Lab 7 – Review on T-SQL

Content:

- Design relational database schema
- Create a BikeStores database
- Create database diagram
- Answer all queries

Duration: 4 teaching periods

Learning outcome:

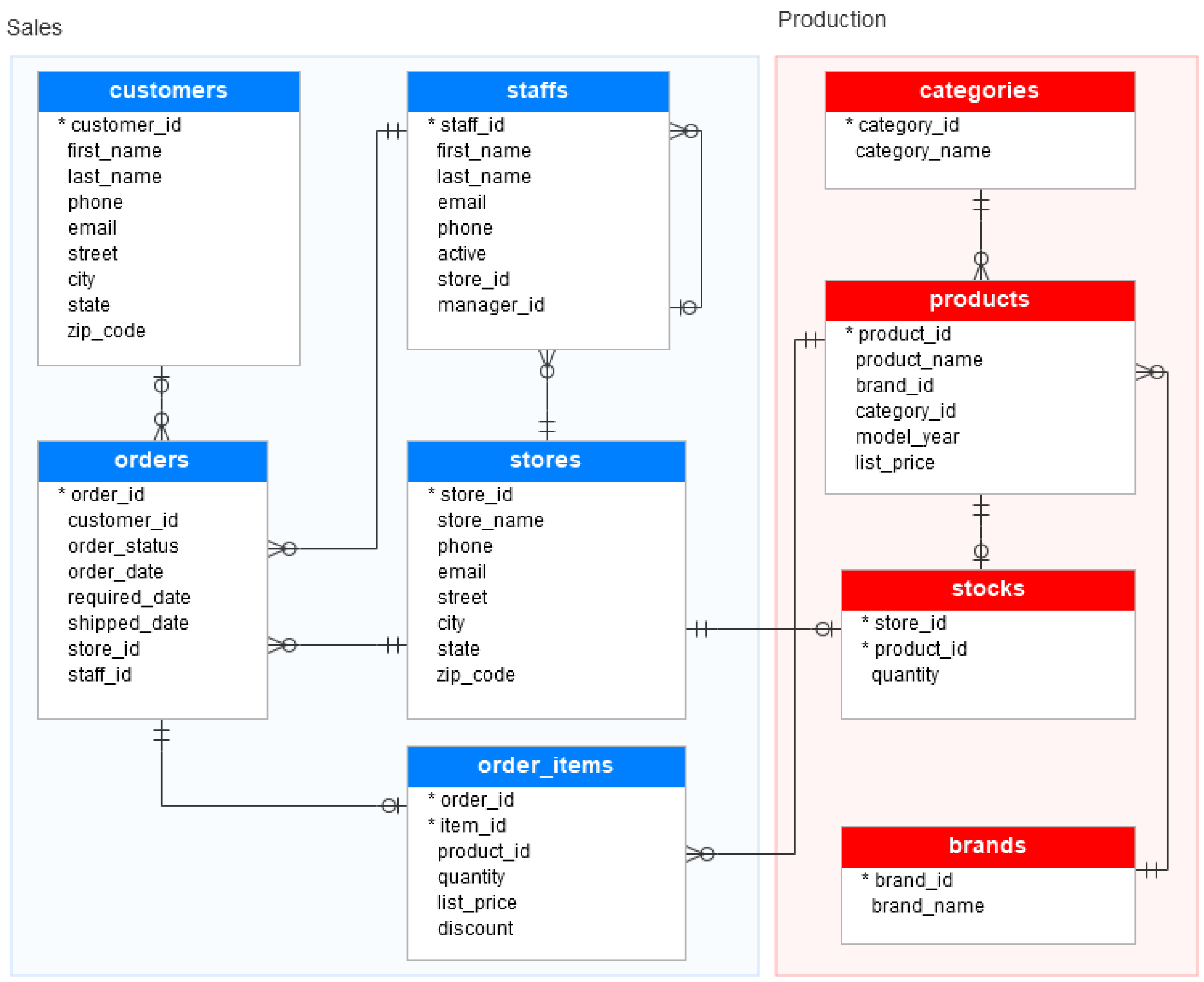
- How to create and manage database
- How to create constraints
- Database design for a practical problem and managing database diagrams in Microsoft SQL Server
- How to answer queries

Part 1: Creating a BikeStores database

Create a **BikeStores** database via the website.

http://www.sqlservertutorial.net/sql-server-sample-database/

The following illustrates the BikeStores database diagram:



As you can see from the diagram, the BikeStores sample database has two schemas sales and production, and these schemas have nine tables.

Database Tables

Table sales.stores

The sales.stores table includes the store's information. Each store has a store name, contact information such as phone and email, and an address including street, city, state, and zip code.

```
CREATE TABLE sales.stores (
        store_id INT IDENTITY (1, 1) PRIMARY KEY,
        store_name VARCHAR (255) NOT NULL,
        phone VARCHAR (25),
        email VARCHAR (255),
        street VARCHAR (255),
        city VARCHAR (255),
        state VARCHAR (10),
        zip_code VARCHAR (5)
);
```

Table sales.staffs

The sales.staffs table stores the essential information of staffs including first name, last name. It also contains the communication information such as email and phone.

A staff works at a store specified by the value in the store_id column. A store can have one or more staffs.

A staff reports to a store manager specified by the value in the manager_id column. If the value in the manager_id is null, then the staff is the top manager.

If a staff no longer works for any stores, the value in the active column is set to zero.

```
CREATE TABLE sales.staffs (
        staff_id INT IDENTITY (1, 1) PRIMARY KEY,
        first_name VARCHAR (50) NOT NULL,
        last_name VARCHAR (50) NOT NULL,
        email VARCHAR (255) NOT NULL UNIQUE,
        phone VARCHAR (25),
        active tinyint NOT NULL,
        store_id INT NOT NULL,
        manager_id INT,
        FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON
DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (manager_id) REFERENCES sales.staffs (staff_id) ON
DELETE NO ACTION ON UPDATE NO ACTION
);
```

Table production.categories

The production.categories table stores the bike's categories such as children bicycles, comfort bicycles, and electric bikes.

```
CREATE TABLE production.categories (
        category_id INT IDENTITY (1, 1) PRIMARY KEY,
        category_name VARCHAR (255) NOT NULL
);
```

Table production.brands

The production.brands table stores the brand's information of bikes, for example, Electra, Haro, and Heller.

```
CREATE TABLE production.brands (
	brand_id INT IDENTITY (1, 1) PRIMARY KEY,
	brand_name VARCHAR (255) NOT NULL
);
```

Table production.products

The production.products table stores the product's information such as name, brand, category, model year, and list price.

Each product belongs to a brand specified by the brand_id column. Hence, a brand may have zero or many products.

Each product also belongs a category specified by the category_id column. Also, each category may have zero or many products.

```
CREATE TABLE production.products (
	product_id INT IDENTITY (1, 1) PRIMARY KEY,
	product_name VARCHAR (255) NOT NULL,
	brand_id INT NOT NULL,
	category_id INT NOT NULL,
	model_year SMALLINT NOT NULL,
	list_price DECIMAL (10, 2) NOT NULL,
	FOREIGN KEY (category_id) REFERENCES production.categories
(category_id) ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (brand_id) REFERENCES production.brands (brand_id)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table sales.customers

The sales.customers table stores customer's information including first name, last name, phone, email, street, city, state and zip code.

```
CREATE TABLE sales.customers (
	customer_id INT IDENTITY (1, 1) PRIMARY KEY,
	first_name VARCHAR (255) NOT NULL,
	last_name VARCHAR (255) NOT NULL,
	phone VARCHAR (25),
	email VARCHAR (255) NOT NULL,
	street VARCHAR (255),
	city VARCHAR (50),
	state VARCHAR (25),
	zip_code VARCHAR (5)
);
```

Table sales.orders

The sales.orders table stores the sales order's header information including customer, order status, order date, required date, shipped date.

It also stores the information on where the sales transaction created (store) and who created it (staff).

Each sales order has a row in the sales_orders table. A sales order has one or many line items stored in the sales.order_items table.

```
CREATE TABLE sales.orders (
	order_id INT IDENTITY (1, 1) PRIMARY KEY,
	customer_id INT,
	order_status tinyint NOT NULL,
	-- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed
	order_date DATE NOT NULL,
	required_date DATE NOT NULL,
	shipped_date DATE,
	store_id INT NOT NULL,
	staff_id INT NOT NULL,
	FOREIGN KEY (customer_id) REFERENCES sales.customers
(customer_id) ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON
DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (staff_id) REFERENCES sales.staffs (staff_id) ON
DELETE NO ACTION ON UPDATE NO ACTION
);
```

Table sales.order_items

The sales.order_items table stores the line items of a sales order. Each line item belongs to a sales order specified by the order_id column.

A sales order line item includes product, order quantity, list price and discount.

```
CREATE TABLE sales.order_items (
	order_id INT,
	item_id INT,
	product_id INT NOT NULL,
	quantity INT NOT NULL,
        list_price DECIMAL (10, 2) NOT NULL,
        discount DECIMAL (4, 2) NOT NULL DEFAULT 0,
        PRIMARY KEY (order_id, item_id),
        FOREIGN KEY (order_id) REFERENCES sales.orders (order_id) ON
DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (product_id) REFERENCES production.products
(product_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table production.stocks

The production.stocks table stores the inventory information i.e. the quantity of a particular product in a specific store.

```
CREATE TABLE production.stocks (
        store_id INT,
        product_id INT,
        quantity INT,
        PRIMARY KEY (store_id, product_id),
        FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON
DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (product_id) REFERENCES production.products
(product_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Part 2: Diagramming the Database

Creating a database diagram based on BikeStores database

Part 3: Answering all the queries

1. Finds the top 10 most expensive products follow list_price descending order.
2. Finds the customer id and the ordered year (year(order_date)) of the customers with the customer id one and two follow customer_id order
3. Finds the number of orders placed by the customer by year
4. Finds the number of customers in every city
5. Finds the number of customers by state and city follow City and State order
6. Finds the minimum and maximum list prices of all products with the model 2018 by brand follow brand_name order

7. Finds the average list price by brand for all products with the model year 2018 follow brand_name order
8. Finds the customers who placed at least two orders per year follow Customer_id order
9. Finds the sales orders whose net values are greater than 20,000
10. Finds product categories whose average list prices are between 500 and 1,000
11. Find the sales orders of the customers (order_id, order_date, customer_id) who locate in New York follow order_date descending order.
12. Finds the names of all mountain bikes and road bikes products that the Bike Stores sell (used subquery)
13. Finds the products whose list prices are greater than or equal to the maximum list price of any product brand
14. Finds the products whose list price is greater than or equal to the maximum list price returned by the subquery
15. Finds the customers who bought products in 2017 (used subquery) follow by first_name, last_name order
16. Finds the customers who did not buy any products in 2017
17. Finds the sales amount grouped by brand and category
18. Finds the sales amount by brand. It defines a grouping set (brand)
19. Finds the sales amount by category. It defines a grouping set (category)
20. Sorts the customers by the city in descending order and the sort the sorted result set by the first name in ascending order
21. Finds a customer list sorted by the length of the first name