VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Database System Lab (CO2014)

ASSIGNMENT 1 REPORT

Retail Management System

Lecturer: Assoc. Prof. Trần Minh Quang

Class : CC01

Students: Lê Đức Cầm - 1952588
Nguyễn Tiến Dương - 1952639
Phạm Mạnh Dũng - 1952633

HO CHI MINH CITY, SEPTEMBER 2021

Contents

1 Requirement description

We are going to design a database to keep track of information for a retail company. Assume that the following requirements were collected from the company:

The retail company is run by many Person. Each Person has a unique Social Security Number (ssn), name, address, date of birth, sex, phone number, and email. The Person objects are categorized in several ways, as discussed below.

Person are categorized based on their role. There are four main types—Clerk, Shipper, Security officer and Customer. Clerk has some certain skills and years of experiment (5, 10, 15 years, etc) to determine whether they are experienced or not. Shipper has vehicle using to present how they can do their job. Security officer also has some skills and physical fitness.

Customer has credit card number and shopping history to record and analyze their preference with the aim of improving their personal experience in the future. Whenever the Customer make a transaction, a Bill is created with the following information: a unique **bill_no**, date, volume, tax and discount (if available). Each Bill contains a number of Product which has a unique product id, image, color and **p_name** which represent the name of supplier.

Each Product include a certain number of Item which is represented by a **item_id** and availability. The Product is provided at a **unit_price** by a Supplier. Each Supplier has a unique **s_name** to differentiate with others.

The retail company may have many Store at different location and area to create favorable conditions for development. Each Store has its Warehouse which has a unique **warehouse_id**, number of floor and address.

An Administrator will be in charge of managing the human resources and facilities. Similar to other people, they also have a unique **admin_id** and **admin_name**.

2 Tool using for database design

We choose an illustration tool called *LucidChart* to demonstrate the conceptual,logical and physical design of our database.

What is LucidChart ?

⋆ Lucidchart is a web-based proprietary platform that allows users to collaborate on drawing, revising and sharing c
⋆ It is produced by Lucid Software Inc., based in Utah, United States.
⋆ Lucidchart runs on browsers that support HTML5. This means it does not require updates of a third-party softwa
⋆ In 2010, Lucidchart announced they had integrated into the Google Apps Marketplace.

Reason for choosing LucidChart for demonstrating our database design:

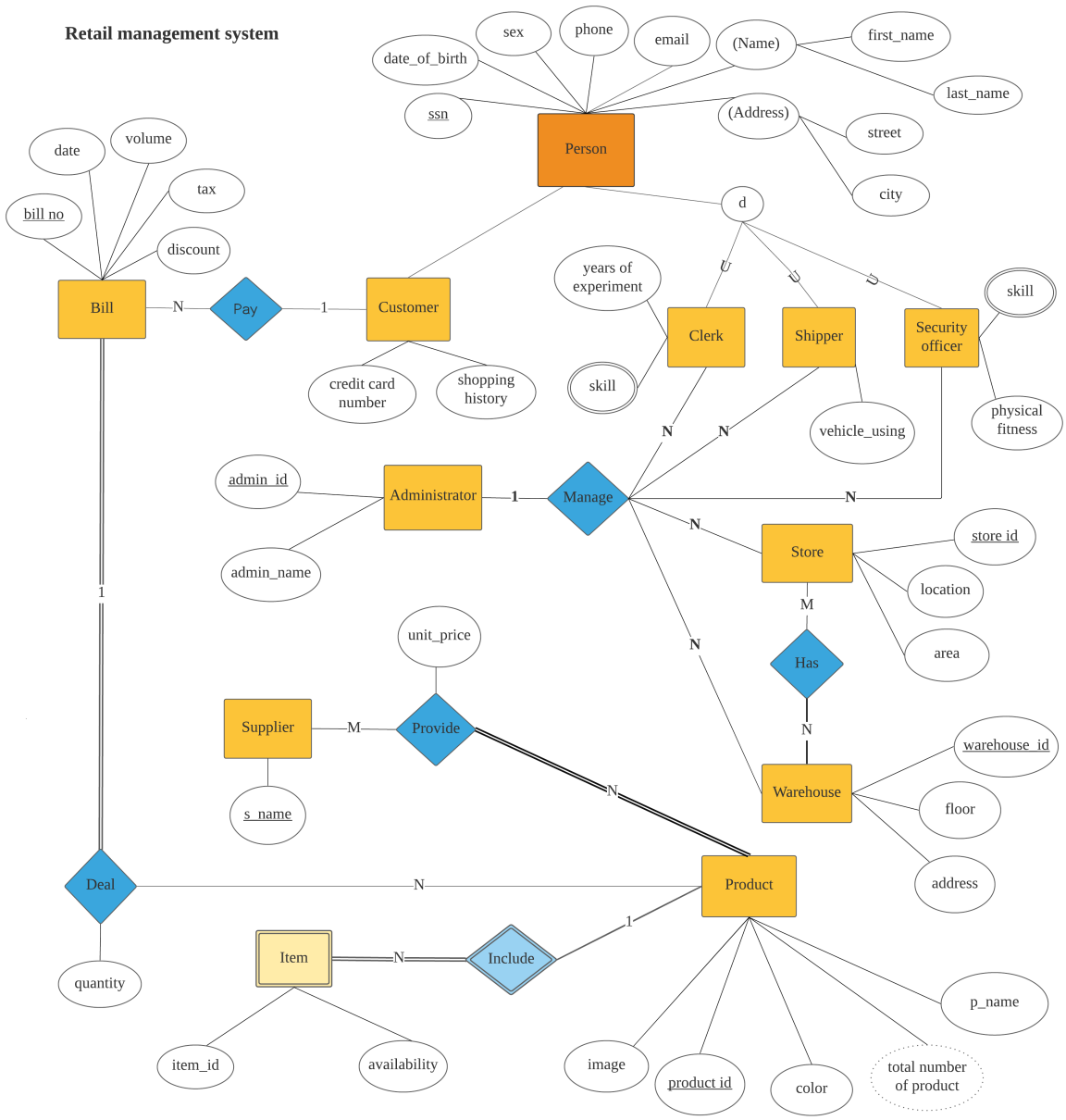
⋆ Easy to use with the abundance of useful document.
⋆ Friendly GUI, provide many colorful visualization tool.
⋆ Allow relationship establishing within a click, easily modify the relationship types.
⋆ Provide wide number of features.
⋆ Import and export code to famous DBMS (MySQL oracle, PostgreSQL, and SQL Server).
⋆ Support unlimited number of collaborators on free version.
⋆ Multiple platform supported (Web based, Android, IOS).
⋆ Can be used online without download anything

3 Conceptual database design

3.1 Conceptual design definition

Conceptual design is an early phase of the design process, in which the broad outlines of function and form of something are articulated. It includes the design of interactions, experiences, processes, and strategies. It involves an understanding of people's needs and how to meet them with products, services, and processes. Common artifacts of conceptual design are concept sketches and models.

3.2 Conceptual model diagram



3.3 Description about conceptual diagram

According to the conceptual model diagram, a Person has a key attribute is **Social Security Number (ssn)**, two composite attributes: name vs address and another information. A Person can be a Customer which has two more further attributes: credit card number and shopping history. Person may be grouped into Clerk, Shipper or Security officer:

- Each of these groups is a subset of Person entity.
- Each is called a sub-class of Person.
- Person is the super-class for each of these sub-classes.
- A Person can only involve in only one in three sub-class because disjoint relationship.

An entity that is a member of a sub-class inherits all attributes of the entity as a member of the super-class, it also inherits all relationships. Example: Clerk may contains more than one skills, years of experiment attribute and all attributes of Person such as ssn, name, address, ...

The Bill entity has a many-to-one Pay relationship with Customer entity which means more than one instance of Bill entity is associated with the Pay relationship.Each Bill entity is involved in the Deal relationship, one participation is represented by double lines and Partial participation is represented by single lines which means not all quantity is involved in the relationship.

An Administrator has **admin_id** is a key attribute and this entity in a one-to-many Manage relationship with Clerk, Shipper, Security officer, Store and Warehouse which means that one administrator can manage many entity and attribute.

A Store has **store_id** is a key attribute and this entity in a many-to-many relationship with Warehouse has **warehouse_id** is a key attribute which means that more than one instance of a store and more than one instance of a warehouse can be associated with the relationship, example: one store can have many warehouse, many stores can have one warehouse.

A Product has **product_id** is a key attribute and **total number of product** is a derived attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type. Deal relationship between Product and Bill is many-to-one and is mandatory which means that many products have only one bill and every instance of a participating entity type must participate in the relationship. Product has many-to-many Provide relationship with Supplier, every instance of a Product must participate in the relationship but - not every instance of a Supplier must participate in the relationship.Moreover, Item is a weak entity related to Product, which means it can not appear without Product entity

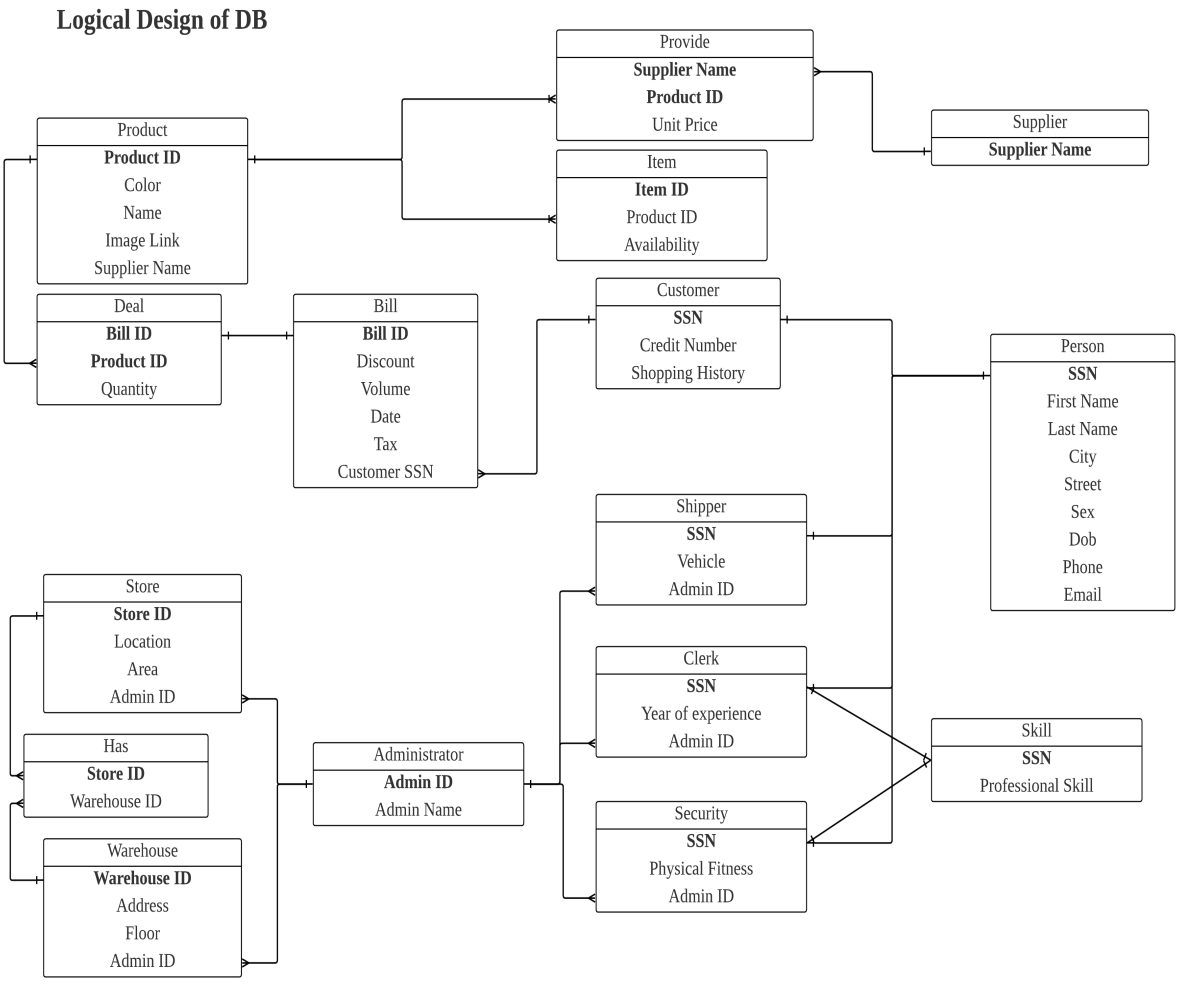
A Supplier has **s_name** is a key attribute and Include is a identifying relationship. The Item has a many-to-one relationship with a Product which means that all of the instance of the Item must be included in a certain Product but Product may not need any Item.

4 Logical Database Design

4.1 Logical design definition

A logical design is a conceptual, abstract design. We define the types of information that we need here.
The process of logical design involves arranging data into a series of logical relationships called entities and attributes. An entity represents a chunk of information. In relational databases, an entity often maps to a table. An attribute is a component of an entity and helps define the uniqueness of the entity. In relational databases, an attribute maps to a column.

4.2 Logical design diagram



4.3 Mapping explanation

⋆ **Step 1: Mapping of Regular Entity Types**:
We have turned strong entities into relations, take the attributes of entities to be the attribute of the tables which have just been created and also set primary key of each entity as the primary key of related relation. This step include entities like 'Bill', 'Administrator', 'Store', 'Warehouse', 'Product', 'Supplier'.

⋆ **Step 2: Mapping of Weak Entity Types**:
This step comprise of mapping Entity 'Item' to the 'Item' tables with its regular attributes and one more attribute as foreign key pointing to primary key of 'Product' relation.The primary key of 'Item' is the combination of its partial key and the primary key of the owner('Product' relation).

⋆ **Step 3,4,5: Mapping of Binary Relationships**:
There is a mandatory membership class in 1:N relationship but N-side is in optional membership class so we construct a new tables named 'Deal'. Others 1:N relationship we simply add new foreign key refer to admin_id in 'Administrator relation, in this case it appear in 5 relations: 'Clerk', 'Shipper', 'Security Officer', 'Store', 'Warehouse'. For M:N relationship we simply create new relation with 2 primary key of 2 entities become the composite key of the new relation, we can see it in 'Provide' and 'Has' relations

⋆ **Step 6: Mapping of Multivalued Entity**:
We just add a new tables called 'Skill' relation with the foreign key point to 'Clerk' and 'Security Officer' and its own attribute named professional skill.

⋆ **Step 7: Mapping of N-ary Relationships**:
This case we don't consider.

⋆ **Step 8,9: Mapping of Specializations and Union**: We use option 8A-Multiple relations-Superclass and sub-classes it means in relations like 'Clerk', 'Customer','Shipper' and 'Security Officer', beside it own attribute we have an attribute as foreign key refer to their superclass 'Person'.

5 Database Management System

5.1 What is MySQL?

MySQL is an open-source relational database management system (RDBMS) with sponsorship and ownership of the Swedish company "MySQL AB", which was bought by Sun Microsystems(now Oracle Corporation). It works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is a free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. It has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability.

This DBMS is used by many database-driven web applications, including phpBB, WordPress, ... and by many popular websites, such as Facebook, MediaWiki, Twitter, and YouTube.

5.2 Reason for choosing MySQL for implementing our DB system

⋆ **Data Security**: MySQL is globally renowned for being the most secure and reliable database management system. The data security and support for transaction processing.

⋆ **High Performance**: MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance.

⋆ **On-Demand Scalability**: MySQL offers unmatched scalability to facilitate the management of deeply embedded apps using a smaller footprint even in massive warehouses that stack terabytes of data.

⋆ **Low Cost**: It provide community version(free-version) for everyone who need a tool to solve their problems.

⋆ **Complete Workflow Control**: MySQL is a comprehensive solution with self-management features that automate everything from space expansion and configuration to data design and database administration.

⋆ **The Flexibility Of Open Source**: All the fears and worries that arise in an open source solution can be brought to an end with My SQL's round-the-clock support and enterprise indemnification.

⋆ **The Most Important Reason**: Easy to use with many available document and online tutorials.

6 Physical Database Design based on selected DBMS

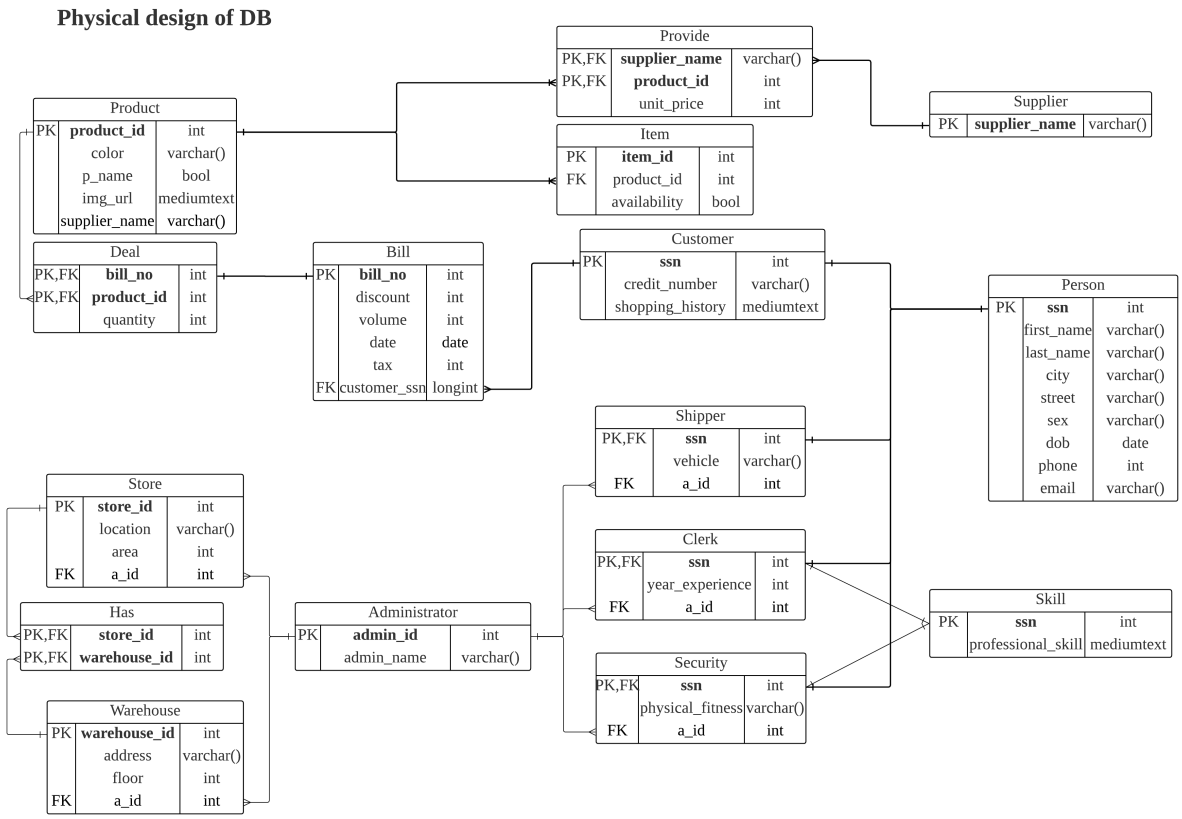
6.1 Physical design definition

Physical database design represents the materialization of a database into an actual system. While logical design can be performed independently of the eventual database platform, many physical database attributes depend on the specifics and semantics of the target DBMS. Physical design is performed in two stages:

1. Conversion of the logical design into table definitions (often performed by an application developer): includes pre-deployment design, table definitions, normalization, primary and foreign key relationships, and basic indexing.

2. Post deployment physical database design (often performed by a database administrator): includes improving performance, reducing I/O, and streamlining administration tasks.

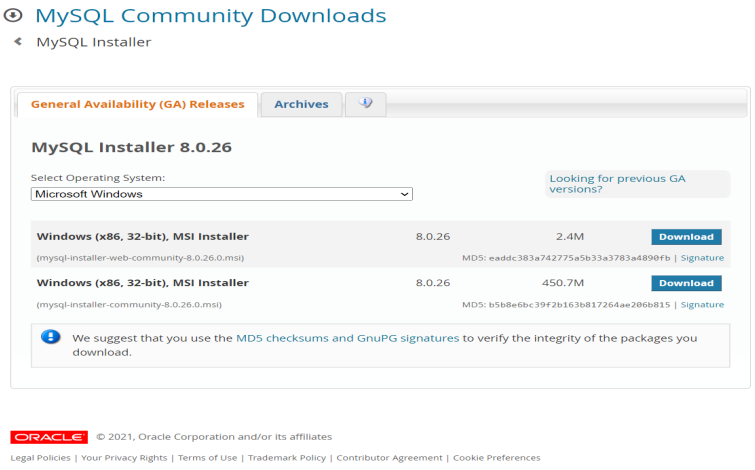
6.2 Physical model diagram



7 Implementation into MySQL

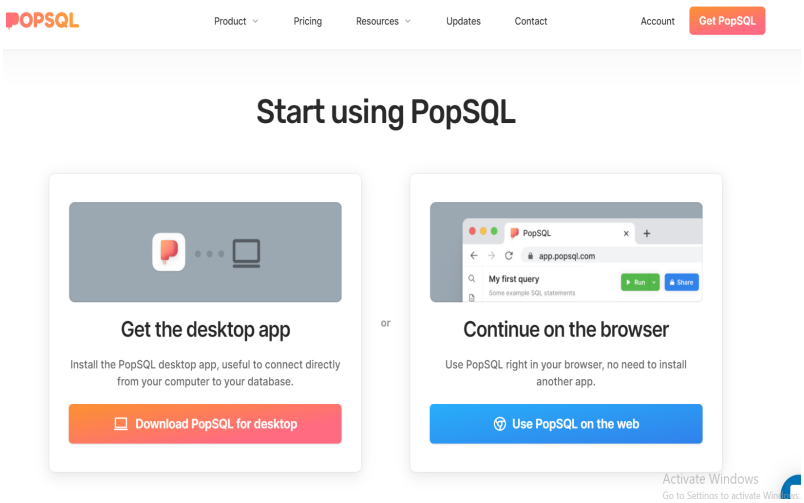
⋆ **Step 0**: Install DBMS and prepare environment

· First go to website of MySQL via the link: https://dev.mysql.com/downloads/installer/ → Then choose which one is suitable for your computer, in our case we use the MySQL for Windows, we can see the figure below:

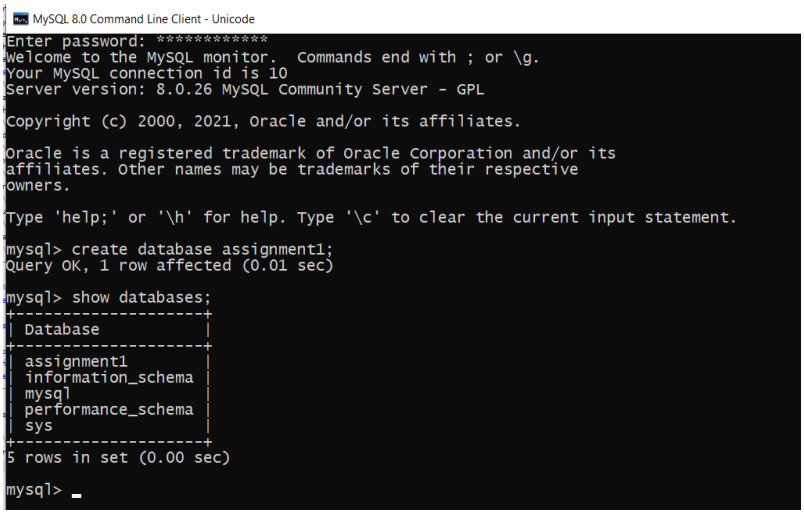


· Next we start installing MySQL to our computer by running the file name "mysql-installer-web-community-8.0.26.0.msi" and just choose default settings with 3 product which are MySQL Server, MySQL Workbench(optional) and MySQL shell, after all you can see a small window like this:

· After that we go to https://popsql.com to set up editor environment, sign in and there will appear a web page like this and you can choose the option which is suitable for you (in our case,we chose to download it)

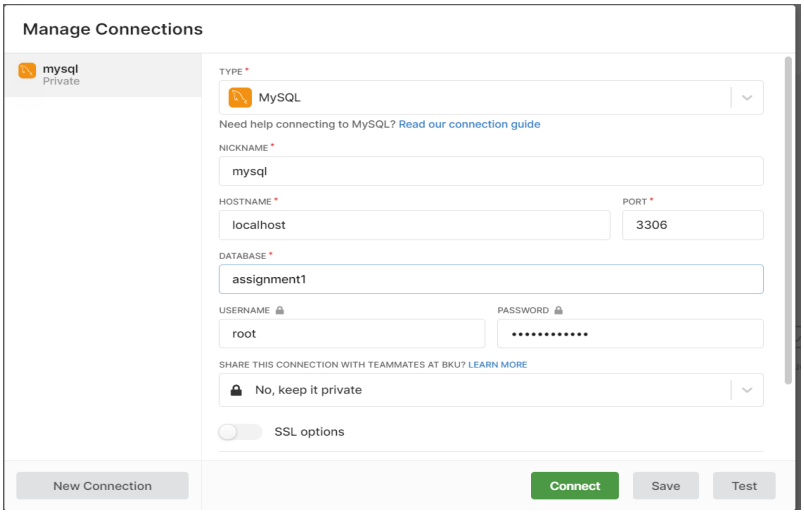


⋆ **Step 1**: Open "MySQL Command Line Client-Unicode" → Enter password → Create a database named "assignment1" and check if it is successfully created



⋆ **Step 2**: Run SQL code in PopSQL(an sql text editor)

- Connect to the database name "assignment1" we have just created



- Create tables followed the physical design above

```
CREATE TABLE `Person` (
  `ssn` int PRIMARY KEY,
  `first_name` varchar(10),
  `last_name` varchar(10),
  `city` varchar(10),
  `street` varchar(10),
  `sex` varchar(10),
  `data_of_birth` date,
  `phone` int,
  `email` varchar(30)
);
CREATE TABLE `Customer` (
  `ssn` int PRIMARY KEY,
  `credit_card_number` varchar(30) NOT NULL,
  `shopping_history` mediumtext
);
CREATE TABLE `Clerk` (
  `ssn` int PRIMARY KEY,
  `year_of_experiment` int NOT NULL,
  `a_id` int
);
CREATE TABLE `Security_Officer` (
  `ssn` int PRIMARY KEY,
  `physical_fitness` varchar(30) NOT NULL,
  `a_id` int
```

```
);
CREATE TABLE `Shipper` (
  `ssn` int PRIMARY KEY,
  `vehicle_using` varchar(20) NOT NULL,
  `a_id` int
);
CREATE TABLE `Skill` (
  `ssn` int PRIMARY KEY,
  `professional_skill` mediumtext
);
CREATE TABLE `Administrator` (
  `admin_id` int PRIMARY KEY,
  `admin_name` varchar(20)
);
CREATE TABLE `Product` (
  `product_id` int PRIMARY KEY,
  `color` varchar(20),
  `p_name` varchar(20) NOT NULL,
  `image_url` text,
  `supplier_name` varchar(20)
);
CREATE TABLE `Item` (
  `product_id` int NOT NULL,
  `item_id` int NOT NULL,
  `Availability` bool NOT NULL DEFAULT 0,
  PRIMARY KEY (`product_id`, `item_id`)
);
CREATE TABLE `Bill` (
  `bill_no` int PRIMARY KEY AUTO_INCREMENT,
  `discount` int,
  `volume` int NOT NULL,
  `date` date,
  `tax` int,
  `customer_ssn` int NOT NULL
);
CREATE TABLE `Deal` (
  `bill_no` int NOT NULL,
  `product_id` int NOT NULL,
  `quantity` int DEFAULT 1,
  PRIMARY KEY (`bill_no`, `product_id`)
);
CREATE TABLE `Supplier` (
  `s_name` varchar(20) PRIMARY KEY
);
CREATE TABLE `Provide` (
  `s_name` varchar(20),
  `product_id` int,
  `unit_price` int NOT NULL,
  PRIMARY KEY (`s_name`, `product_id`)
```

```
);
CREATE TABLE `Store` (
  `store_id` int PRIMARY KEY,
  `location` varchar(20) NOT NULL,
  `area` int DEFAULT 50,
  `a_id` int
);
CREATE TABLE `Warehouse` (
  `warehouse_id` int PRIMARY KEY,
  `address` varchar(20) NOT NULL,
  `floor` int DEFAULT 1,
  `a_id` int
);
CREATE TABLE `Has` (
  `warehouse_id` int,
  `store_id` int,
  PRIMARY KEY (`warehouse_id`, `store_id`)
);
```

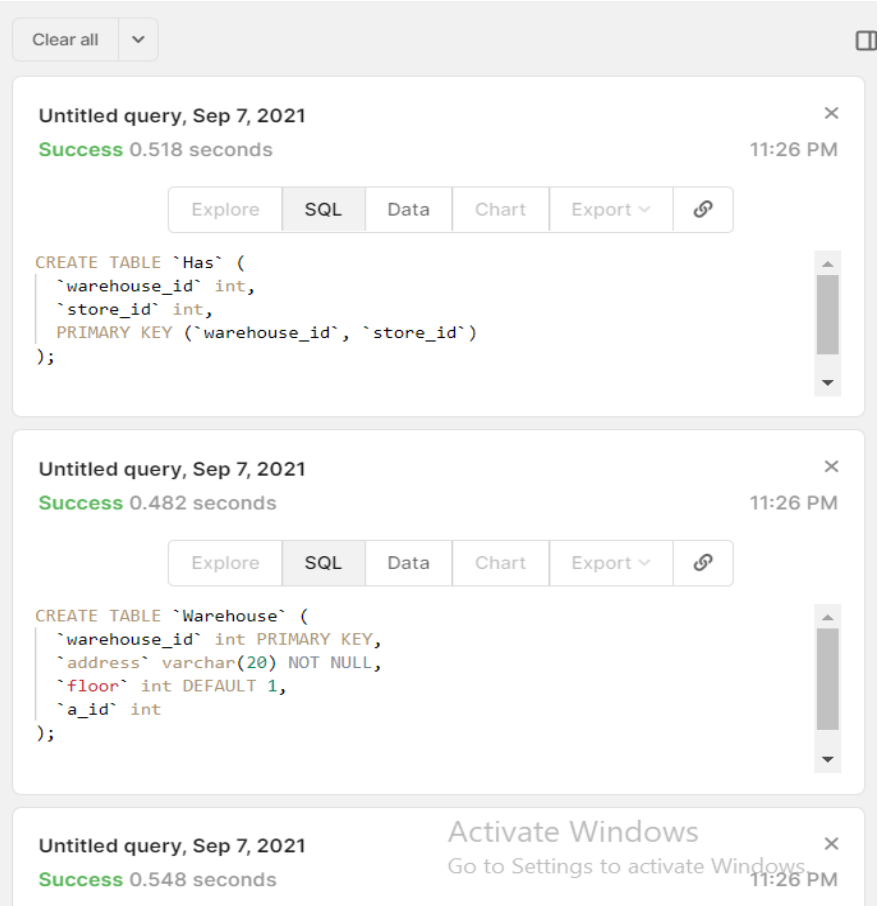
- Create all relations to complete our database creation

```
ALTER TABLE `Customer` ADD FOREIGN KEY (`ssn`) REFERENCES `Person`
↪  (`ssn`);

ALTER TABLE `Clerk` ADD FOREIGN KEY (`ssn`) REFERENCES `Person` (`ssn`);

ALTER TABLE `Security_Officer` ADD FOREIGN KEY (`ssn`) REFERENCES `Person`
↪  (`ssn`);

ALTER TABLE `Shipper` ADD FOREIGN KEY (`ssn`) REFERENCES `Person` (`ssn`);

ALTER TABLE `Skill` ADD FOREIGN KEY (`ssn`) REFERENCES `Clerk` (`ssn`);

ALTER TABLE `Skill` ADD FOREIGN KEY (`ssn`) REFERENCES `Security_Officer`
↪  (`ssn`);

ALTER TABLE `Clerk` ADD FOREIGN KEY (`a_id`) REFERENCES `Administrator`
↪  (`admin_id`);

ALTER TABLE `Security_Officer` ADD FOREIGN KEY (`a_id`) REFERENCES
↪  `Administrator` (`admin_id`);

ALTER TABLE `Shipper` ADD FOREIGN KEY (`a_id`) REFERENCES `Administrator`
↪  (`admin_id`);

ALTER TABLE `Bill` ADD FOREIGN KEY (`customer_ssn`) REFERENCES `Customer`
↪  (`ssn`);

ALTER TABLE `Item` ADD FOREIGN KEY (`product_id`) REFERENCES `Product`
↪  (`product_id`);
```

```
ALTER TABLE `Deal` ADD FOREIGN KEY (`product_id`) REFERENCES `Product`
↪  (`product_id`);

ALTER TABLE `Deal` ADD FOREIGN KEY (`bill_no`) REFERENCES `Bill`
↪  (`bill_no`);

ALTER TABLE `Provide` ADD FOREIGN KEY (`s_name`) REFERENCES `Supplier`
↪  (`s_name`);

ALTER TABLE `Provide` ADD FOREIGN KEY (`product_id`) REFERENCES `Product`
↪  (`product_id`);

ALTER TABLE `Has` ADD FOREIGN KEY (`warehouse_id`) REFERENCES `Warehouse`
↪  (`warehouse_id`);

ALTER TABLE `Has` ADD FOREIGN KEY (`store_id`) REFERENCES `Store`
↪  (`store_id`);

ALTER TABLE `Warehouse` ADD FOREIGN KEY (`a_id`) REFERENCES
↪  `Administrator` (`admin_id`);

ALTER TABLE `Store` ADD FOREIGN KEY (`a_id`) REFERENCES `Administrator`
↪  (`admin_id`);
```

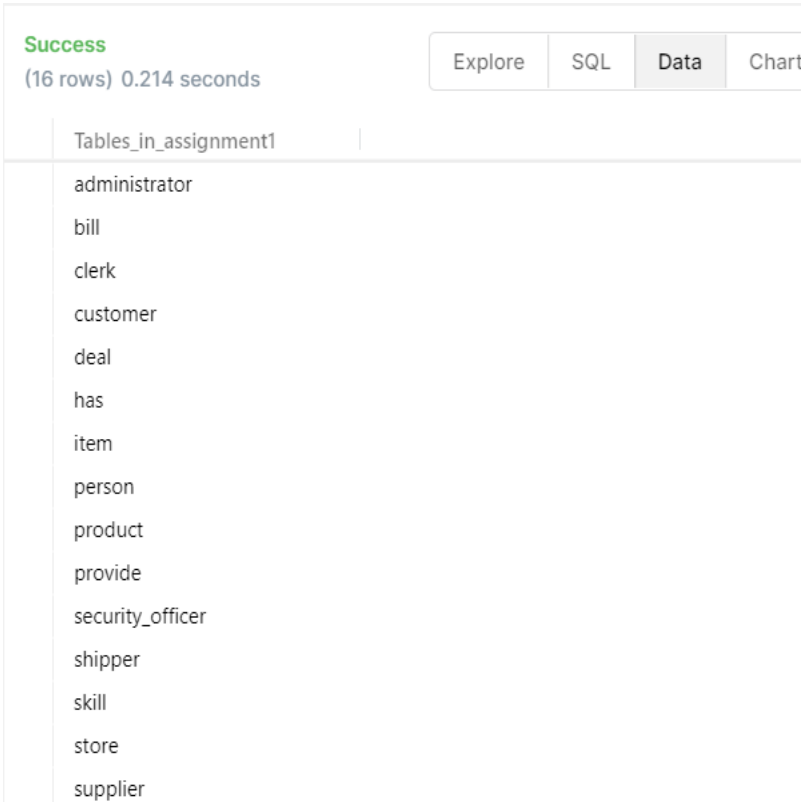
⋆ **Step 3**: Check all tables and relations to make sure whether they have been successfully created or not.

- See the result below to check the table-creating process.
  · Ensure all queries worked well:



· Make sure all tables are successfully created by running following command:

```
SHOW tables;
```



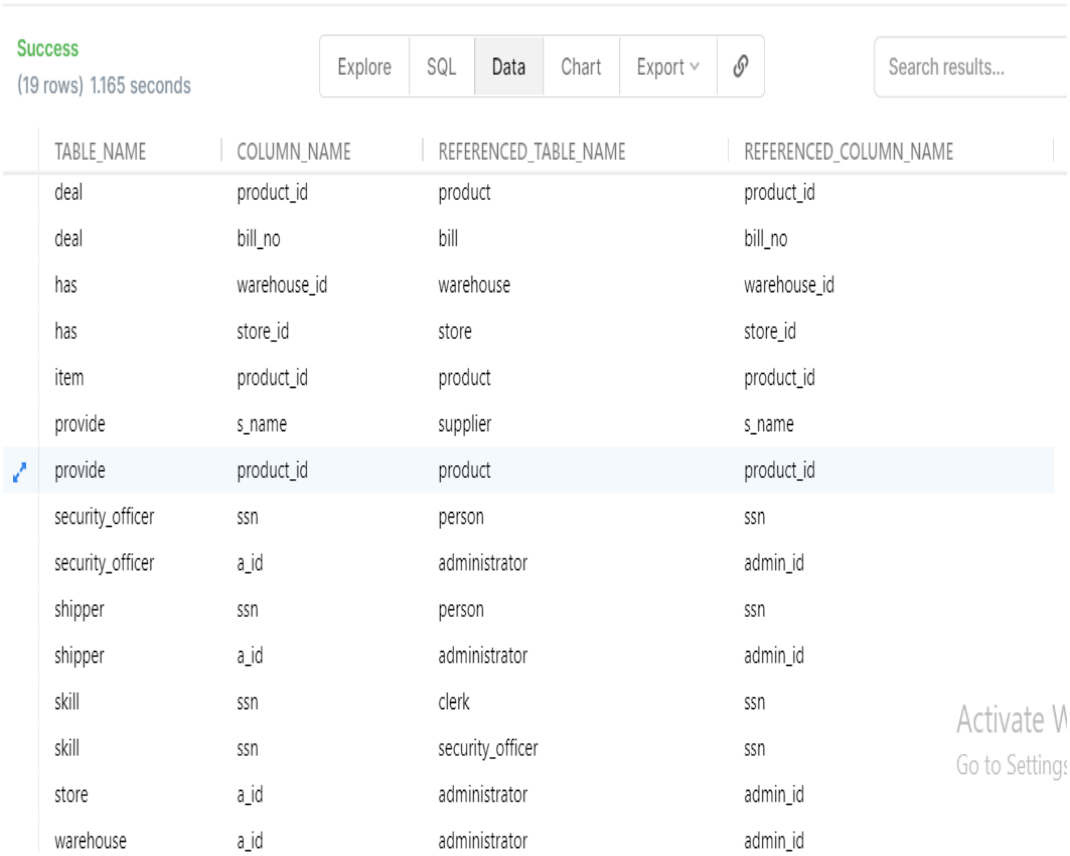
- Below figure are relationships between tables in our relational DB when executing following command:

```
SELECT                              -- Foreign key schema
  `TABLE_NAME`,                                 -- Foreign key table
  `COLUMN_NAME`,                                -- Foreign key column
  `REFERENCED_TABLE_NAME`,                      -- Origin key table
  `REFERENCED_COLUMN_NAME`                      -- Origin key column
FROM
  `INFORMATION_SCHEMA`.`KEY_COLUMN_USAGE`   -- Will fail if user don't have
  ↪  privilege
WHERE
  `TABLE_SCHEMA` = SCHEMA()                     -- Detect current schema in USE
  AND `REFERENCED_TABLE_NAME` IS NOT NULL; -- Only tables with foreign
  ↪  keys
```

Success
(19 rows) 1.165 seconds

Explore | SQL | Data | Chart | Export | Search results...

TABLE_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
bill	customer_ssn	customer	ssn
clerk	ssn	person	ssn
clerk	a_id	administrator	admin_id
customer	ssn	person	ssn



Success
(19 rows) 1.165 seconds

Explore | SQL | Data | Chart | Export | Search results...

TABLE_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
deal	product_id	product	product_id
deal	bill_no	bill	bill_no
has	warehouse_id	warehouse	warehouse_id
has	store_id	store	store_id
item	product_id	product	product_id
provide	s_name	supplier	s_name
provide	product_id	product	product_id
security_officer	ssn	person	ssn
security_officer	a_id	administrator	admin_id
shipper	ssn	person	ssn
shipper	a_id	administrator	admin_id
skill	ssn	clerk	ssn
skill	ssn	security_officer	ssn
store	a_id	administrator	admin_id
warehouse	a_id	administrator	admin_id

- Now we add some values to our database.

```
    -- insert into Administrator table
INSERT INTO Administrator (admin_id,admin_name)
VALUES (1,"A");

INSERT INTO Administrator (admin_id,admin_name)
VALUES (1,"B");

INSERT INTO Administrator (admin_id,admin_name)
VALUES (3,"");

INSERT INTO Administrator (admin_id,admin_name)
VALUES (4,"Cam");

INSERT INTO Administrator (admin_id,admin_name)
VALUES (5,"Dung");

INSERT INTO Administrator (admin_id,admin_name)
VALUES (6,"Duong");
    -- insert into Warehouse table
INSERT INTO Warehouse (warehouse_id,address,floor,a_id)
```

```
VALUES (1,"Street1",5,1);

INSERT INTO Warehouse (warehouse_id,address,floor,a_id)
VALUES (2,"Street2",'1',1);

INSERT INTO Warehouse (warehouse_id,address,floor,a_id)
VALUES (3,"Street1",'1',2);

INSERT INTO Warehouse (warehouse_id,address,floor,a_id)
VALUES (4,"Street1",NULL,2);
```

- Let see our tables whose values have just been added via these two command:

```
SELECT * FROM Administrator;
SELECT * FROM Warehouse;
```

· These are tuples in 2 tables:

Success
(6 rows) 0.128 seconds

Explore

admin_id	admin_name
1	A
2	B
3	
4	Cam
5	Dung
6	Duong

Success
(4 rows) 0.074 seconds

Explore SQL

warehouse_id	address	floor	a_id
1	Street1	5	1
2	street2	1	1
3	Street1	1	2
4	Street1	NULL	2

- Finally we retrieve data from Administrator and Warehouse tables with the help of foreign key to check if it work or not:

```
SELECT
    Warehouse.a_id, Warehouse.warehouse_id,
    ↪  Administrator.admin_id,Administrator.admin_name
FROM
    Warehouse
INNER JOIN
    Administrator
ON
    Warehouse.a_id=Administrator.admin_id
WHERE
    Administrator.admin_id < Warehouse.warehouse_id;

-- retrieve some warehouse's information and its admin's name, id , with
↪  condition admin_id < warehouse's id
```

· We can see the result below: