use for informed decision making.

Major Distinguishing Features of OLTP and OLAP

- Database design
  - OLTP system - adopts an entity-relationship (ER) data model and an application-oriented database design.
  - OLAP system - adopts either a star or a snowflake model and a subject-oriented database design.
- View
  - OLTP system - focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations.
  - OLAP system – spans multiple versions of a database schema, due to the evolutionary process of an organization.
    - Also deals a with information that originates from different organizations, integrating information from many data stores.
    - Because of their huge volume, OLAP data are stored on multiple storage media.

Major Distinguishing Features of OLTP and OLAP

- Access patterns
  - OLTP system – the access patterns of an OLTP system consist mainly of short, atomic transactions.
    - Such a system requires concurrency control and recovery mechanisms.
  - OLAP system - accesses to OLAP systems are mostly read-only operations (because most data warehouses store historic rather than up-to-date information), although many could be complex queries.

Summary of OLTP vs. OLAP

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Why Have a Separate Data Warehouse?

- Promote the high performance of OLTP and OLAP systems.
- An operational database is designed and tuned from known tasks and workloads like indexing and hashing using primary keys, searching for particular records, and optimizing known queries.
- Data warehouse queries involve the computation of large data groups at summarized levels, and may require the use of special data organization, access, and implementation methods based on multidimensional views.
- Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.

Why Have a Separate Data Warehouse?

- An operational database supports the concurrent processing of multiple transactions.
  - Concurrency control and recovery mechanisms (e.g., locking and logging) are required to ensure the consistency and robustness of transactions.
- An OLAP query often needs read-only access of data records for summarization and aggregation.
  - Concurrency control and recovery mechanisms, if applied for such OLAP operations, may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

Why Have a Separate Data Warehouse?

- The separation of database systems from data warehouses is based on the different structures, contents, and uses of the data.
  - Decision support requires historic data, whereas operational databases do not typically maintain historic data.
  - Decision support requires consolidation (e.g., aggregation and summarization) of data from heterogeneous sources, resulting in high-quality, clean, integrated data.
  - Operational databases contain only detailed raw data, such as transactions.
- Because the two systems provide quite different functionalities and require different kinds of data, it is presently necessary to maintain separate databases.
- However, many vendors of operational relational database management systems are beginning to optimize such systems to support OLAP queries.

Why a Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
  - missing data: Decision support requires historical data which operational DBs do not typically maintain
  - data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

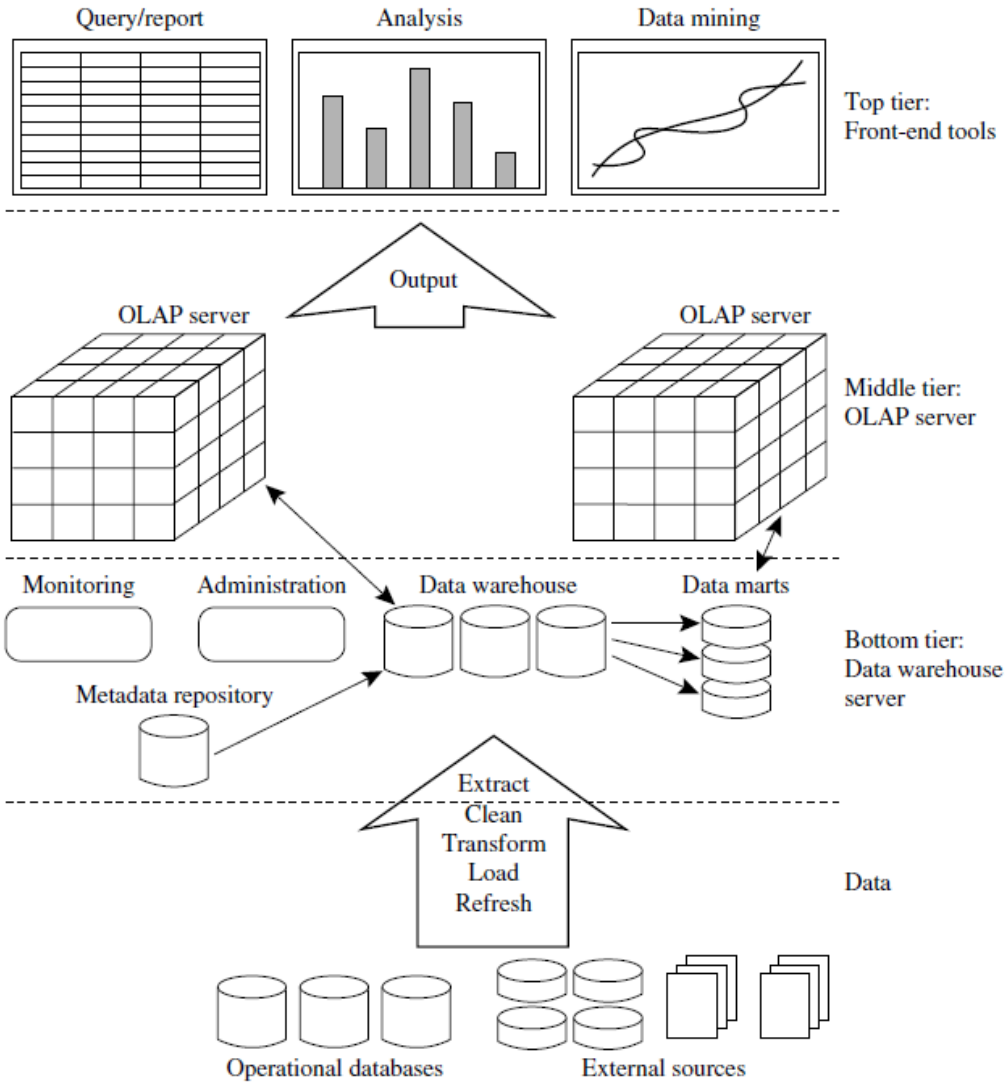
Data Warehousing: A Three-tiered Architecture

- The **bottom tier** is a warehouse database server that is almost always a relational database system.
  - Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants).
  - These tools and utilities perform data extraction, cleaning, and, as well as load and refresh functions to update the data warehouse.
  - Data are extracted using application program interfaces known as gateways.
    - A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server.
    - Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection).
  - This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

Data Warehousing: A Three-tiered Architecture

- The **middle tier** is an OLAP server that is typically implemented using
  - Either a relational OLAP (ROLAP) model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations)
  - Or a multidimensional OLAP (MOLAP) model (i.e., a special-purpose server that directly implements multidimensional data and operations).
- The **top tier** is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

A Three-tiered Data Warehousing Architecture



Data Warehouse Models

- Data Warehouse Models
  - Enterprise Warehouse
  - Data Mart
  - Virtual Warehouse
- Enterprise warehouse model
  - Collects all of the information about subjects spanning the entire organization.
  - Typically contains detailed data as well as summarized data
  - Ranges in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.
  - Be implemented on traditional mainframes, computer super servers, or parallel architecture platforms.
  - Requires extensive business modeling and may take years to design and build.

Data Warehouse Models

- Data mart model
  - Contains a subset of corporate-wide data that is of value to a specific group of users.
  - The scope is confined to specific selected subjects such as a marketing data mart confined its subjects to customer, item, and sales.
  - Data marts are usually implemented on low-cost departmental servers that are Unix/Linux or Windows based.
  - The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years.
  - Data marts can be categorized as
    - Independent - are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area.
    - Dependent - are sourced directly from enterprise data warehouses.

Data Warehouse Models

- Virtual warehouse model
  - A virtual warehouse is a set of views over operational databases.
  - For efficient query processing, only some of the possible summary views may be materialized.
  - A virtual warehouse is easy to build but requires excess capacity on operational database servers.

Data Warehouse Models (Recap.)

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

A Recommended Approach for Data Warehouse Development

- Implement the warehouse in an incremental and evolutionary manner.

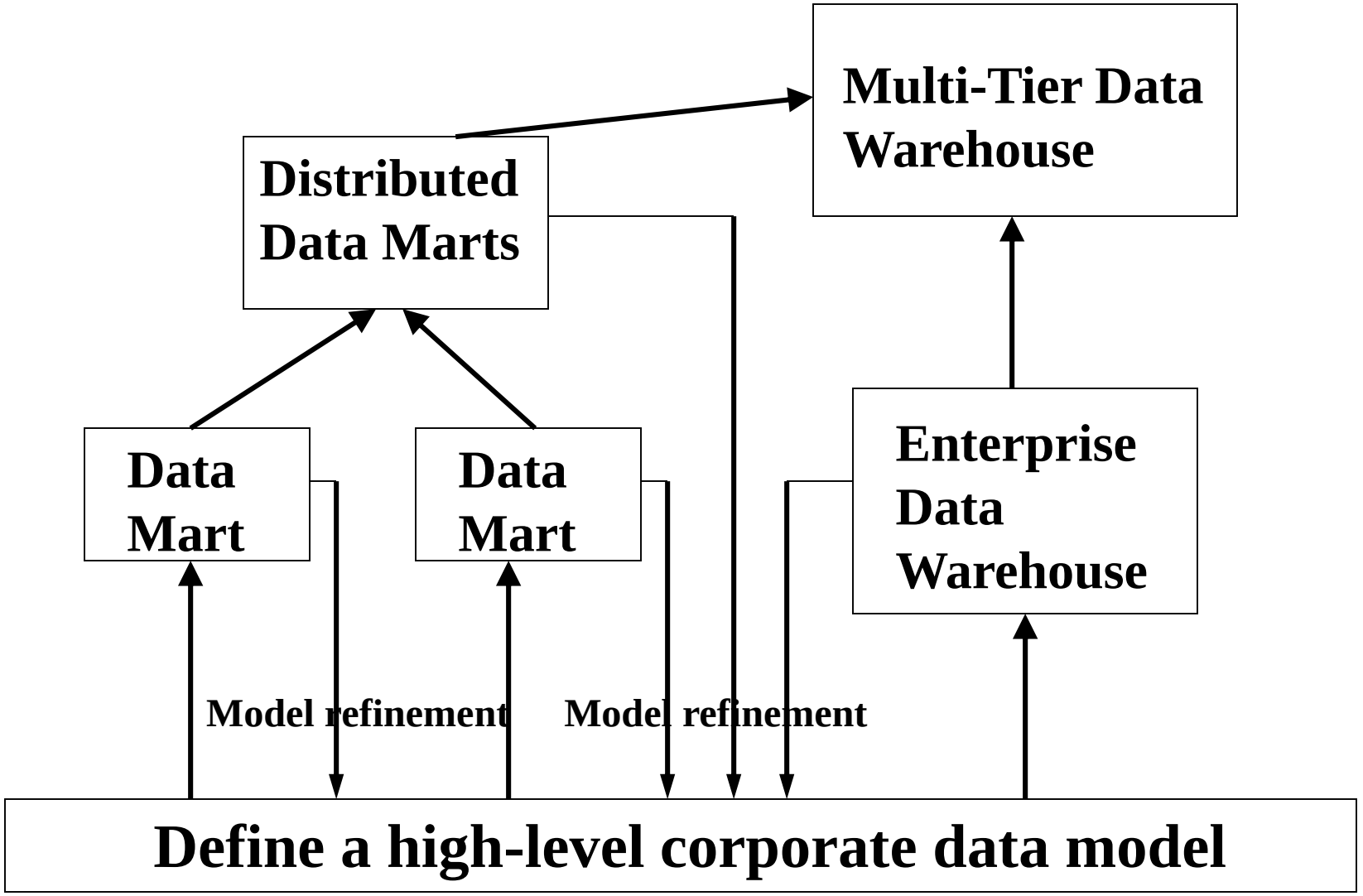
  (1) A high-level corporate data model is defined within a reasonably short period (such as one or two months) that provides a corporate-wide, consistent, integrated view of data among different subjects and potential usages - greatly reduce future integration problems.

  (2) Independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set noted before.

  (3) Distributed data marts can be constructed to integrate different data marts via hub servers.

  (4) A multitier data warehouse is constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.

Data Warehouse Development: A Recommended Approach



Back-end Tools and Utilities

- Data warehouse systems use back-end tools and utilities to populate and refresh their data.
- These tools and utilities include the functions
  - Data extraction
    - gathers data from multiple, heterogeneous, and external sources.
  - Data cleaning
    - detects errors in the data and rectifies them when possible.
  - Data transformation
    - converts data from legacy or host format to warehouse format.
  - Load
    - sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
  - Refresh
    - propagates the updates from the data sources to the warehouse.

Metadata

- Metadata is the data that define warehouse objects.
  - Created for the data names and definitions of the given warehouse.
  - Additionally Created for timestamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.
- Metadata play a very different role than other data warehouse data.
  - A directory to help the decision support system analyst locate the contents of the data warehouse
  - A guide to the data mapping when data are transformed from the operational environment to the data warehouse environment.
  - A guide to the algorithms used for summarization between the current detailed data and the lightly summarized data, and between the lightly summarized data and the highly summarized data.
- Metadata should be stored and managed persistently (i.e., on disk).

Metadata Repository Contain

- Data warehouse structure - the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- Operational metadata - data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- Algorithms used for summarization - measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- Mapping from the operational environment to the data warehouse - source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).
- Data related to system performance - indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.
- Business metadata - business terms and definitions, data ownership information, and charging policies.

Pros and Cons of Top-down and Bottom-up Approaches to Data Warehouse Development

- The top-down approach
  - Serves as a systematic solution and minimizes integration problems.
  - Is expensive and lacks flexibility due to the difficulty in achieving consistency and consensus for a common data model for the entire organization.
- The bottom-up approach
  - Provides flexibility, low cost, and rapid return of investment.
  - Can lead to problems when integrating various disparate data marts into a consistent enterprise data warehouse.

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

Data Warehouse Modeling: Data Cube and OLAP

- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - **Dimension tables**, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - **Fact table** contains **measures** (such as dollars_sold) and keys to each of the related dimension tables
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

Data Cube - Dimensions

- Dimensions – perspectives or entities with respect to which an organization wants to keep records.
  - Allow the store to keep track of things like monthly sales of items and the branches and locations at which the items were sold.
  - Each dimension may have a table associated with it, called a dimension table, which further describes the dimension.
  - A dimension table for item may contain the attributes item name, brand, and type.
- Facts - numeric measures.
  - Think of them as the quantities by which we want to analyze relationships between dimensions.
  - e.g., facts for a sales data warehouse include dollars sold (sales amount in dollars), units sold (number of units sold), and amount budgeted.
  - The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.

An Example of Data Cube and Multidimensional Data Model

- Looking at this 2-D representation of the sales for Vancouver shown with respect to the time dimension (organized in quarters) and the item dimension (organized according to the types of items sold).
  - The measure displayed is dollars sold (in thousands).

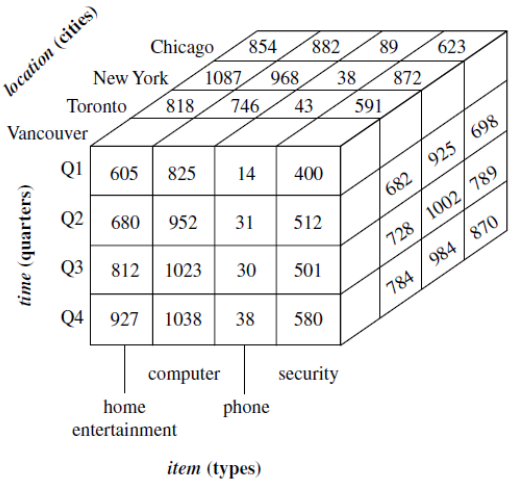
location = *"Vancouver"*				
	item *(type)*			
time *(quarter)*	*home entertainment*	*computer*	*phone*	*security*
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

An Example of Data Cube and Multidimensional Data Model

- View the sales data with a third dimension.
- View the data according to time and item, as well as location, for the cities Chicago, New York, Toronto, and Vancouver.
- Table representation of 3-D data

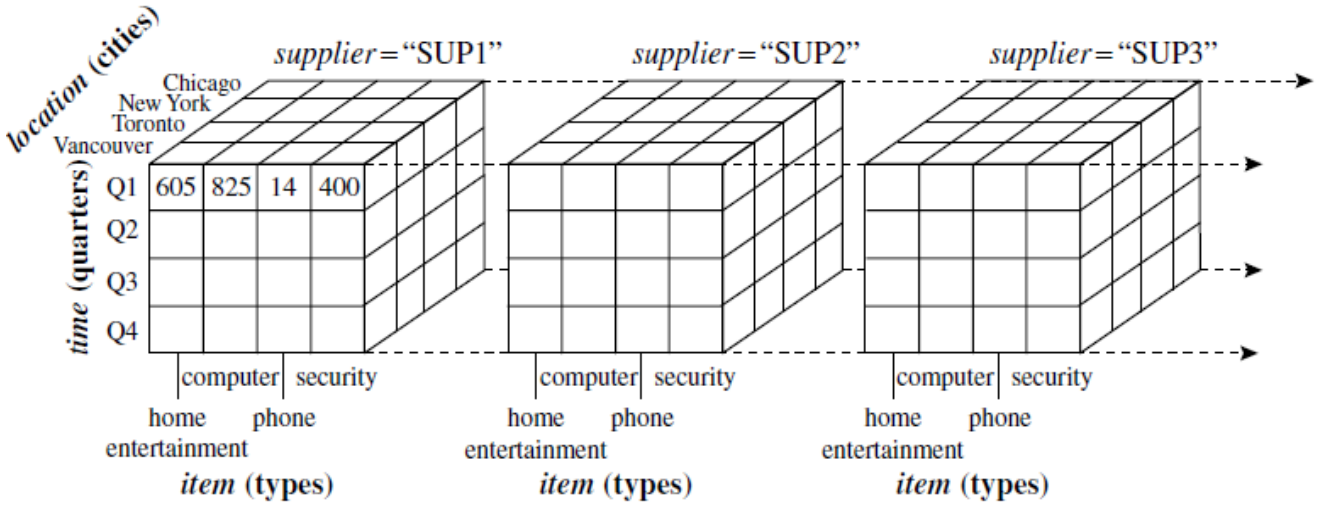
	location = "Chicago"				***location*** = "New York"				***location*** = "Toronto"				***location*** = "Vancouver"			
	item				***item***				***item***				***item***			
	home				*home*				*home*				*home*			
time	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

- Data cube representation of 3-D data.



An Example of Data Cube and Multidimensional Data Model

- View our sales data with an additional fourth dimension such as supplier - viewing things in 4-D becomes tricky.
- We can think of a 4-D cube as being a series of 3-D cubes.



- If we continue in this way, we may display any n-dimensional data as a series of (n-1)-dimensional "cubes."

Data Cube

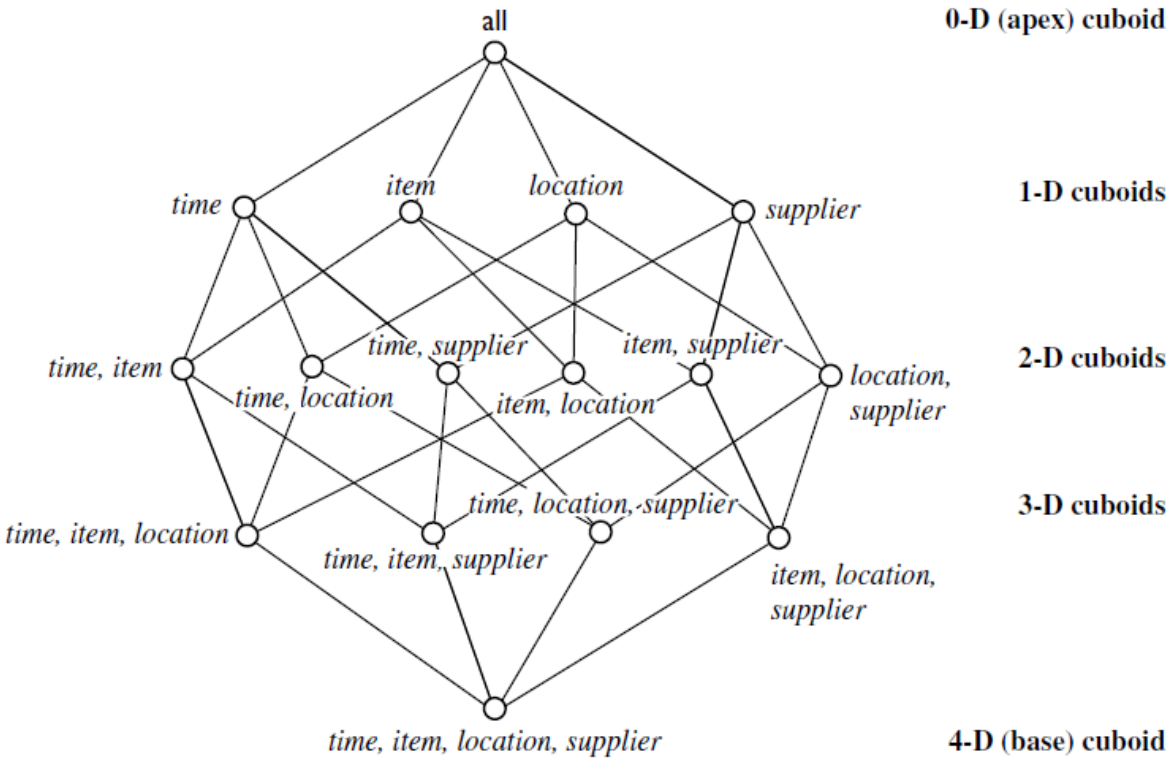
- The actual physical storage of data cube may differ from its logical representation.
- The important thing to remember is that data cubes are n-dimensional and do not confine data to 3-D.
- A data cube is often referred to as a cuboid.
- Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions.
- The result would form a lattice of cuboids, each showing the data at a different level of summarization, or group-by.
- The lattice of cuboids is then referred to as a data cube.

Cuboids

- Base cuboid - the cuboid that holds the lowest level of summarization.
  - e.g., the base cuboid for the given time, item, location, and supplier dimensions.
- (Nonbase) cuboid
  - e.g., a 3-D (nonbase) cuboid for time, item, and location, summarized for all suppliers.
- Apex cuboid - the 0-D cuboid, which holds the highest level of summarization, is called the apex cuboid.
  - e.g., it is the total sales, or dollars sold, summarized over all four dimensions.
  - Typically, is denoted by all.

An Example of a Lattice of Cuboids

- A lattice of cuboids forming a data cube for the dimensions time, item, location, and supplier.
- Each cuboid represents a different degree of summarization.

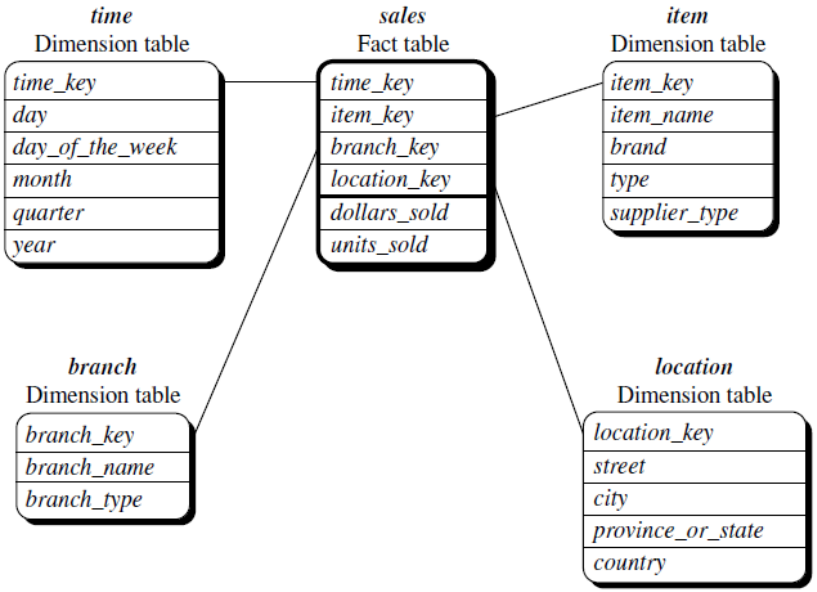


Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

Multidimensional Data Models - Star Schema

- The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.
- Data warehouse contains
  - A large central table containing the bulk of the data with no redundancy
  - A set of smaller attendant tables (dimension tables), one for each dimension.
- Star schema of sales data warehouse with time, item, branch, and location dimensions.
  - The schema contains a central fact table for sales that contains keys to each of the four dimensions, along with two measures: dollars sold and units sold.



Multidimensional Data Models - Star Schema

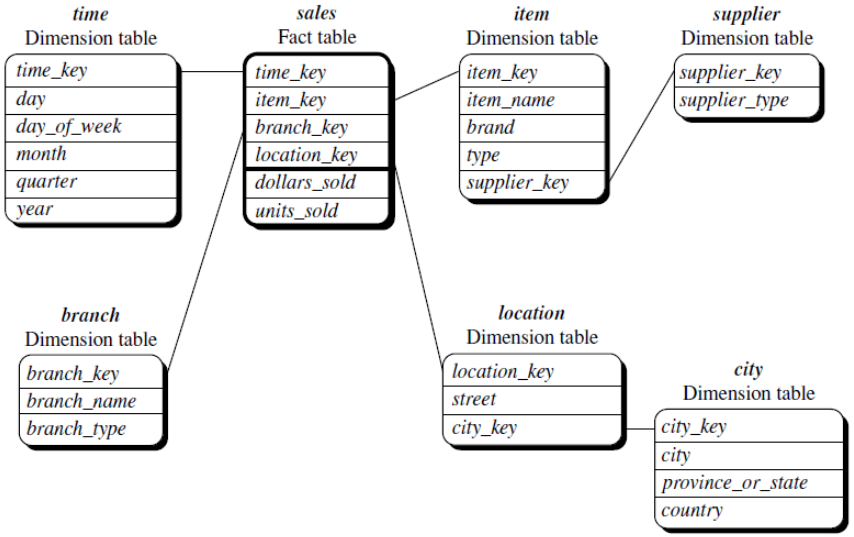
- Each dimension is represented by only one table, and each table contains a set of attributes.
  - e.g., the location dimension table contains the attribute set {location key, street, city, province or state, country}.
- This constraint may introduce some redundancy.
  - e.g., "Urbana" and "Chicago" are both cities in the state of Illinois, USA.
  - Entries for such cities in the location dimension table will create redundancy among the attributes province or state and country; that is, (...., Urbana, IL, USA) and (...., Chicago, IL, USA).
- The attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).

Multidimensional Data Models - Snowflake schema

- A variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables.
- The dimension tables of the snowflake model may be kept in normalized form to reduce redundancies.
- The snowflake structure can reduce the effectiveness of browsing, since more joins are needed to execute a query.
  - The system performance may be adversely impacted.
- Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

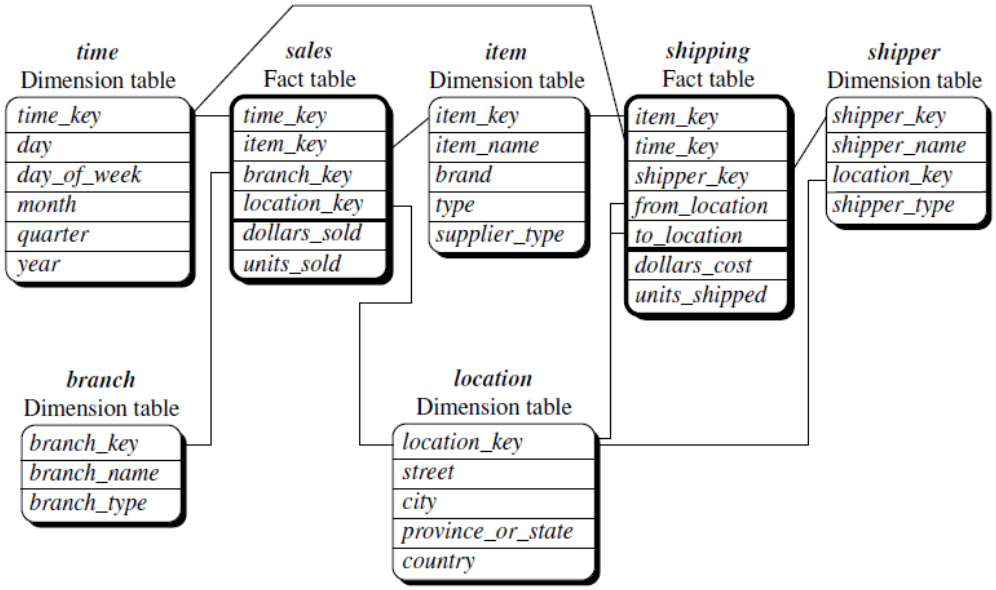
An Example of A Snowflake Schema

- The single dimension table for item in the star schema is normalized resulting in new item and supplier tables.
  - e.g., the item dimension table now contains the attributes item key, item name, brand, type, and supplier key, where supplier key is linked to the supplier dimension table, containing supplier key and supplier type information.
- The single dimension table for location in the star schema is normalized into two new tables: location and city.
- The city key in the new location table links to the city dimension.
- Further normalization can be performed on province or state and country.



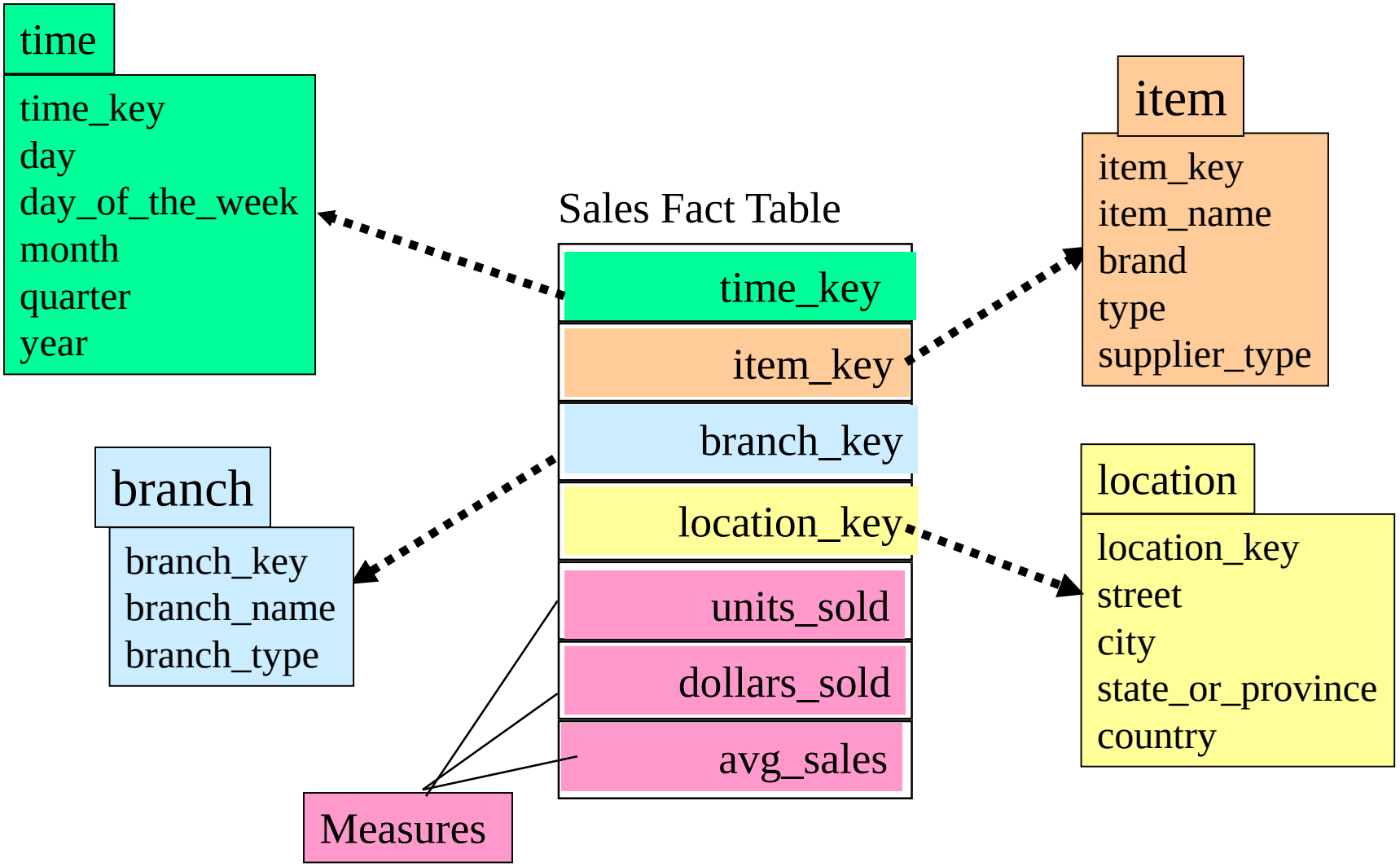
Multidimensional Data Models - Fact Constellation

- Sophisticated applications may require multiple fact tables to share dimension tables.
- Viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

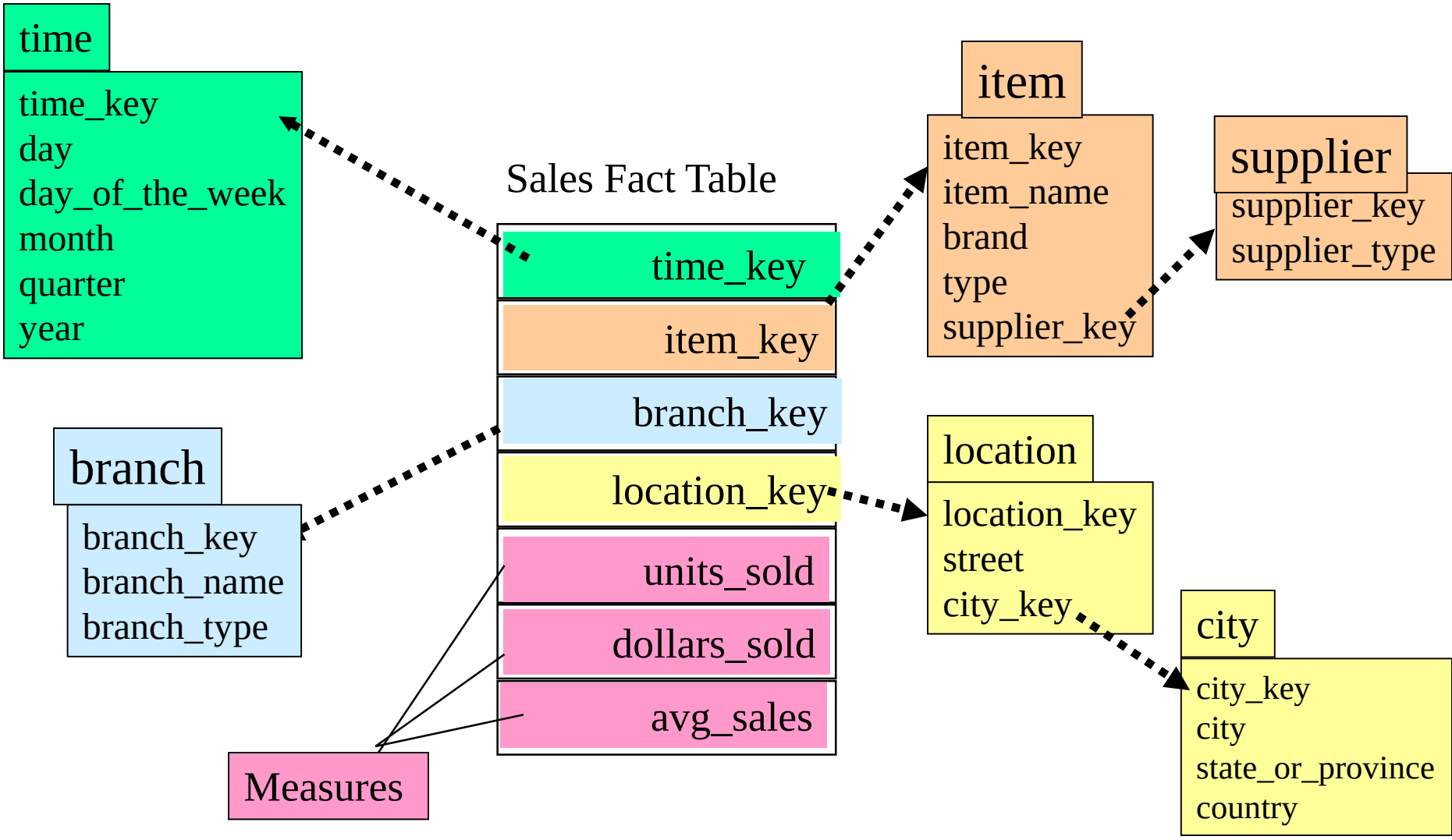


- This schema specifies two fact tables, sales and shipping.
- The sales table definition is identical to that of the star schema.
- The shipping table has five dimensions, or keys—item key, time key, shipper key, from location, and to location—and two measures—dollars cost and units shipped.
- A fact constellation schema allows dimension tables to be shared between fact tables.
  - e.g., the dimensions tables for time, item, and location are shared between the sales and shipping fact tables.

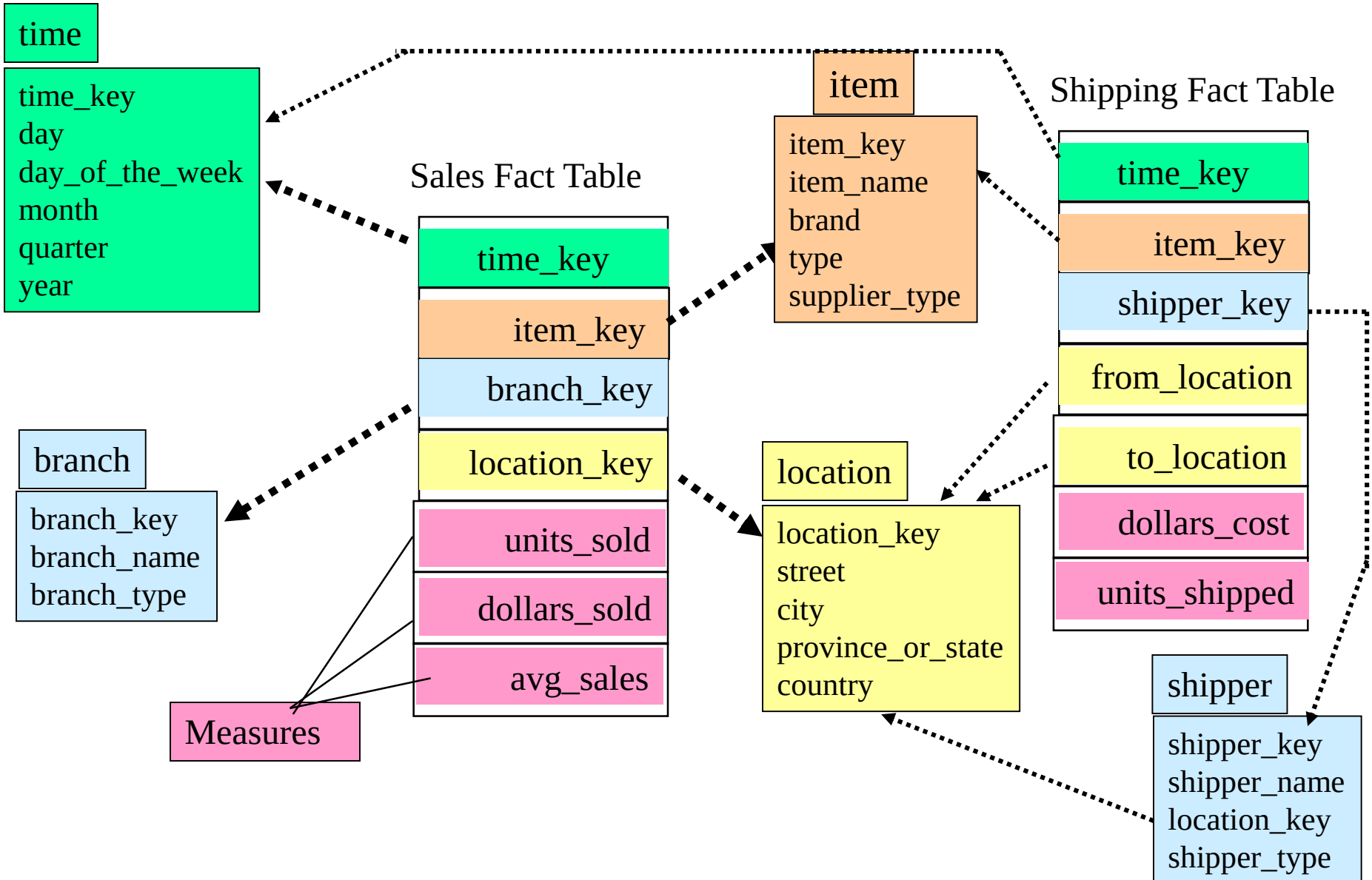
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation

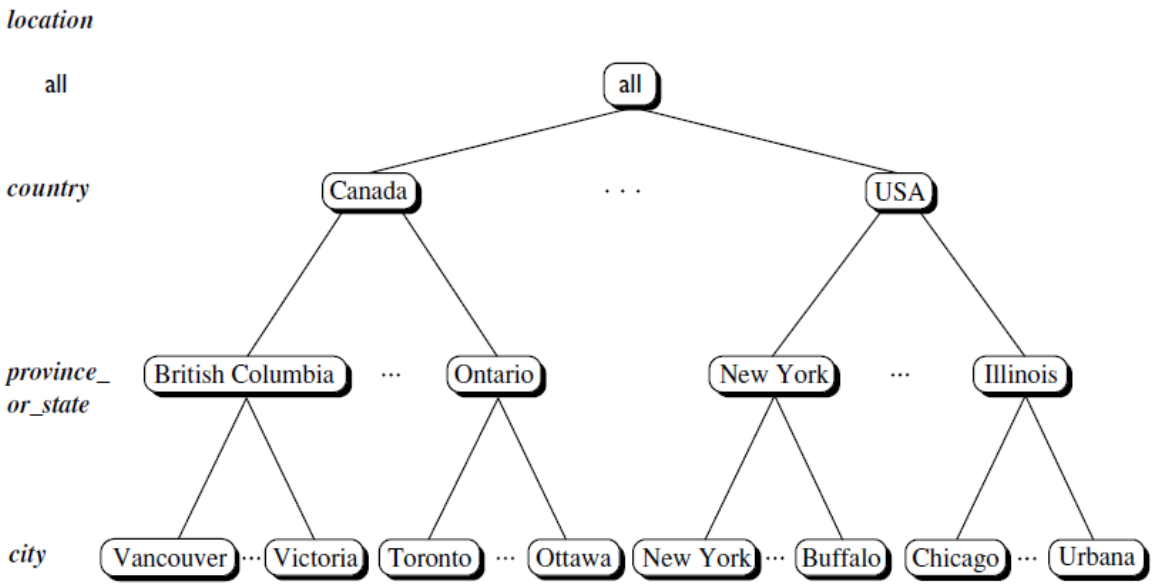


Data Warehouse vs. Data Mart

- A data warehouse
  - Collects information about subjects that span the entire organization, such as customers, items, sales, assets, an personnel, and thus its scope is enterprise-wide.
  - The fact constellation schema is commonly used, since it can model multiple, interrelated subjects.
- A data mart
  - A department subset of the data warehouse that focuses on selected subjects, and thus its scope is department-wide.
  - The star or snowflake schema is commonly used, since both are geared toward modeling single subjects.

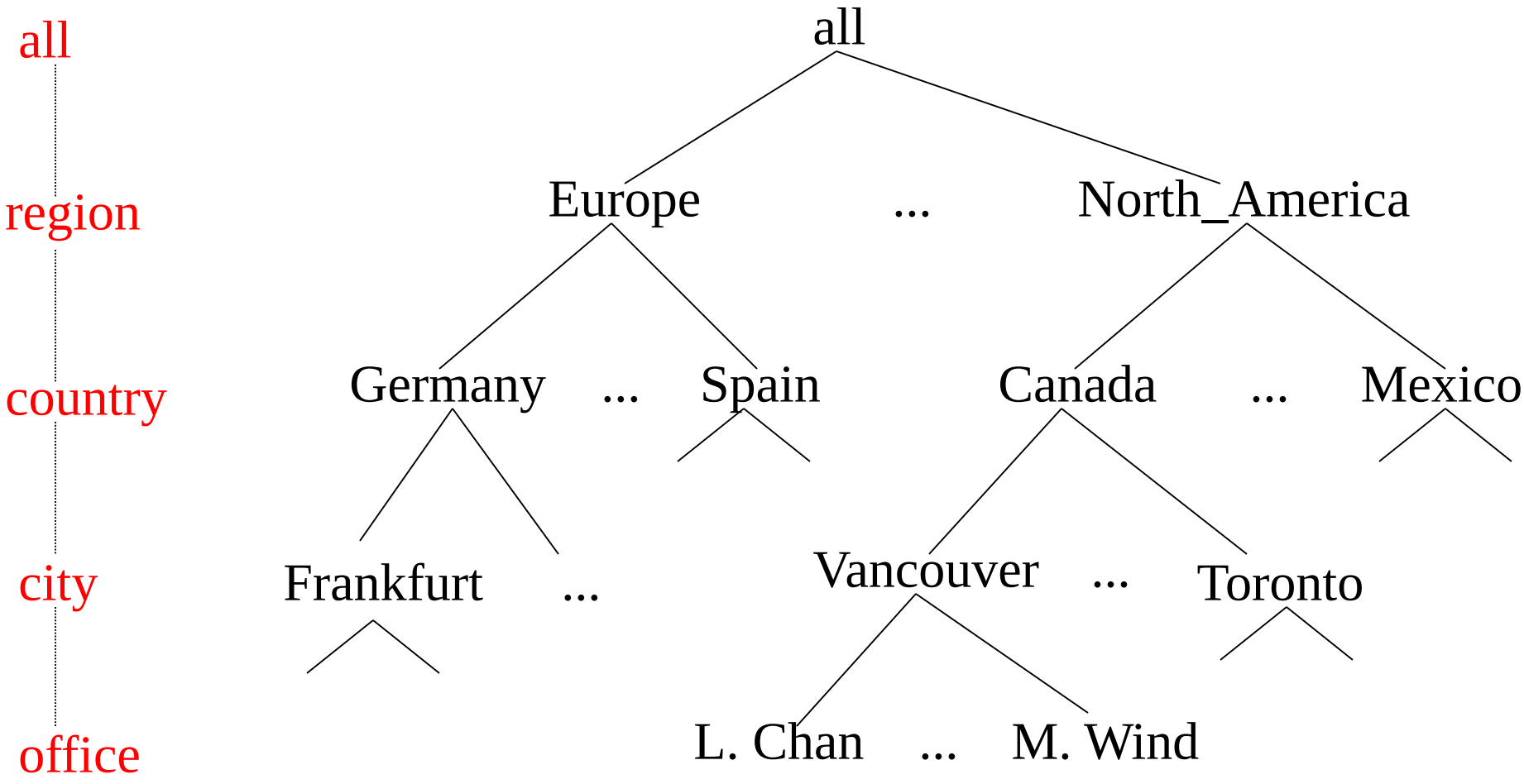
Dimensions: Concept Hierarchies

- A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.
- A concept hierarchy for the dimension location.



- Each city can be mapped to the province or state to which it belongs.
- The provinces and states can be mapped to the country to which they belong.
- These mappings form a concept hierarchy for the dimension location, mapping a set of low-level concepts (i.e., cities) to higher-level, more general concepts (i.e., countries).

A Concept Hierarchy: Dimension (location)



Measures: Their Categorization and Computation

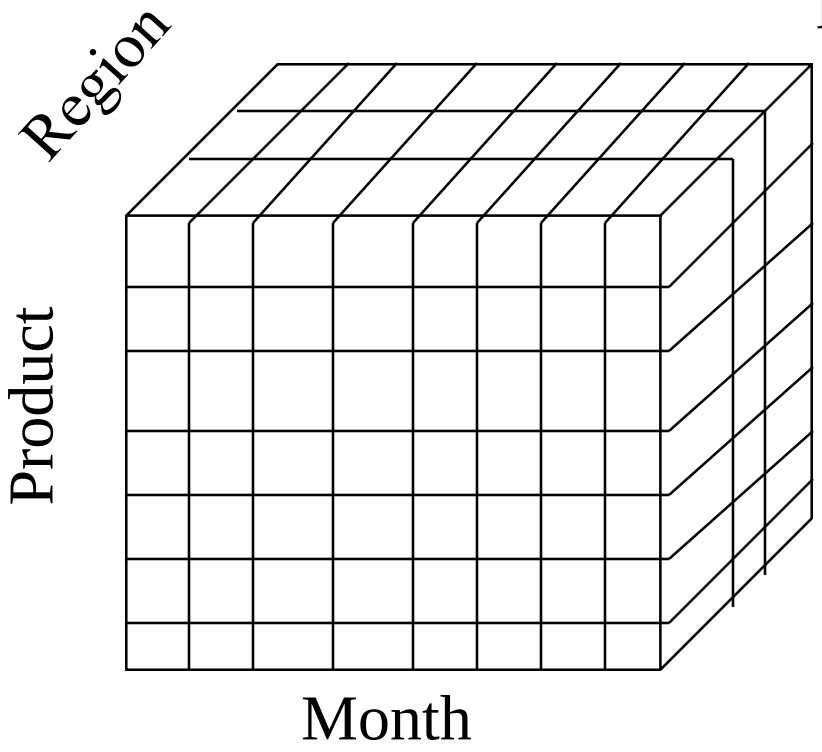
- A data cube measure is a numeric function that can be evaluated at each point in the data cube space.
- A measure value is computed for a given point by aggregating the data corresponding to the respective dimension–value pairs defining the given point.
- Measures can be organized into three categories based on the kind of aggregate functions used.
  - Distributive
  - Algebraic
  - Holistic

Data Cube Measures: Three Categories

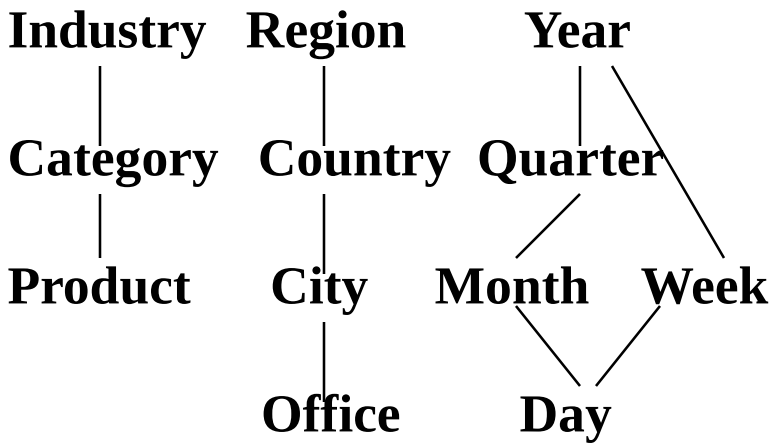
- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg(), min_N(), standard_deviation()
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

Multidimensional Data

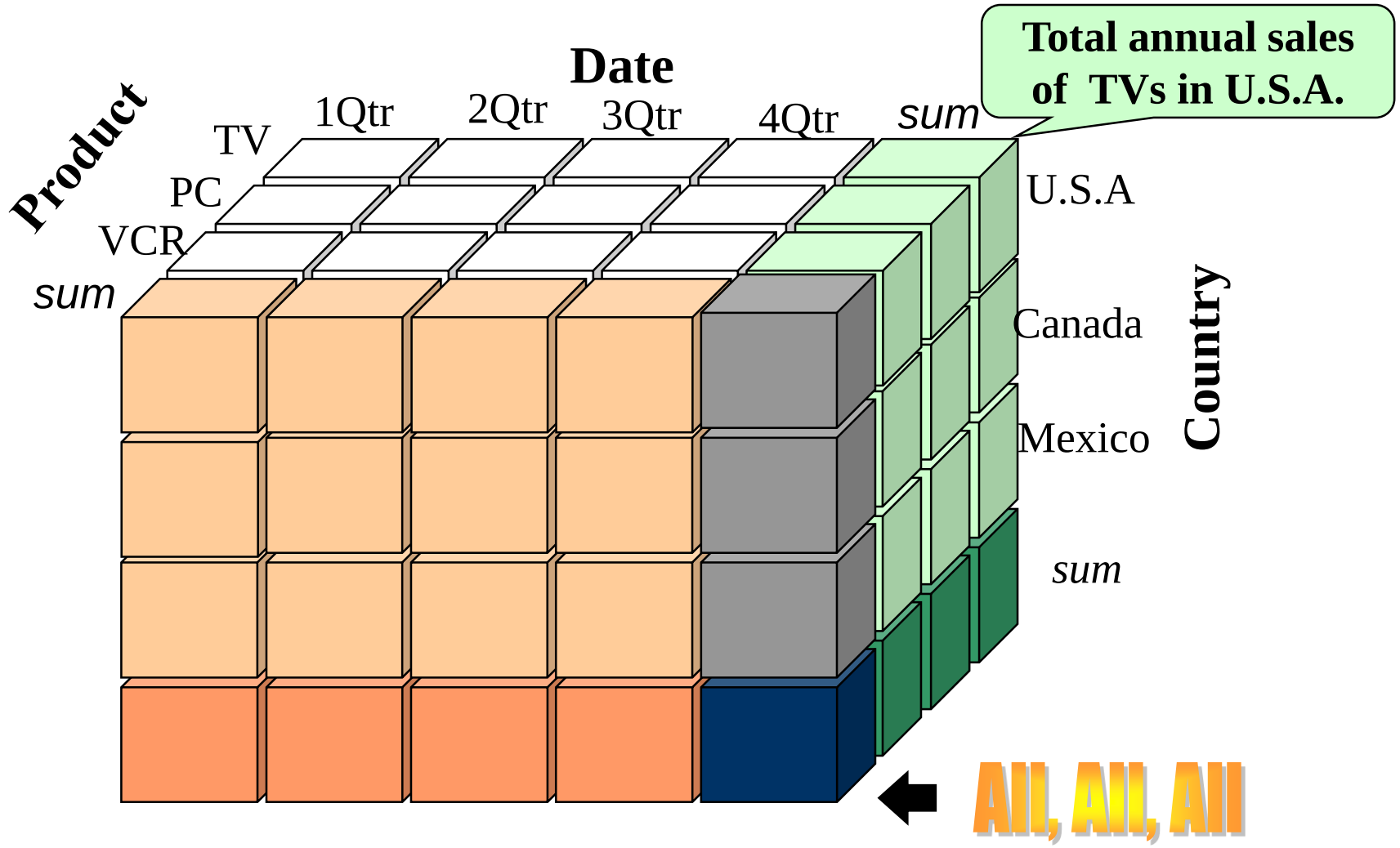
- Sales volume as a function of product, month, and region



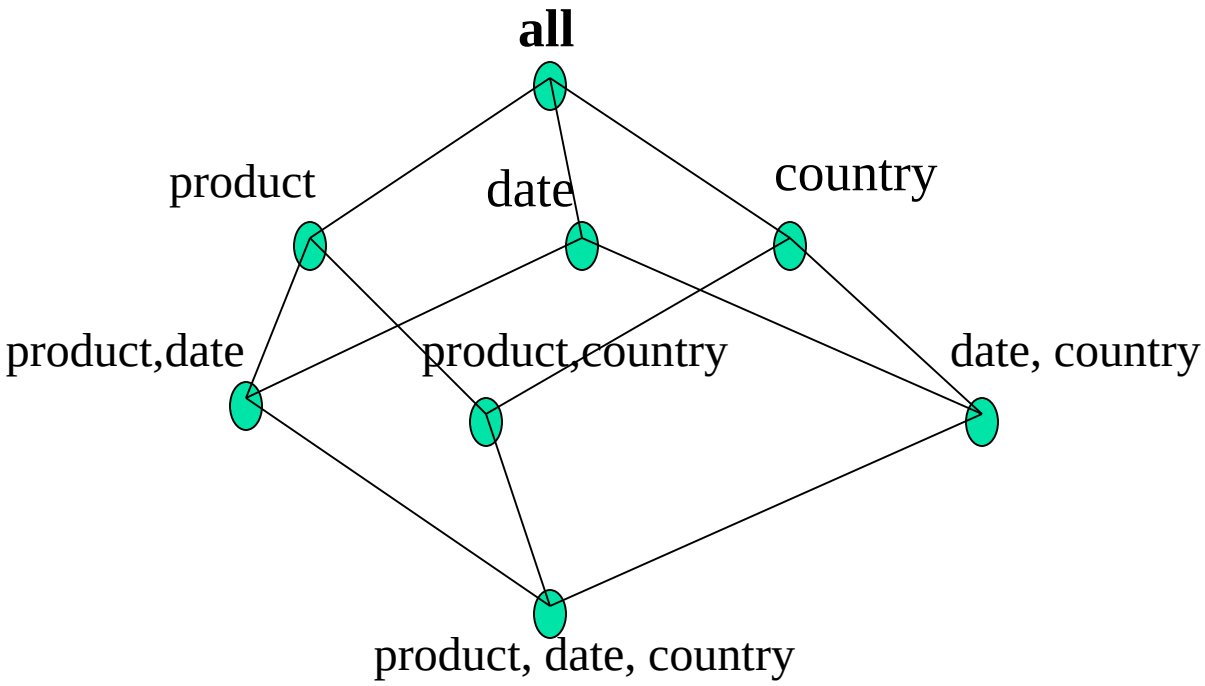
Dimensions: *Product, Location, Time*
Hierarchical summarization paths



A Sample Data Cube



Cuboids Corresponding to the Cube



0-D (*apex*) cuboid

1-D cuboids

2-D cuboids

3-D (*base*) cuboid

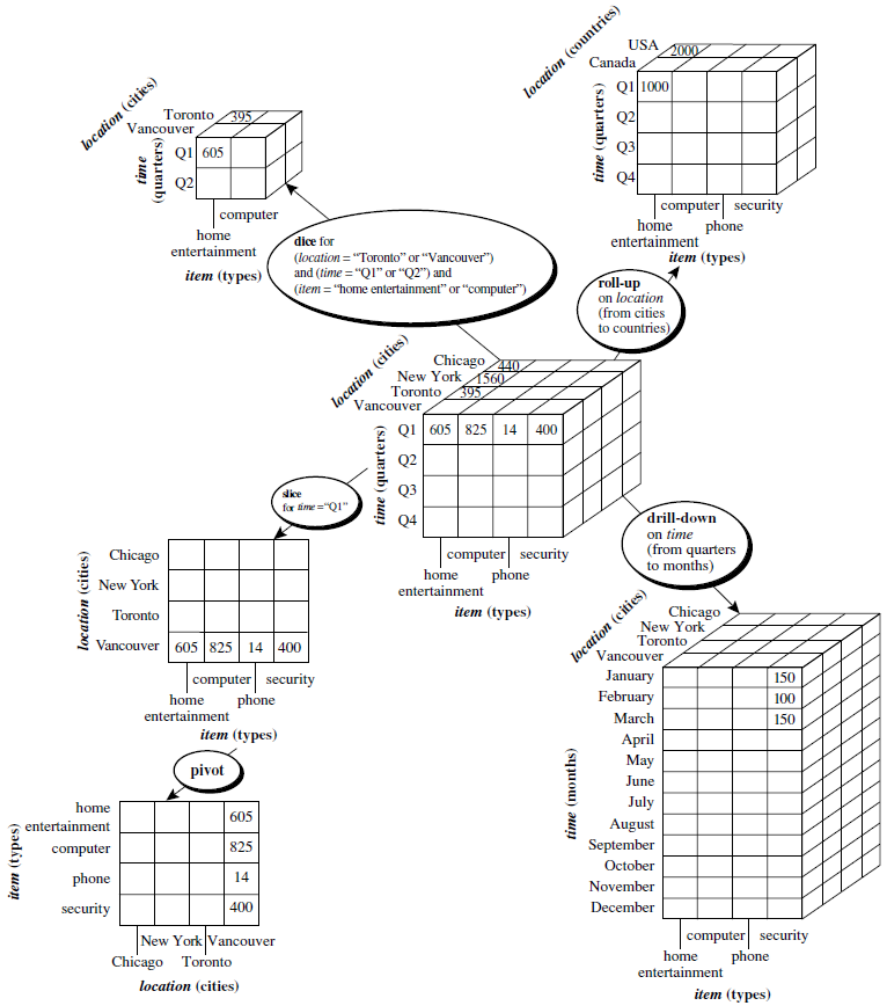
OLAP Operations

- In the multidimensional model, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies.
- This organization provides users with the flexibility to view data from different perspectives.
- A number of OLAP data cube operations exist to materialize these different views, allowing interactive querying and analysis of the data at hand.
- Hence, OLAP provides a user-friendly environment for interactive data analysis.

OLAP Operations

- Roll-up (also called the drill-up operation)
  - Performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction.
  - Roll-up may be performed by removing, say, the time dimension, resulting in an aggregation of the total sales by location, rather than by location and by time.
- Drill-down (the reverse of roll-up)
  - It navigates from less detailed data to more detailed data.
  - Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.
- Slice
  - Performs a selection on one dimension of the given cube, resulting in a subcube.
- Dice
  - Perform a selection on two or more dimensions, resulting in a subcube.
- Pivot (also called rotate)
  - A visualization operation that rotates the data axes in view to provide an alternative data presentation.

OLAP Operations

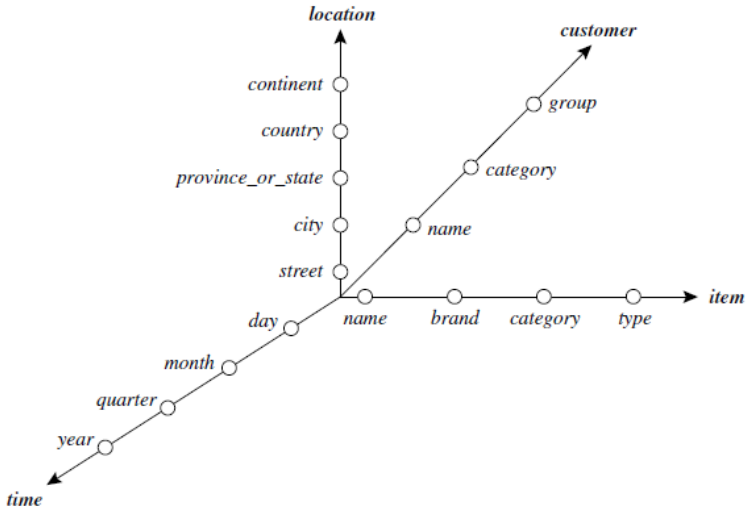


OLAP Systems vs. Statistical Databases

- A statistical database is a database system that is designed to support statistical applications.
- Many OLAP systems' characteristics (e.g., the use of a multidimensional data model and concept hierarchies, the association of measures with dimensions, and the notions of roll-up and drill-down) also exist in earlier work on statistical databases (SDBs).
- Similarities between the two types of systems are rarely discussed, mainly due to differences in terminology and application domains.
- OLAP and SDB systems, however, have distinguishing differences.
  - While SDBs tend to focus on socio-economic applications, OLAP has been targeted for business applications.
  - Privacy issues regarding concept hierarchies are a major concern for SDBs.
    - e.g., given summarized socioeconomic data, it is controversial to allow users to view the corresponding low-level data.
- Unlike SDBs, OLAP systems are designed for efficiently handling huge amounts of data.

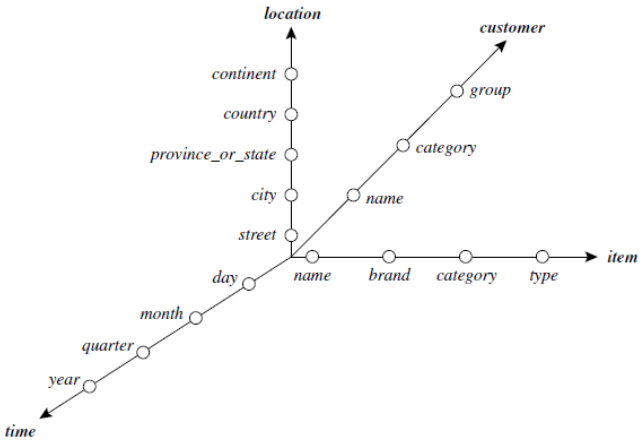
A Starnet Query Model for Querying Multidimensional Databases

- The querying of multidimensional databases can be based on a starnet model, which consists of radial lines emanating from a central point, where each line represents a concept hierarchy for a dimension.
- Each abstraction level in the hierarchy is called a footprint.
- These represent the granularities available for use by OLAP operations such as drill-down and roll-up.

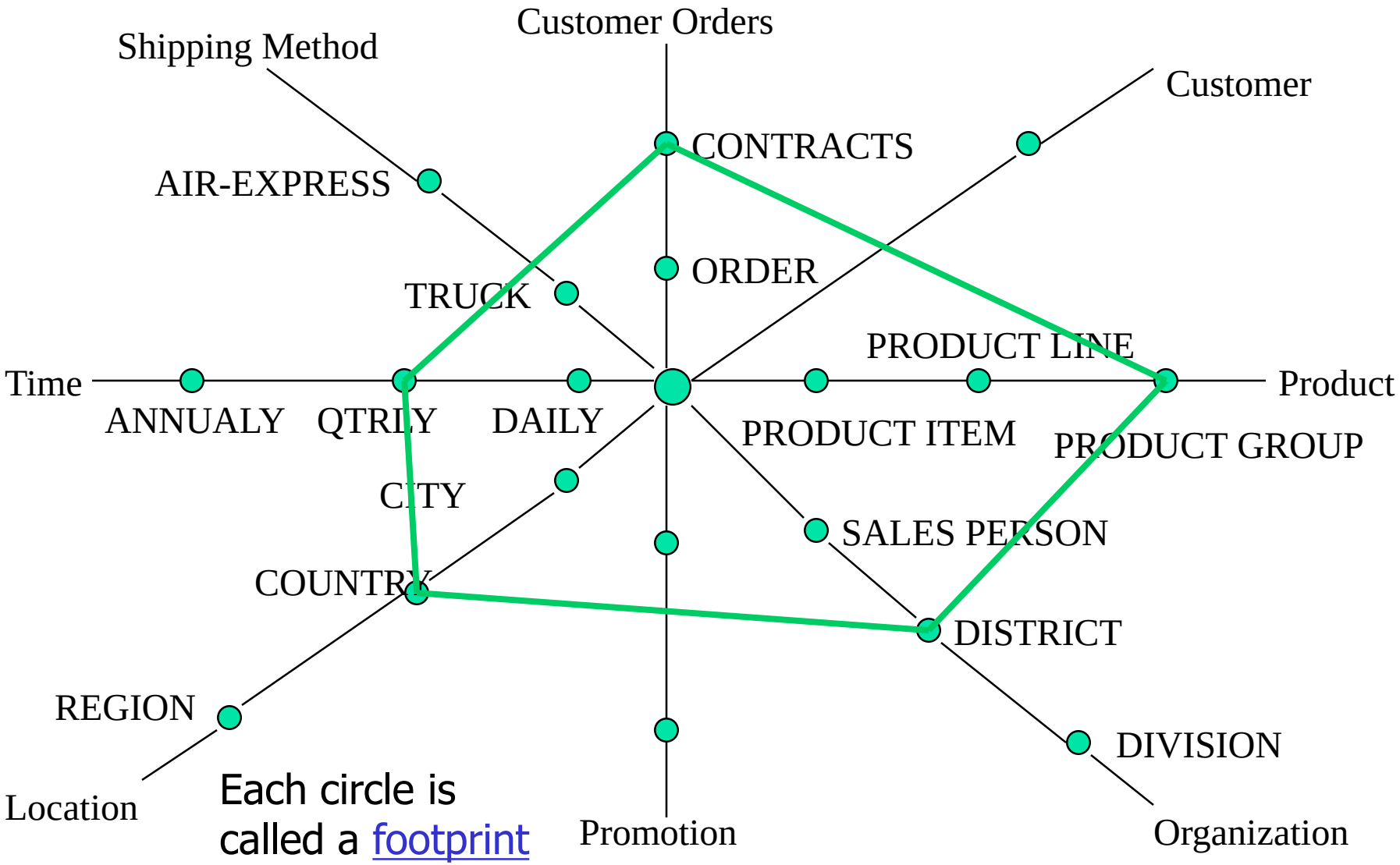


A Starnet Query Model for Querying Multidimensional Databases

- This starnet consists of four radial lines, representing concept hierarchies for the dimensions location, customer, item, and time, respectively.
- Each line consists of footprints representing abstraction levels of the dimension.
  - e.g., the time line has four footprints: "day," "month," "quarter," and "year."
- Concept hierarchies can be used to generalize data by replacing low-level values by higher-level abstractions, or to specialize data by replacing higher-level abstractions with lower-level values.

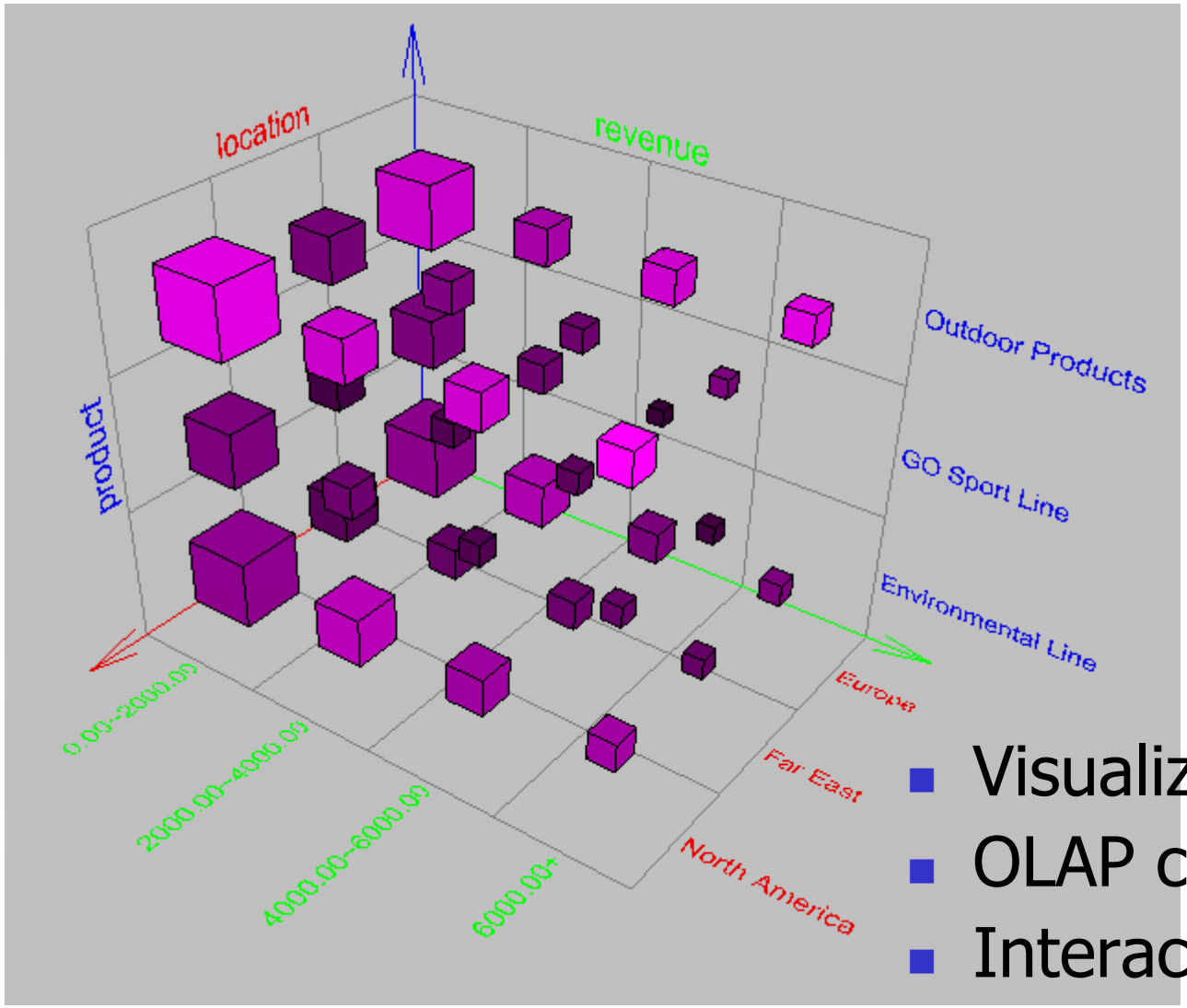


A Star-Net Query Model



Each circle is called a footprint

Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage 
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

Data Warehouse Design and Usage

- What goes into a data warehouse design?
- How are data warehouses used?
- How do data warehousing and OLAP relate to data mining?

Gains from having a data warehouse

- Provide a competitive advantage by presenting relevant information from which to measure performance and make critical adjustments to help win over competitors.
- Enhance business productivity because it is able to quickly and efficiently gather information that accurately describes the organization.
- Facilitates customer relationship management because it provides a consistent view of customers and items across all lines of business, all departments, and all markets.
- Bring about cost reduction by tracking trends, patterns, and exceptions over long periods in a consistent and reliable manner.

Data warehouse design

- To design an effective data warehouse we need to understand and analyze business needs and construct a business analysis framework.
- The construction of a large and complex information system can be viewed as the construction of a large and complex building, for which the owner, architect, and builder have different views.
- These views are combined to form a complex framework that represents the top-down, business-driven, or owner's perspective, as well as the bottom-up, builder-driven, or implementor's view of the information system.
- Four different views regarding a data warehouse design must be considered
  - top-down view
  - data source view
  - data warehouse view
  - business query view.

Four different views regarding a data warehouse design

- Top-down view – allows the selection of the relevant information necessary for the data warehouse.
  - This information matches current and future business needs.
- Data source view – exposes the information being captured, stored, and managed by operational systems.
  - This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables.
- Data warehouse view – includes fact tables and dimension tables.
  - It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- Business query view - is the data perspective in the data warehouse from the end-user's viewpoint.

Data Warehouse Design Process

- A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both.
- Top-down approach – starts with overall design and planning.
  - Useful in cases where the technology is mature and well known, and where the business problems that must be solved are clear and well understood.
- Bottom-up approach – starts with experiments and prototypes.
  - Useful in the early stage of business modeling and technology development.
  - Allows to move forward at considerably less expense and to evaluate the technological benefits before making significant commitments.
- Combined approach - exploits the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach.

Data Warehouse Design Process

- From the software engineering point of view, large software systems can be developed using one of two methodologies:
  - Waterfall method - performs a structured and systematic analysis at each step before proceeding to the next.
  - Spiral method - involves the rapid generation of increasingly functional systems, with short intervals between successive releases.
    - Good choice for data marts, because the turnaround time is short, modifications can be done quickly, and new designs and technologies can be adapted in a timely manner.

Steps of Data Warehouse Design Process

- Choose a business process to model (e.g., orders, invoices, shipments, inventory, account administration, sales, or the general ledger).
  - If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed.
  - If the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.
- Choose the business process grain, which is the fundamental, atomic level of data to be represented in the fact table for this process (e.g., individual transactions, individual daily snapshots, and so on).
- Choose the dimensions that will apply to each fact table record.
  - Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
- Choose the measures that will populate each fact table record.
  - Typical measures are numeric additive quantities like dollars sold and units sold.

Data Warehouse Design Process (Recap.)

- Top-down, bottom-up approaches or a combination of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- From software engineering point of view
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the *grain* (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record

Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation 
- Data Generalization by Attribute-Oriented Induction
- Summary

Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels?

$$T = \prod_{i=1}^{n} (L_i + 1)$$

- Materialization of data cube
  - Materialize every (cuboid) (**full materialization**), none (**no materialization**), or some (**partial materialization**)
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.

The "Compute Cube" Operator

- Cube definition and computation in DMQL

  define cube sales [item, city, year]: sum (sales_in_dollars)

  compute cube sales

- Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

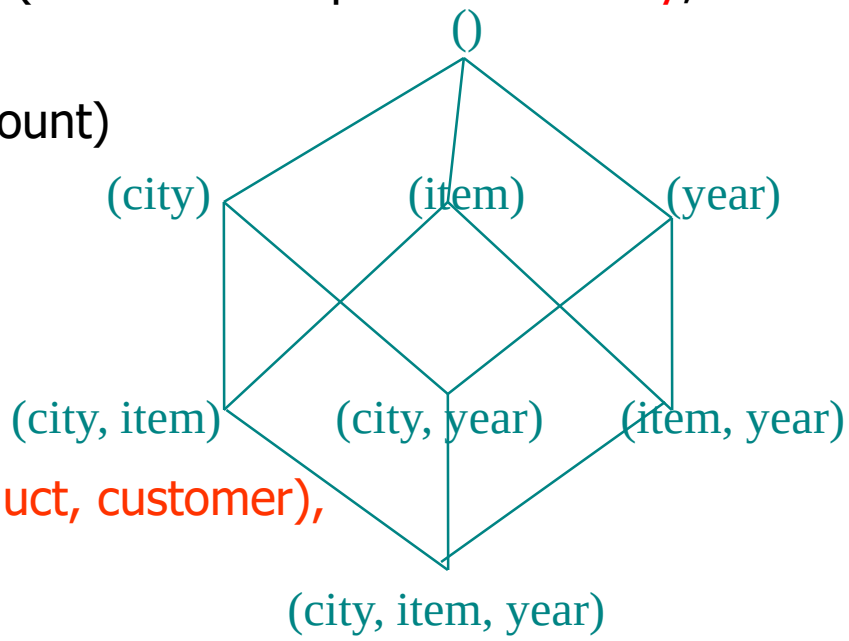
  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

- Need compute the following Group-Bys

  (date, product, customer),
  (date,product),(date, customer), (product, customer),
  (date), (product), (customer)
  ()



Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The i-th bit is set if the i-th row of the base table has the value for the indexed column
- Not suitable for high cardinality domains
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region

RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

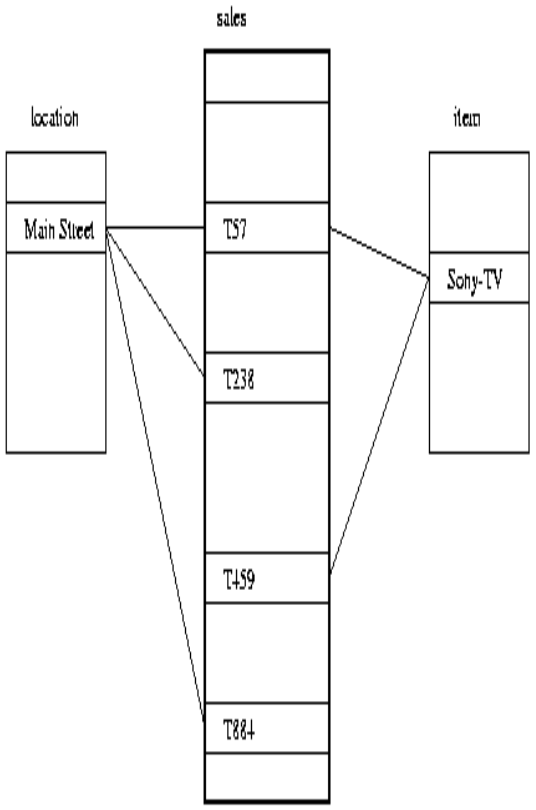
Indexing OLAP Data: Join Indices

- Traditional indexing maps the value in a given column to a list of rows having that value.
- In contrast, join indexing registers the joinable rows of two relations from a relational database.
- If two relations R(RID, A) and S(B, SID) join on the attributes A and B,
  - The join index record contains the pair (RID, SID) where

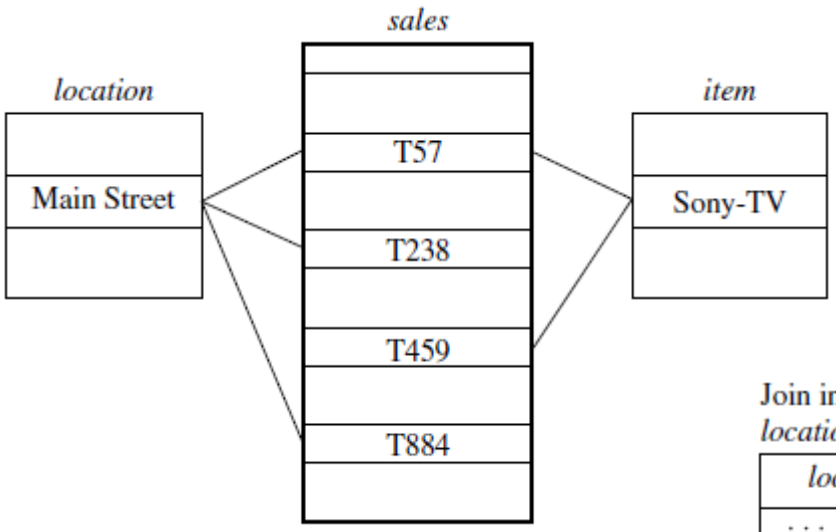
    RID and SID are record identifiers from the R and S relations.
- The join index records can identify joinable tuples without performing costly join operations.
- Join indexing is especially useful for maintaining the relationship between a foreign key and its matching primary keys, from the joinable relation.

Indexing OLAP Data: Join Indices

- Join index: JI(R-id, S-id) where R (R-id, …) ▷◁ S (S-id, …)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join
- In data warehouses, join index relates the values of the dimensions of a star schema to rows in the fact table.
  - E.g. fact table: Sales and two dimensions city and product
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions



An Example of Join Indices



Join index table for *location/sales*

location	*sales_key*
. . .	. . .
Main Street	T57
Main Street	T238
Main Street	T884
. . .	. . .

Join index table for *item/sales*

item	*sales_key*
. . .	. . .
Sony-TV	T57
Sony-TV	T459
. . .	. . .

Join index table linking *location* and *item* to *sales*

location	*item*	*sales_key*
. . .	. . .	. . .
Main Street	Sony-TV	T57
. . .	. . .	. . .

Efficient Processing OLAP Queries

- Determine which operations should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection on a materialized cuboid
- Determine which materialized cuboid(s) should be selected for OLAP op.
  - Let the query to be processed be on {brand, province_or_state} with the condition *"year = 2004"*, and there are 4 materialized cuboids available:

    1) {*year, item_name, city*}

    2) {*year, brand, country*}

    3) {*year, brand, province_or_state*}

    4) {*item_name, province_or_state*} where *year = 2004*

    Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability
- Multidimensional OLAP (MOLAP)
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

Attribute-Oriented Induction

- Proposed in 1989 (KDD '89 workshop)
- Not confined to categorical data nor particular measures
- How it is done?
  - Collect the task-relevant data (*initial relation*) using a relational database query
  - Perform generalization by attribute removal or attribute generalization
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
  - Interaction with users for knowledge presentation

Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the University database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,

  Select * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa)

  from student

  where student_status in {“Msc”, “MBA”, “PhD” }

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

Class Characterization: An Example

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

Gender \ Birth_Region	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

Basic Principles of Attribute-Oriented Induction

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*
- Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*'s higher level concepts are expressed in terms of other attributes
- Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize *A*
- Attribute-threshold control: typical 2-8, specified/default
- Generalized relation threshold control: control the final relation/rule size

Attribute-Oriented Induction: Basic Algorithm

- InitialRel: Query processing of task-relevant data, deriving the *initial relation*.
- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- PrimeGen: Based on the PreGen plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Presentation of Generalized Results

- Generalized relation:
  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.
- Cross tabulation:
  - Mapping results into cross tabulation form (similar to contingency tables).
  - Visualization techniques:
  - Pie charts, bar charts, curves, cubes, and other visual forms.
- Quantitative characteristic rules:
  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

$$grad(x) \wedge male(x) \Rightarrow$$
$$birth\_region(x) = \text{"}Canada\text{"}[t:53\%] \vee birth\_region(x) = \text{"}foreign\text{"}[t:47\%].$$

Mining Class Comparisons

- Comparison: Comparing two or more classes
- Method:
  - Partition the set of relevant data into the target class and the contrasting class(es)
  - Generalize both classes to the same high level concepts
  - Compare tuples with the same high level descriptions
  - Present for every tuple its description and two measures
    - support - distribution within single class
    - comparison - distribution between classes
  - Highlight the tuples with strong discriminant features
- Relevance Analysis:
  - Find attributes (features) which best distinguish different classes

Concept Description vs. Cube-Based OLAP

- **Similarity**:
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing
- **Differences**:
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms

Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

Summary

- Data warehousing: A multi-dimensional model of a data warehouse
  - A data cube consists of *dimensions* & *measures*
  - Star schema, snowflake schema, fact constellations
  - OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data Warehouse Architecture, Design, and Usage
  - Multi-tiered architecture
  - Business analysis design framework
  - Information processing, analytical processing, data mining, OLAM (Online Analytical Mining)
- Implementation: Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
  - OLAP servers: ROLAP, MOLAP, HOLAP
- Data generalization: Attribute-oriented induction

References (I)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
- E. F. Codd, S. B. Codd, and C. T. Salley. Beyond decision support. Computer World, 27, July 1993.
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- A. Gupta and I. S. Mumick. Materialized Views: Techniques, Implementations, and Applications. MIT Press, 1999.
- J. Han. Towards on-line analytical mining in large databases. *ACM SIGMOD Record*, 27:97-107, 1998.
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD'96
- J. Hellerstein, P. Haas, and H. Wang. Online aggregation. SIGMOD'97

References (II)

- C. Imhoff, N. Galemmo, and J. G. Geiger. Mastering Data Warehouse Design: Relational and Dimensional Techniques. John Wiley, 2003
- W. H. Inmon. Building the Data Warehouse. John Wiley, 1996
- R. Kimball and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2ed. John Wiley, 2002
- P. O'Neil and G. Graefe. Multi-table joins through bitmapped join indices. *SIGMOD Record*, 24:8–11, Sept. 1995.
- P. O'Neil and D. Quass. Improved query performance with variant indexes. SIGMOD'97
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998
- S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. ICDE'94
- A. Shoshani. OLAP and statistical databases: Similarities and differences. PODS'00.
- D. Srivastava, S. Dar, H. V. Jagadish, and A. V. Levy. Answering queries with aggregation using views. *VLDB'96*
- P. Valduriez. Join indices. ACM Trans. Database Systems, 12:218-246, 1987.
- J. Widom. Research problems in data warehousing. CIKM'95
- K. Wu, E. Otoo, and A. Shoshani, Optimal Bitmap Indices with Efficient Compression, ACM Trans. on Database Systems (TODS), 31(1): 1-38, 2006