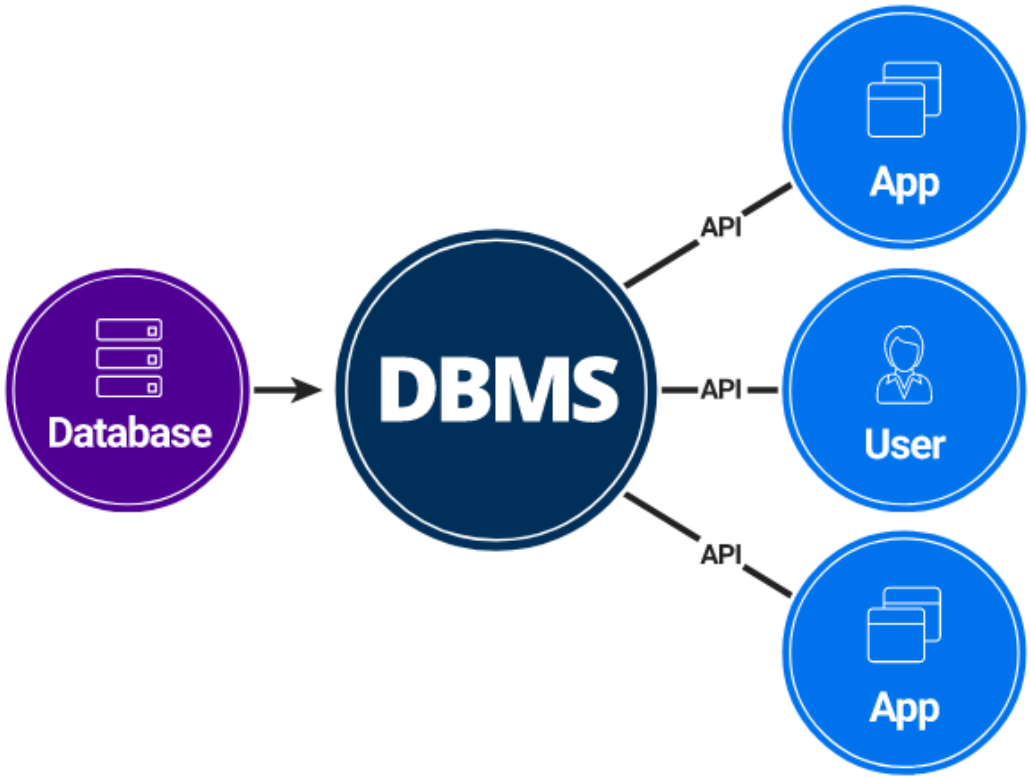
Part 5: Working with data - Database

Dr. Nguyen Quang Huy

June 16, 2021

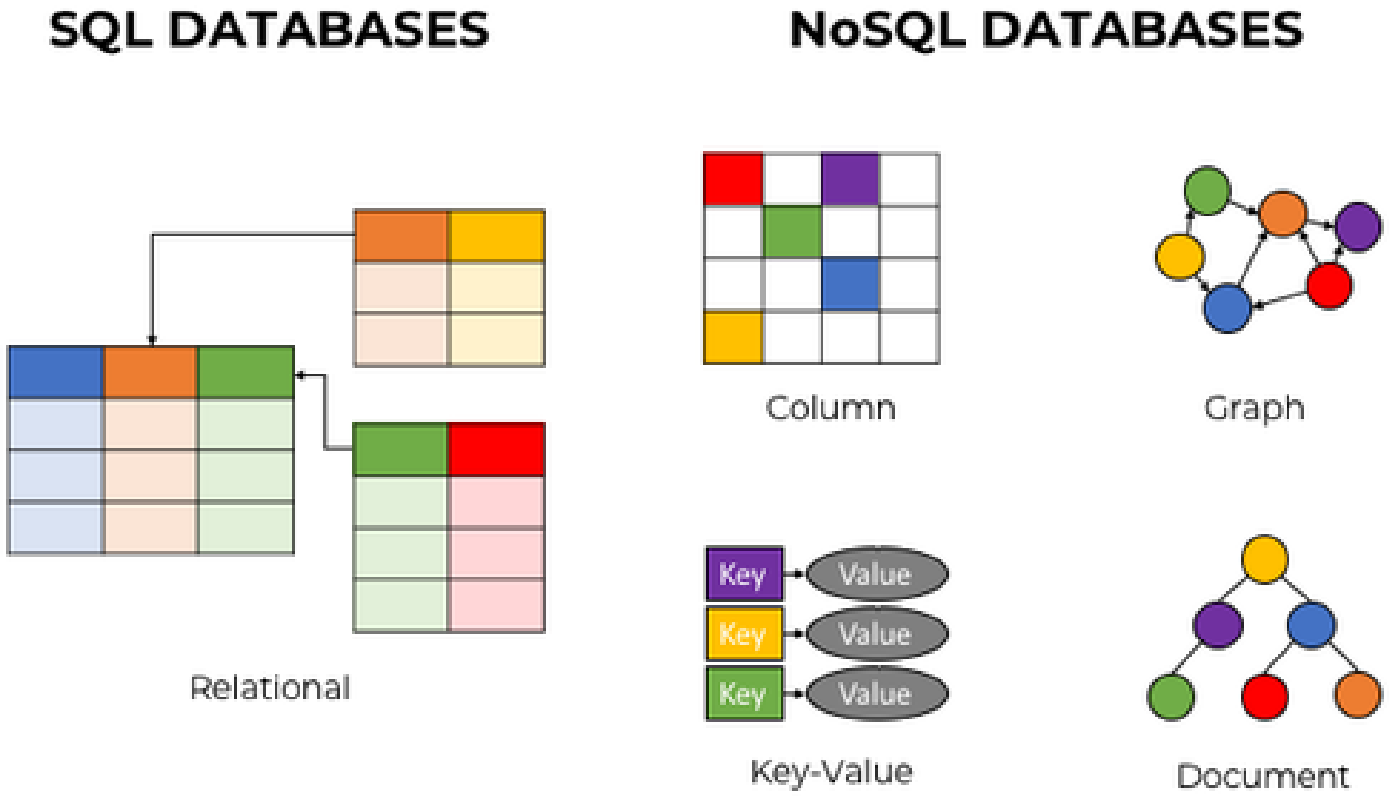
What is a database

- A **database** is a collection of data stored in a format that can be *easily accessed*
- To manage databases, we use a software application called **database management system (DBMS)**



Database management system

- Database management systems are classified into 2 categories: relational DBMS and non-relational DBMS



Structured Query Language (SQL)

- SQL or SEQUEL is the language that we use to work with relational database management system (RDBMS)
- SQL stands for **S**tructured **Q**uery **L**anguage which was developed at IBM and initially called SEQUEL (**S**tructured **E**nglish **Que**ry **L**anguage)
- There are many RDBMS, but the most popular ones are
  - MySQL: Open source database.
  - SQL Server: developed by Microsoft
  - Oracle database
- In this section, we will learn **MySQL**.
- All of these RDBMS are similar and based on the standard SQL specification.

Install MySQL

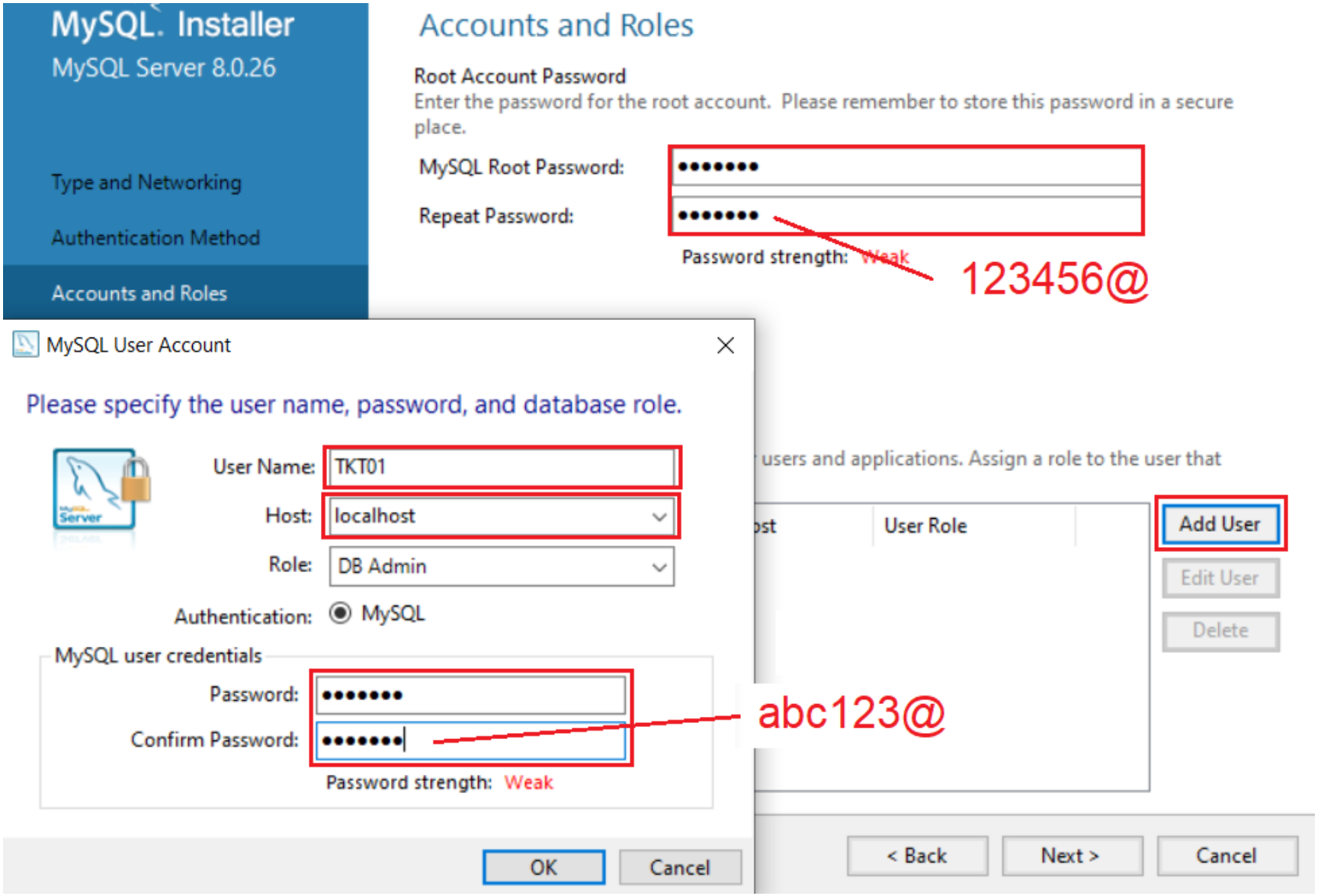
- MySQL community server: a free version of the world's most popular open source database that is supported by an active community of open source developers.
- MySQL Workbench: a unified visual tool for database architects and developers.
- To download MySQL community server and MySQL workbench, use the following links

https://dev.mysql.com/downloads/windows/installer/8.0.html

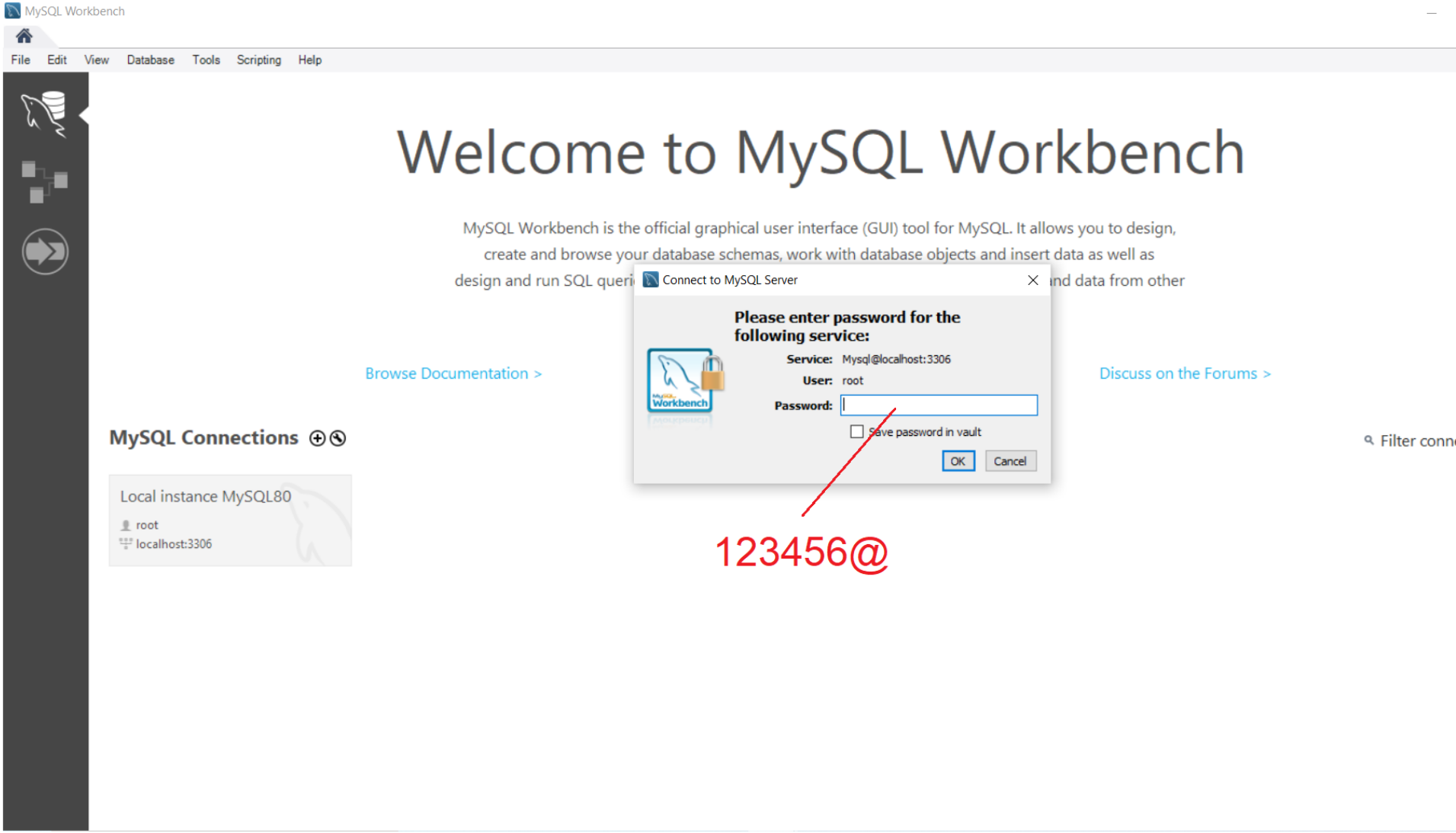
- To install MySQL community server and MySQL workbench to your computer, using the following guideline

https://www.thegioididong.com/game-app/huong-dan-cach-tai-cai-dat-mysql-ban-moi-nhat-chi-tiet-tung-1299084

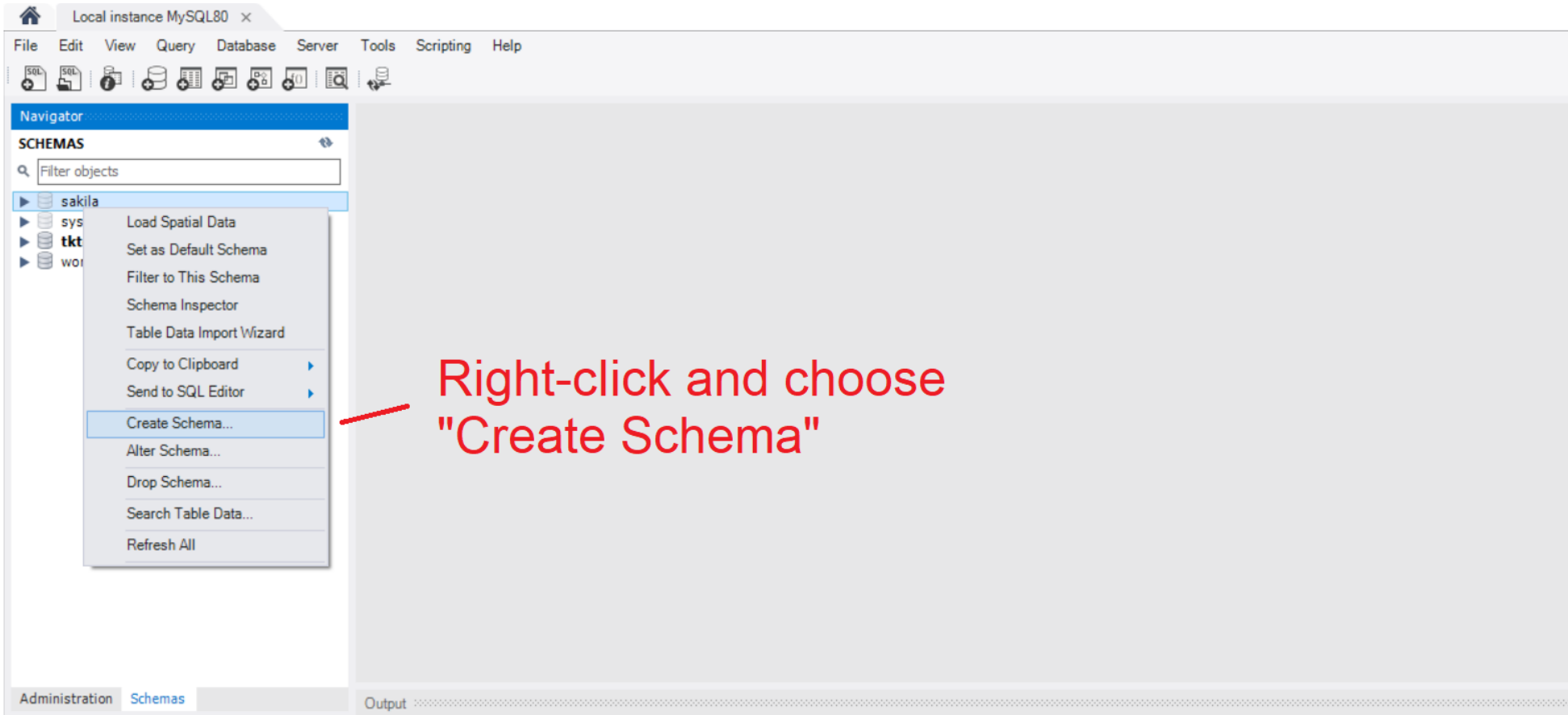
Install MySQL



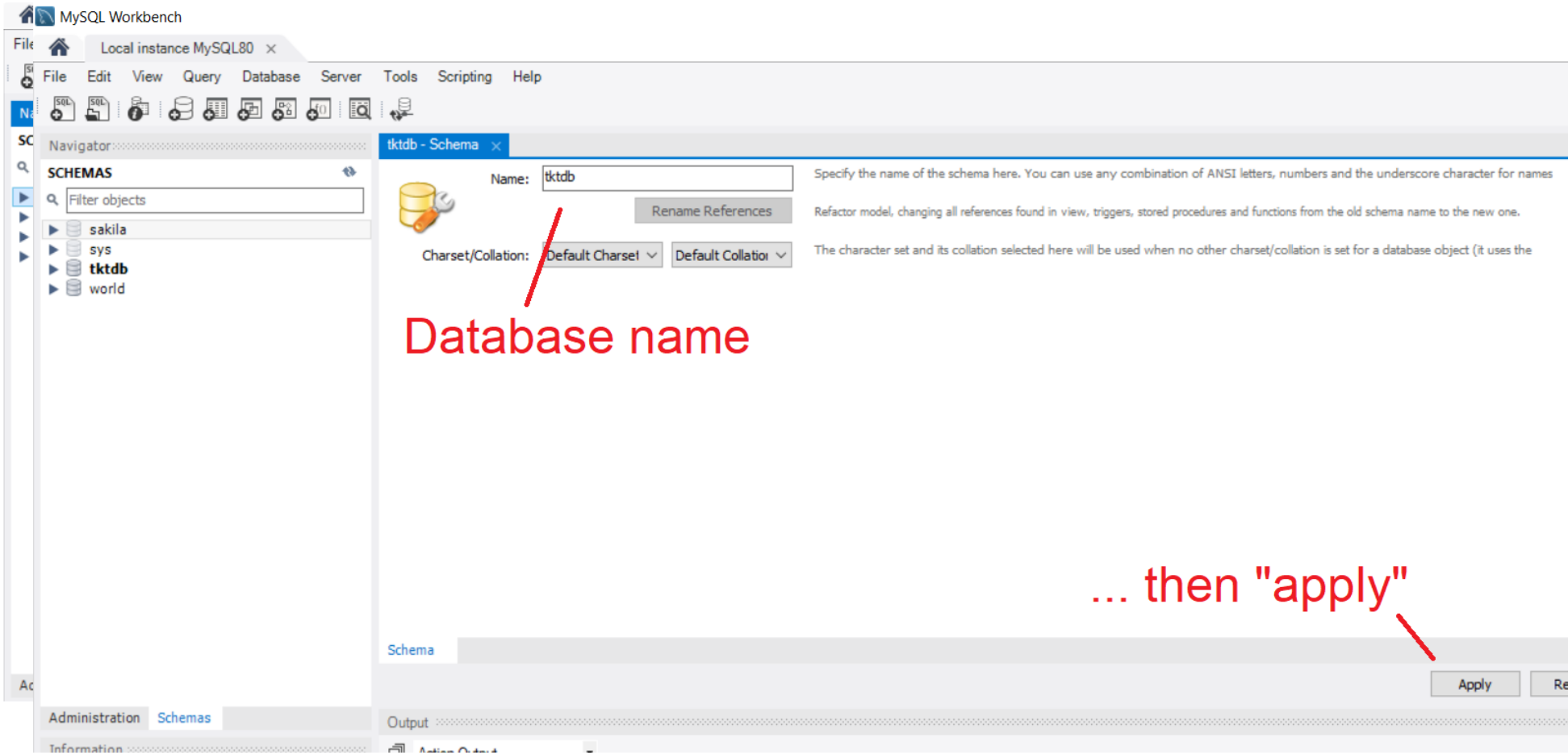
Install MySQL



Create a new database

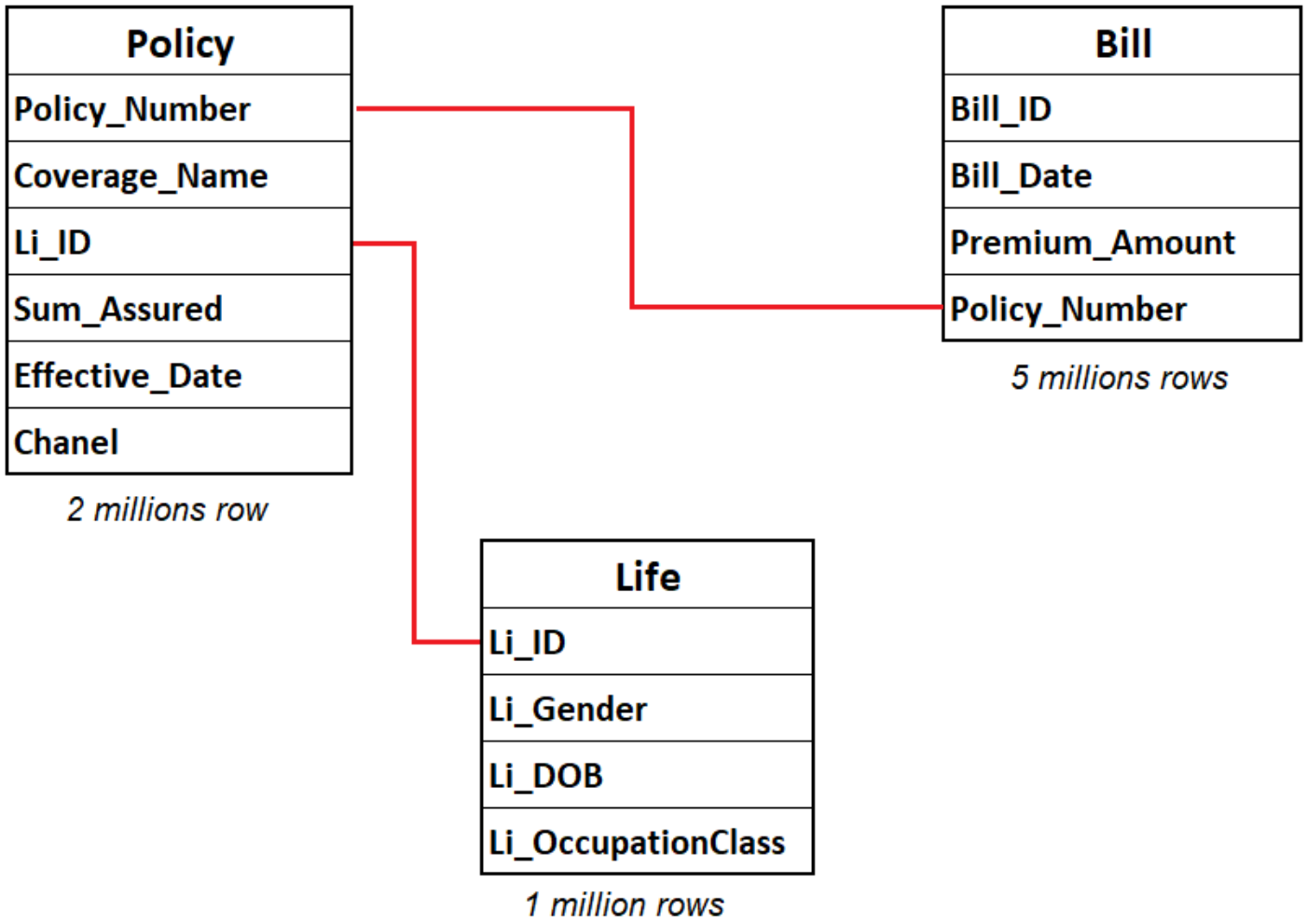


Create a new database



Create a new database

Our database



Understanding our database

Policy table: thông tin về hợp đồng bảo hiểm

- *Policy_Number*: Số hợp đồng, hợp đồng là contract giữa người mua bảo hiểm và công ty bảo hiểm
- *Coverage_Name*: Mã sản phẩm bảo hiểm, ví dụ như mã "TL1" là sản phẩm bảo hiểm tai nạn cá nhân, "TL2" là sản phẩm bảo hiểm sức khỏe, "TL3" là sản phẩm bảo hiểm bệnh hiểm nghèo, ...
- *Li_ID*: mã khách hàng, là số ID của người được sản phẩm bảo hiểm bảo vệ.
- *Sum_Assured*: là số tiền mà công ty bảo hiểm trả cho người được bảo hiểm trong trường hợp rủi ro xảy ra.
- *Effective_Date*: ngày mà hợp đồng bắt đầu có hiệu lực.
- *Chanel*: kênh phân phối, Banca: kênh ngân hàng và Agency: kênh đại lý

Understanding our database

Life table: thông tin về người được bảo hiểm

- *Li_ID*: mã khách hàng, là số ID của người được sản phẩm bảo hiểm bảo vệ.
- *Li_Gender*: là giới tính của người được bảo vệ. Chỉ nhận 1 trong 2 giá trị là "Male" hoặc "Female"
- *Li_DOB*: là ngày sinh của người được bảo vệ.
- *Li_OccupationClass*: cho biết nhóm nghề nghiệp của người được bảo vệ (thường chia làm 4 nhóm từ 1 đến 4). Những người thuộc nhóm thấp (chẳng hạn 1) ít có khả năng xảy ra rủi ro hơn những người ở nhóm cao hơn (chẳng hạn 3).

Understanding our database

Bill table: thông tin đóng phí của khách hàng.

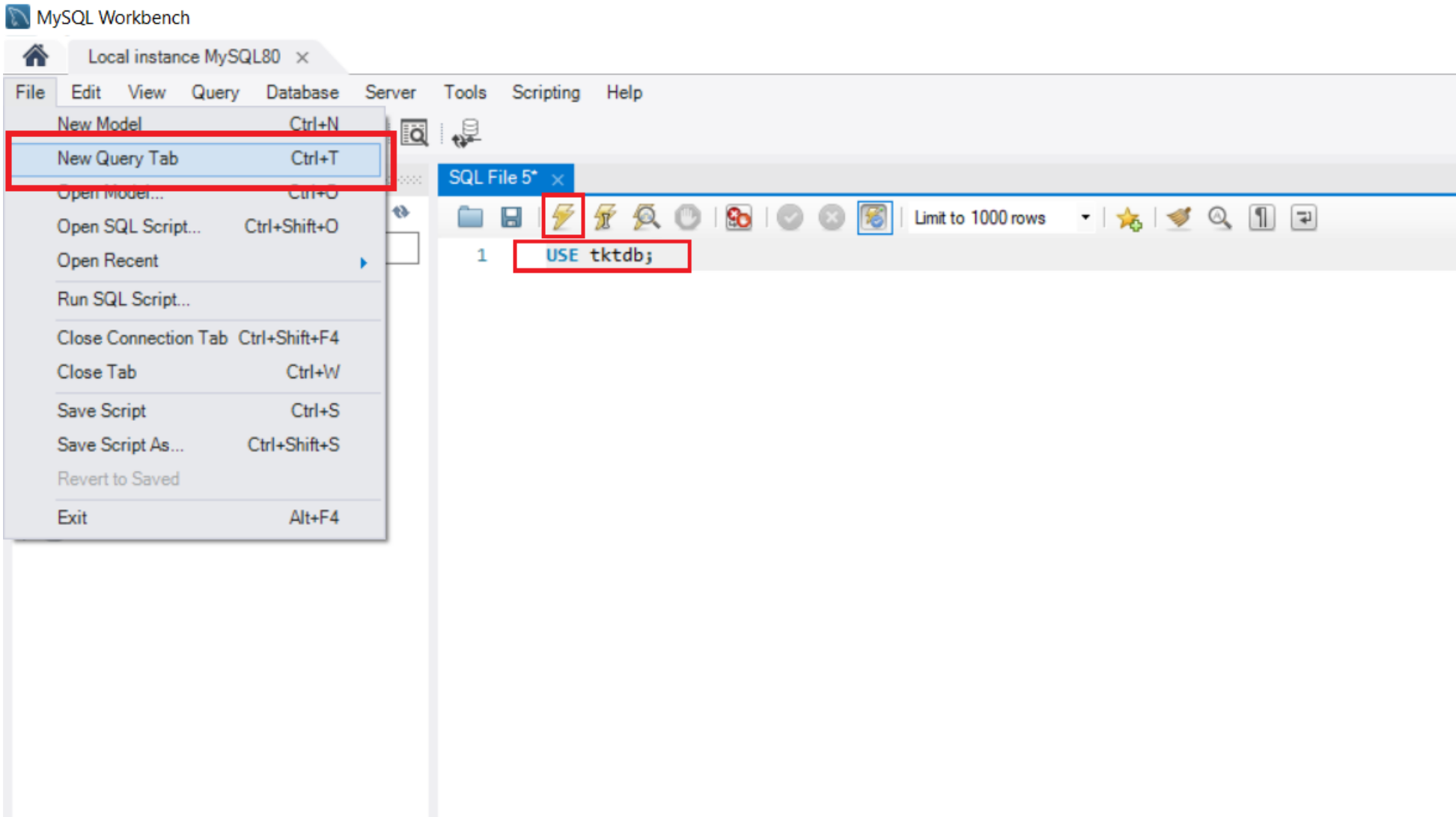
- *Bill_ID*: mã số của hóa đơn (giao dịch)
- *Bill_Date*: là ngày giao dịch của hóa đơn
- *Bill_Amount*: là số tiền mà khách hàng đóng phí.
- *Policy_Number*: là mã số hợp đồng, cho biết giao dịch đóng phí tương ứng với hợp đồng nào.

Understanding our database

- Khi bạn phải quản lý và làm việc với dữ liệu
  1. Từ nhiều nguồn khác nhau: **Policy** table từ hệ thống quản lý hợp đông, **Life** table từ bộ phận nghiệp vụ, **Bill** table từ bộ phận tài chính kế toán
  2. Thường xuyên được cập nhật (theo ngày, tuần, tháng)
  3. Size dữ liệu vượt quá dung lượng cho phép của excel

  hãy sử dụng database để quản lý dữ liệu.
- File data.rar chứa dữ liệu ở dạng file .csv
  - 20 files chứa thông tin hợp đồng
  - 20 files chứa thông tin người được bảo hiểm
  - 20 files chứa thông tin hóa đơn đóng phí
- File *"Upload data to database.R"* là code R dùng để upload 60 file này vào 3 relational tables trên cơ sở dữ liệu **tktdb**

Manipulating data in a database



SELECT ... FROM statement

- Lấy ra từ bảng, tên là table1, các cột có lên là column1, column2, column3 ...

```
SELECT column1, column2, column3
FROM table1; #end with ";"
```

```
SELECT Li_ID, Li_DOB
FROM life;
```

- Lấy ra tất cả các cột từ table1

```
SELECT *
FROM table1;
```

```
SELECT *
FROM life; # All columns from life table
```

Using “WHERE” to filter

- Lấy ra dữ liệu bao gồm các cột 1,2,3 trong table1, được lọc theo điều kiện *con1*

```
SELECT column1, column2, column3
FROM table1
WHERE con1;
```

```
SELECT Li_ID, Li_DOB
FROM life
WHERE Li_DOB < '1970-01-01';
```

Using "ORDER BY" to sort

- Lấy ra dữ liệu bao gồm các cột 1,2,3 trong table1, được lọc theo điều kiện *con1*, dữ liệu được sắp xếp theo giá trị trong cột *column1* tăng dần

```
SELECT column1, column2, column3
FROM table1
WHERE con1
ORDER by column1;
```

```
SELECT Li_ID, Li_DOB
FROM life
WHERE Li_DOB < '1970-01-01'
ORDER BY Li_DOB;
# ORDER BY Li_DOB DESC; #decreasing order
```

SELECT a new column

- Thêm 1 cột mới, không có sẵn trong bảng life

```
SELECT Li_ID, Li_DOB, Li_OccupationClass + 10
FROM life
```

- Để gán tên cho cột mới, sử dụng "AS"

```
SELECT
  Li_ID,
  Li_DOB,
  Li_OccupationClass,
  (Li_OccupationClass + 10) AS 'New_Occupation_Class'
FROM life;
```

SELECT DISTINCT column

- SELECT DISTINCT được sử dụng để loại bỏ các dữ liệu bị lặp trong các cột dữ liệu

```
SELECT distinct Li_OccupationClass
FROM life;
```

- SELECT DISTINCT có thể được sử dụng với nhiều cột dữ liệu

```
SELECT distinct
  Li_Gender,
  Li_OccupationClass
FROM life;
```

Exercise

Từ bảng **Policy**

1. Lấy thông tin của tất cả các hợp đồng có ngày hiệu lực là ngày 01 tháng 02 năm 2001 (điều kiện: Effective_Date = '2001-02-01')
2. Lấy thông tin của tất cả các hợp đồng có ngày hiệu lực là ngày 01 tháng 02 năm 2001 và được phát hành qua kênh Banca (điều kiện: Effective_Date = '2001-02-01' and Chanel = 'Banca')
3. Hợp đồng nào phát hành ngày 01 tháng 02 năm 2001 có số tiền bảo hiểm lớn nhất,nhỏ nhất? (sử dụng ORDER BY)
4. Những mã sản phẩm nào được bán qua kênh Banca trong ngày 01 tháng 02 năm 2001 (SELECT DISTINCT kết hợp với điều kiện)

Solution

```
Select * from Policy
where Effective_Date = '2001-02-01';

Select * from Policy
where Effective_Date = '2001-02-01' and Chanel = 'Banca';

Select * from Policy
where Effective_Date = '2001-02-01' order by Sum_Assured;

Select * from Policy
where Effective_Date = '2001-02-01' order by Sum_Assured DESC;

Select distinct Coverage_Name
from Policy
where Effective_Date = '2001-02-01' and Chanel = 'Banca';
```

"AND", "OR", and "NOT" operators

Lấy thông tin các hợp đồng có ngày hiệu lực 2001-02-01 và có số tiền bảo hiểm (Sum_Assured) nhỏ hơn 120.000.000 hoặc số tiền bảo hiểm lớn hơn 980.000.000.

```
Select *
from Policy
where Effective_Date = '2001-02-01'
    and (Sum_Assured < 120000000 or Sum_Assured > 980000000);
```

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and NOT(Sum_Assured >= 120000000 and Sum_Assured <= 98000000
```

IN operator

Lấy thông tin các hợp đồng có ngày hiệu lực 2001-02-01 và có mã sản phẩm là 'END1' hoặc 'END2' hoặc 'END3'

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and (Coverage_Name = 'END1' or
  Coverage_Name = 'END2' or Coverage_Name = 'END3');
```

IN operator

Lấy thông tin các hợp đồng có ngày hiệu lực 2001-02-01 và có mã sản phẩm là 'END1' hoặc 'END2' hoặc 'END3'

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and (Coverage_Name = 'END1' or
  Coverage_Name = 'END2' or Coverage_Name = 'END3');
```

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and Coverage_Name IN ('END1','END2', 'END3');
```

LIKE operator

LIKE được sử dụng trong các biểu thức điều kiện của biến dạng ký tự (string). Giả sử bạn không nhớ chính xác tên mã sản phẩm, chỉ nhớ mã sản phẩm bắt đầu bằng 'EN'

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and Coverage_Name LIKE 'EN%' ;
```

Mã sản phẩm có chứa 'UL'

```
Select *
from Policy
where Effective_Date = '2001-02-01'
  and Coverage_Name LIKE '%UL%' ;
```

LIMIT operator

LIMIT được sử dụng để giới hạn số dòng khi lấy dữ liệu. Xem 2 ví dụ sau:

- Lấy thông tin 5 hợp đồng đầu tiên (theo thứ tự trong dữ liệu) có ngày hiệu lực 2001-02-01

```
Select *
from Policy
where Effective_Date = '2001-02-01'
LIMIT 5;
```

- Lấy ra thông tin 3 hợp đồng, bắt đầu từ hợp đồng thứ 10 theo thứ tự trong dữ liệu, mà có ngày hiệu lực là 2001-02-01

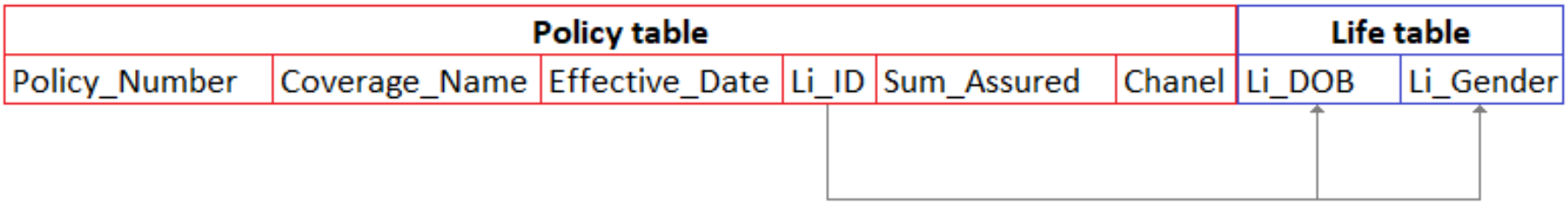
```
Select *
from Policy
where Effective_Date = '2001-02-01'
LIMIT 10, 3;
```

Exercise

1. Lấy ra các hợp đồng có mã sản phẩm là "UL1" và có số tiền bảo hiểm lớn hơn 950 triệu.
2. Lấy ra các hợp đồng có mã sản phẩm có chứa chữ "UL" và có số tiền bảo hiểm từ 110 triệu đến 120 triệu.
3. Lấy ra dữ liệu về người được bảo hiểm là nam (Male) và có năm sinh là năm 1985.
4. Lấy ra dữ liệu về 5 người được bảo hiểm đầu tiên có OccupationClass là 4.
5. Lấy ra dữ liệu về 5 hợp đồng có số tiền bảo hiểm nhỏ nhất thỏa mãn các điều kiện: 1) Phát hành vào ngày 2001-02-01 2) Mã sản phẩm KHÔNG có chữ "TL"

INNER JOIN

- INNER JOIN cho phép lấy thông tin từ nhiều table khác nhau vào 1 bảng
- Từ bảng Policy, để tính được phí của mỗi hợp đồng, ta cần thêm vào các thông tin từ bảng Life: Giới tính của người được bảo hiểm (cột Li_Gender), ngày sinh của người được bảo hiểm (cột Li_DOB), và occupation class (cột Li_OccupationClass), các thông tin này tương ứng với mã của người được bảo hiểm (cột Li_ID) của bảng Policy



INNER JOIN

- Cách sử dụng INNER JOIN để lấy thông tin từ 2 bảng Policy và Life như sau:

```
Select *
from Policy
JOIN Life
    on Policy.Li_ID = Life.Li_ID;
```

- Cột Li_ID bị lặp lại do cả 2 bảng đều có cột này. Sử dụng cú pháp như sau để loại bỏ đi cột Li_ID không cần thiết

```
Select Policy.*, Li_DOB, Li_Gender, Li_OccupationClass
from Policy
JOIN Life
    on Policy.Li_ID = Life.Li_ID;
```

INNER JOIN

INNER JOIN có thể được sử dụng để ghép nhiều bảng: Cần lấy ra thông tin từ 3 bảng **Bill**, **Policy**, **Life** với các thông tin sau

- Tất cả các cột của bảng **Bill**
- Cột Effective_Date và Li_ID từ bảng **Policy** tương ứng với cột Policy_Number từ bảng **Bill**
- Cột Li_Gender và Li_DOB tương ứng với cột Li_ID từ bảng **Policy**

INNER JOIN

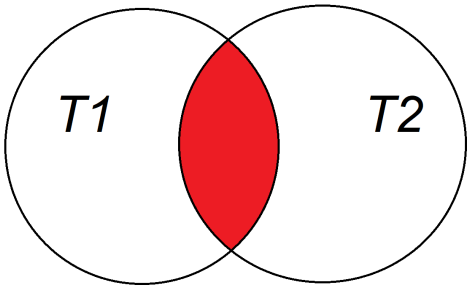
INNER JOIN có thể được sử dụng để ghép nhiều bảng: Cần lấy ra thông tin từ 3 bảng **Bill**, **Policy**, **Life** với các thông tin sau

- Tất cả các cột của bảng **Bill**
- Cột Effective_Date và Li_ID từ bảng **Policy** tương ứng với cột Policy_Number từ bảng **Bill**
- Cột Li_Gender và Li_DOB tương ứng với cột Li_ID từ bảng **Policy**

```
Select Bill.*, Effective_Date, Policy.Li_ID, Life.Li_Gender, L
from Bill
JOIN Policy
    on Bill.Policy_Number = Policy.Policy_Number
JOIN Life
  on Policy.Li_ID = Life.Li_ID;
```

OUTER JOIN

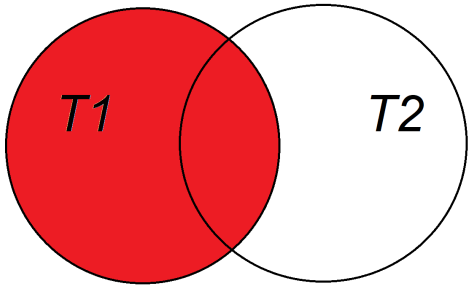
INNER JOIN



*Select * from T1*
Join T2
on T1.key = T2.key

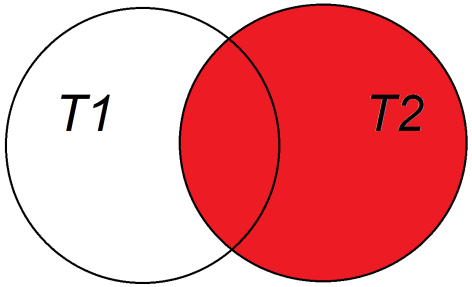
OUTER JOIN

LEFT JOIN



*Select * from T1*
Left Join T2
on T1.key = T2.key

RIGHT JOIN



*Select * from T1*
Right Join T2
on T1.key = T2.key

To figure out the difference between inner join and outer join, let try the following example:

```
CREATE TABLE t1 ( A INT, B INT);
CREATE TABLE t2 (B INT, C varchar(2));
INSERT INTO t1 VALUES (1,1),(2,1),(3,3),(4,3),(5,5),(6,5);
INSERT INTO t2 VALUES (1,'X'),(2,'Y'),(3,'Z'),(4,'T');
```

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# (INNER) JOIN
SELECT *
  FROM t1
  JOIN t2
    on t1.B = t2.B;
```

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# (INNER) JOIN
SELECT *
  FROM t1
  JOIN t2
    on t1.B = t2.B;
```

→ Chỉ có các dòng mà giá trị của cột B đồng thời xuất hiện ở cả 2 bảng t1 và bảng t2 được lẩy ra.

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# LEFT JOIN
SELECT *
  FROM t1
  LEFT JOIN t2
    on t1.B = t2.B;
```

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# LEFT JOIN
SELECT *
  FROM t1
  LEFT JOIN t2
    on t1.B = t2.B;
```

$\rightarrow$ Tất cả các dòng tương ứng với giá trị cột B trong bảng t1 (bảng bên trái) đều được lấy ra. Các cột tương ứng với giá trị trong cột B của bảng t1 không tìm thấy trong cột B của bảng t2 sẽ được gán giá trị là NULL

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# LEFT JOIN
SELECT *
  FROM t1
  RIGHT JOIN t2
    on t1.B = t2.B;
```

OUTER JOIN

Try the following code to find out the difference between JOIN, LEFT JOIN and RIGHT JOIN

```
# LEFT JOIN
SELECT *
  FROM t1
  RIGHT JOIN t2
    on t1.B = t2.B;
```

$\rightarrow$ Tất cả các dòng tương ứng với giá trị cột B trong bảng t2 (bảng bên phải) đều được lấy ra. Các cột tương ứng với giá trị trong cột B của bảng t2 không tìm thấy trong cột B của bảng t1 sẽ được gán giá trị là NULL.

UNION

UNION được sử dụng để nối 2 bảng theo hàng.

- 2 bảng phải có cùng số cột
- Các cột tương ứng phải có cùng kiểu biến

Lấy thông tin của tất cả những người sinh ngày 29 tháng 02 năm 1964 từ bảng Life, sau đó thêm vào 1 cột có tên là "Risk_Level". Cột này nhận giá trị là "Normal" nếu Li_OccupationClass nhận giá trị là 1 hoặc 2, và nhận giá trị là "High" nếu Li_OccupationClass nhận giá trị là 3 hoặc 4

```
SELECT *, "Normal" as Risk_Level FROM Life
  where Li_DOB = '1964-02-29' and Li_OccupationClass IN (1,2)
UNION
SELECT *, "High" as Risk_Level FROM Life
  where Li_DOB = '1964-02-29' and Li_OccupationClass IN (3,4);
```

INSERT

INSERT được sử dụng để insert dữ liệu vào một table trong database. Để thêm thông tin của một số người được bảo hiểm vào bảng Life:

```
INSERT INTO Life
VALUES
  ('LI00001','Male','2000-01-01',3),
  ('LI00002','Female','2001-12-31',2);
```

We can check that these lives insured are inserted into Life table

```
Select * from Life where Li_ID in ('LI00001','LI00002')
```

UPDATE

UPDATE được sử dụng để update dữ liệu hiện có trong 1 table. Giả sử ta muốn thay thế ngày sinh của người được bảo hiểm mã 'LI00001' bằng '1995-12-31', sử dụng Update như sau

```
SET SQL_SAFE_UPDATES = 0; #temporarily turn off safe update m
UPDATE Life
SET Li_DOB = '1995-12-31' where Li_ID = 'LI00001';
```

UPDATE có thể được sử dụng để thay thế nhiều giá trị cùng lúc: Giả sử ta muốn đổi OccupationClass của 2 lives insured vừa được thêm vào thành 1:

```
UPDATE Life
SET Li_OccupationClass = 1 where Li_ID in ('LI00001','LI00002'
#CHECK UPDATE
Select * from Life where Li_ID in ('LI00001','LI00002');
```