ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO MÔN HỆ CƠ SỞ DỮ LIỆU

HỆ THỐNG GỬI XE MÁY TỰ ĐỘNG

NHÓM 5 – LỚP CN01 – HK241

Giảng viên hướng dẫn: Võ Thị Kim Anh

SINH VIÊN THỰC HIỆN

Võ Lý Đắc Duy – 2252125

Phan Thảo Vy – 2252930

Võ Văn Hiếu – 2252219

Thành phố Hồ Chí Minh – 12/2024

I. Giới thiệu về DBMS nhóm đã lựa chọn

Để xây dựng hệ cơ sở dữ liệu quản lý bãi giữ xe tự động, nhóm quyết định lựa chọn DBMS là MySQL. Đây là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS - Relational Database Management System) mã nguồn mở, được phát triển bởi Oracle Corporation. MySQL sử dụng ngôn ngữ SQL (Structured Query Language) để quản lý và thao tác dữ liệu, hỗ trợ các tính năng mạnh mẽ như giao dịch, bảo mật, và khả năng mở rộng. Đặc biệt, MySQL được sử dụng rộng rãi trong các ứng dụng web và doanh nghiệp nhờ vào hiệu suất cao, tính linh hoạt và khả năng tích hợp dễ dàng với các ngôn ngữ lập trình như PHP, Python, Java, và nhiều ngôn ngữ khác.

Một số đặc điểm nổi bật của MySQL:

- Hiệu năng cao: Hỗ trợ xử lý khối lượng lớn dữ liệu nhanh chóng.
- Dễ sử dụng: Cung cấp giao diện đơn giản và nhiều công cụ hỗ trợ.
- Khả năng mở rộng: Phù hợp từ các ứng dụng nhỏ đến hệ thống lớn.
- Tính bảo mật: Cung cấp cơ chế xác thực người dùng và quản lý quyền truy cập chặt chẽ.
- Đa nền tảng: Chạy tốt trên nhiều hệ điều hành như Windows, macOS, Linux.

Các bước cài đặt cấu hình MySQL:

- Bước 1: Tải xuống MySQL. Truy cập trang web chính thức của MySQL: MySQL Downloads.
- Bước 2: Cài đặt MySQL. Chọn **Developer Default** để cài đặt đầy đủ server, client, và các công cụ phát triển
- Bước 3: Xác nhận các thành phần cần cài đặt và nhấn **Execute** để tiến hành.
- Bước 4: Sau khi cài đặt, tùy chỉnh MySQL bằng cách:
  + Chọn kiểu cấu hình (Standalone MySQL Server hoặc Clustered).
  + Đặt mật khẩu cho tài khoản root và tạo thêm tài khoản người dùng (nếu cần).
  + Cấu hình cổng (mặc định: 3306) và dịch vụ Windows cho MySQL.
- Bước 5: Cấu hình MySQL
- Bước 6: Kiểm tra và kết nối MySQL
  + Đăng nhập MySQL
  + Tạo cơ sở dữ liệu mới bằng lệnh: CREATE SCHEMA CSDL_TN;

```
1 •  CREATE SCHEMA CSDL_TN;
```

II. Các câu lệnh SQL để tạo CSDL và các bảng

Sau khi tạo cơ sở dữ liệu, sử dụng lệnh USE CSDL_TN để truy cập vào schema:

```
3 •  USE CSDL_TN;
```

Tiếp theo là tạo các bảng tương ứng với các thực thể đã được đưa ra bao gồm:

- PERSON

```
4 •  CREATE TABLE PERSON (
5        SSN VARCHAR(12) PRIMARY KEY,
6        FIRSTNAME VARCHAR(50) NOT NULL,
7        MIDNAME VARCHAR(50),
8        LASTNAME VARCHAR(50) NOT NULL,
9        DOB DATE NOT NULL,
10       EMAIL VARCHAR(100) NOT NULL,
11       SEX CHAR(1) NOT NULL,
12       ADDRESS VARCHAR(255) NOT NULL
13   );
```

- MANAGER

```
14 •  CREATE TABLE MANAGER (
15        MANAGER_ID VARCHAR(10),
16        MNG_SSN VARCHAR(12),
17        START_WORKING_DAY DATE NOT NULL,
18        MNG_START_TIME DATETIME NOT NULL,
19        PRIMARY KEY (MNG_SSN),
20        FOREIGN KEY (MNG_SSN) REFERENCES PERSON(SSN)
21    );
```

- PARKING_LOT

```
24 •  CREATE TABLE PARKING_LOT (
25        PARKINGLOT_ID VARCHAR(10) PRIMARY KEY,
26        CAPACITY INT NOT NULL,
27        ADDRESS VARCHAR(255) NOT NULL,
28        PARKINGLOT_NAME VARCHAR(100),
29        MNG_SSN  VARCHAR(12),
30        FOREIGN KEY (MNG_SSN) REFERENCES MANAGER(MNG_SSN)
31    );
```

- CUSTOMER

```sql
CREATE TABLE CUSTOMER (
    CUS_ID VARCHAR(10) NOT NULL,
    CUS_SSN VARCHAR(12),
    PRIMARY KEY (CUS_SSN),
    FOREIGN KEY (CUS_SSN) REFERENCES PERSON(SSN)
);
```

- RESPONSE

```sql
CREATE TABLE RESPONSE (
    RESPONSE_ID VARCHAR(10) PRIMARY KEY,
    STATUS_RE VARCHAR(50) NOT NULL,
    TYPE_RE VARCHAR(100) NOT NULL,
    TITLE_RE VARCHAR(200) NOT NULL,
    MESSAGE TEXT NOT NULL,
    MNG_SSN  VARCHAR(12),
    CU_SSN  VARCHAR(12) NOT NULL,
    TIME_RE DATETIME NOT NULL,
    RATING INT,
    CAUSE TEXT,
    EVIDENCE TEXT,
    FOREIGN KEY (MNG_SSN) REFERENCES MANAGER(MNG_SSN),
    FOREIGN KEY (CU_SSN) REFERENCES CUSTOMER(CUS_SSN)
);
```

- MOTORBIKE

```sql
CREATE TABLE MOTORBIKE (
    LICENSE_PLATE VARCHAR(20) PRIMARY KEY,
    COLOR VARCHAR(20),
    BRAND VARCHAR(50),
    TYPE VARCHAR(50),
    CUS_SSN VARCHAR(12),
    MONTHLY_PART DATETIME,
    FOREIGN KEY (CUS_SSN) REFERENCES CUSTOMER(CUS_SSN)
);
```

- EMPTY_SLOT

```sql
CREATE TABLE EMPTY_SLOT(
    POSITION_NUMBER INT NOT NULL,
    PKL_ID VARCHAR(10),
    ISEMPTY BOOLEAN NOT NULL DEFAULT TRUE,
    PRIMARY KEY (POSITION_NUMBER, PKL_ID),
    FOREIGN KEY (PKL_ID) REFERENCES PARKING_LOT(PARKINGLOT_ID)
);
```

- TICKET

```sql
CREATE TABLE TICKET (
    TICKET_ID VARCHAR(10) PRIMARY KEY,
    TICKER_CODE VARCHAR(20),
    TICKET_STATUS VARCHAR(50) NOT NULL,
    LICENSE_PLATE VARCHAR(20) NOT NULL,
    PARKINGLOT_ID VARCHAR(10) NOT NULL,
    EMPTY_NUM INT NOT NULL,
    CHECKOUT_TIME DATETIME,
    CHECKIN_TIME DATETIME NOT NULL,
    FOREIGN KEY (EMPTY_NUM, PARKINGLOT_ID) REFERENCES EMPTY_SLOT(POSITION_NUMBER, PKL_ID),
    FOREIGN KEY (LICENSE_PLATE) REFERENCES MOTORBIKE(LICENSE_PLATE)
);
```

- ACCOUNT

```sql
CREATE TABLE ACCOUNT (
    ACCOUNT_ID VARCHAR(10) PRIMARY KEY,
    USERNAME VARCHAR(50) NOT NULL,
    PASSWORD VARCHAR(50) NOT NULL
);
```

- CUSTOMER_ACCOUNT

```sql
CREATE TABLE CUSTOMER_ACCOUNT (
    CUS_ACC_ID VARCHAR(10) PRIMARY KEY,
    CUS_SSN VARCHAR(12),
    CREATION_DATE DATE,
    QR_CODE VARCHAR(20),
    FOREIGN KEY (CUS_SSN) REFERENCES CUSTOMER(CUS_SSN),
    FOREIGN KEY (CUS_ACC_ID) REFERENCES ACCOUNT(ACCOUNT_ID)
);
```

- BANK_ACCOUNT

```sql
CREATE TABLE BANK_ACCOUNT (
    BANK_ACC_NUMBER VARCHAR(16) PRIMARY KEY,
    ACCOUNT_NAME VARCHAR(100) NOT NULL,
    BALANCE DECIMAL(10, 2) NOT NULL,
    BRANCH VARCHAR(100),
    BANK_NAME VARCHAR(100),
    CUS_ACC_ID VARCHAR(10) NOT NULL,
    FOREIGN KEY (CUS_ACC_ID) REFERENCES CUSTOMER_ACCOUNT(CUS_ACC_ID)
);
```

- PAYMENT

```sql
CREATE TABLE PAYMENT (
    TRANSACTION_ID VARCHAR(10) PRIMARY KEY,
    TIC_ID VARCHAR(10),
    PAYMENT_METHOD VARCHAR(50) NOT NULL,
    PAYMENT_TIME DATETIME,
    TOTAL_AMOUNT DECIMAL(10, 2) NOT NULL,
    BANK_NUM VARCHAR(16) DEFAULT NULL,
    PAY_STATUS VARCHAR(10)  NOT NULL,
    FOREIGN KEY (BANK_NUM) REFERENCES BANK_ACCOUNT(BANK_ACC_NUMBER),
    FOREIGN KEY (TIC_ID) REFERENCES TICKET(TICKET_ID)
);
```

- NOTIFICATION

```sql
CREATE TABLE NOTIFICATION (
    NOTI_ID VARCHAR(10) PRIMARY KEY,
    NOTI_TITLE VARCHAR(255) NOT NULL,
    NOTI_MESSAGE TEXT,
    NOTI_TIME DATETIME,
    NOTI_TYPE VARCHAR(50),
    NOTI_STATUS VARCHAR(50),
    CUS_SSN VARCHAR(12),
    FOREIGN KEY (CUS_SSN) REFERENCES CUSTOMER(CUS_SSN)
);
```

- MANAGER_ACCOUNT

```sql
CREATE TABLE MANAGER_ACCOUNT (
    MNG_ACC_ID VARCHAR(10) PRIMARY KEY,
    MNG_SSN VARCHAR(12),
    PERMISSION VARCHAR(50),
    FOREIGN KEY (MNG_SSN) REFERENCES MANAGER(MNG_SSN),
    FOREIGN KEY (MNG_ACC_ID) REFERENCES ACCOUNT(ACCOUNT_ID)
);
```

- PHONE

```sql
CREATE TABLE PHONE (
    OWNER_SSN VARCHAR(12),
    PHONE_NUM VARCHAR(10) NOT NULL,
    PRIMARY KEY (OWNER_SSN, PHONE_NUM),
    FOREIGN KEY (OWNER_SSN) REFERENCES PERSON(SSN)
);
```

- WARNING_NOTIFICATION

```sql
CREATE TABLE WARNING_NOTIFICATION(
    NOTIFICATION_ID VARCHAR(10) PRIMARY KEY,
    PRIORITY VARCHAR(20),
    MNG_SSN VARCHAR(20),
    FOREIGN KEY (NOTIFICATION_ID) REFERENCES NOTIFICATION(NOTI_ID),
    FOREIGN KEY (MNG_SSN) REFERENCES MANAGER(MNG_SSN)
);
```

- REMINDER_NOTIFICATION

```sql
CREATE TABLE REMINDER_NOTIFICATION(
    NOTIFICATION_ID VARCHAR(10) PRIMARY KEY,
    EXPIRATION_TIME DATETIME,
    LINK VARCHAR(255),
    FOREIGN KEY (NOTIFICATION_ID) REFERENCES NOTIFICATION(NOTI_ID)
);
```

- PAYMENT_NOTIFICATION

```sql
CREATE TABLE PAYMENT_NOTIFICATION(
    NOTIFICATION_ID VARCHAR(10) PRIMARY KEY,
    TRANSACTION_STATUS VARCHAR(20),
    PAYMENT_DUE_TIME VARCHAR(5),
    TRANSACTION_ID VARCHAR(10),
    FOREIGN KEY (NOTIFICATION_ID) REFERENCES NOTIFICATION(NOTI_ID),
    FOREIGN KEY (TRANSACTION_ID) REFERENCES PAYMENT(TRANSACTION_ID)
);
```

Sau khi khởi tạo, DBMS thông báo các bảng đã được tạo thành công:

10	17:10:45	CREATE SCHEMA CSDL_TN	1 row(s) affected
11	17:10:45	USE CSDL_TN	0 row(s) affected
12	17:11:30	CREATE TABLE PERSON (SSN VARCHAR(12) PRIMARY KEY, FIRSTNAME VARCHAR(50) NOT NUL...	0 row(s) affected
13	17:11:30	CREATE TABLE MANAGER (MANAGER_ID VARCHAR(10), MNG_SSN VARCHAR(12), START_WO...	0 row(s) affected
14	17:11:30	CREATE TABLE PARKING_LOT (PARKINGLOT_ID VARCHAR(10) PRIMARY KEY, CAPACITY INT NO...	0 row(s) affected
15	17:11:30	CREATE TABLE CUSTOMER (CUS_ID VARCHAR(10) NOT NULL, CUS_SSN VARCHAR(12), PRIMA...	0 row(s) affected
16	17:11:30	CREATE TABLE RESPONSE (RESPONSE_ID VARCHAR(10) PRIMARY KEY, STATUS_RE VARCHAR(...	0 row(s) affected
17	17:11:30	CREATE TABLE MOTORBIKE (LICENSE_PLATE VARCHAR(20) PRIMARY KEY, COLOR VARCHAR(20...	0 row(s) affected
18	17:11:30	CREATE TABLE TICKET (TICKET_ID VARCHAR(10) PRIMARY KEY, TICKER_CODE VARCHAR(20), ...	0 row(s) affected
19	17:11:30	CREATE TABLE ACCOUNT (ACCOUNT_ID VARCHAR(10) PRIMARY KEY, USERNAME VARCHAR(50)...	0 row(s) affected
20	17:11:30	CREATE TABLE CUSTOMER_ACCOUNT (CUS_ACC_ID VARCHAR(10) PRIMARY KEY, CUS_SSN VA...	0 row(s) affected
21	17:11:30	CREATE TABLE BANK_ACCOUNT (BANK_ACC_NUMBER VARCHAR(16) PRIMARY KEY, ACCOUNT_...	0 row(s) affected
22	17:11:30	CREATE TABLE PAYMENT (TRANSACTION_ID INT AUTO_INCREMENT PRIMARY KEY, TIC_ID VAR...	0 row(s) affected
23	17:11:30	CREATE TABLE NOTIFICATION (NOTI_ID VARCHAR(10) PRIMARY KEY, NOTI_TITLE VARCHAR(255...	0 row(s) affected
24	17:11:30	CREATE TABLE MANAGER_ACCOUNT (MNG_ACC_ID VARCHAR(10) PRIMARY KEY, MNG_SSN VA...	0 row(s) affected
25	17:11:30	CREATE TABLE EMPTY_SLOT(POSITION_NUMBER INT NOT NULL, PKL_ID VARCHAR(10), PRIM...	0 row(s) affected
26	17:11:30	CREATE TABLE PARK(LICENSE_PLATE VARCHAR(20) NOT NULL, PARKINGLOT_ID VARCHAR(10) N...	0 row(s) affected
27	17:11:30	CREATE TABLE PHONE (OWNER_SSN VARCHAR(12), PHONE_NUM VARCHAR(10) NOT NULL, P...	0 row(s) affected
28	17:11:30	CREATE TABLE FEEDBACK(RESPONSE_ID VARCHAR(10) PRIMARY KEY, RATING INT, FOREIGN...	0 row(s) affected
29	17:11:30	CREATE TABLE ISSUE(RESPONSE_ID VARCHAR(10) PRIMARY KEY, CAUSE TEXT, FOREIGN KE...	0 row(s) affected
30	17:11:30	CREATE TABLE COMPLAINT(RESPONSE_ID VARCHAR(10) PRIMARY KEY, EVIDENCE TEXT, FOREIG...	0 row(s) affected
31	17:11:30	CREATE TABLE WARNING_NOTIFICATION(NOTIFICATION_ID VARCHAR(10) PRIMARY KEY, PRIOR...	0 row(s) affected
32	17:11:30	CREATE TABLE REMINDER_NOTIFICATION(NOTIFICATION_ID VARCHAR(10) PRIMARY KEY, EXPI...	0 row(s) affected
33	17:11:30	CREATE TABLE PAYMENT_NOTIFICATION(NOTIFICATION_ID VARCHAR(10) PRIMARY KEY, TRAN...	0 row(s) affected

Cuối cùng là tạo dữ liệu giả cho hệ cơ sở dữ liệu để có thể thực hiện các thao tác đối với dữ liệu ở phần sau.

III. Thao tác với dữ liệu

A. LỊCH SỬ

TRUY VẤN LỊCH SỬ GỬI XE

Đây là một Store Procedure để lấy thông tin lịch sử gửi xe của tài khoản người dùng hiện tại. Các trường thông tin của lịch sử là: mã vé, thời gian gửi, thời gian kết thúc, tổng số giờ gửi, biển số xe, tên bãi xe và vị trí gửi.

```
DELIMITER //
CREATE PROCEDURE GetParkingHistory(IN customerAccountID VARCHAR(10))
BEGIN
    SELECT
        t.TICKET_ID AS `Ticket ID`,
        t.CHECKIN_TIME AS `Check-in Time`,
        t.CHECKOUT_TIME AS `Check-out Time`,
        CONCAT(
            FLOOR(TIMESTAMPDIFF(MINUTE, t.CHECKIN_TIME, t.CHECKOUT_TIME) / 60), ' giờ ',
            MOD(TIMESTAMPDIFF(MINUTE, t.CHECKIN_TIME, t.CHECKOUT_TIME), 60), ' phút'
        ) AS `Duration`,
        m.LICENSE_PLATE AS `License Plate`,
        p.PARKINGLOT_NAME AS `Parking Lot`,
        t.EMPTY_NUM AS `Slot Number`
    FROM TICKET t
    JOIN MOTORBIKE m ON t.LICENSE_PLATE = m.LICENSE_PLATE
    JOIN CUSTOMER c ON m.CUS_SSN = c.CUS_SSN
    JOIN CUSTOMER_ACCOUNT ca ON c.CUS_SSN = ca.CUS_SSN
    JOIN PARKING_LOT p ON t.PARKINGLOT_ID = p.PARKINGLOT_ID
    WHERE ca.CUS_ACC_ID = customerAccountID AND t.TICKET_STATUS = 'Expired'
    ORDER BY t.CHECKIN_TIME DESC;
END //
DELIMITER ;
```

Để tiến hành truy xuất lịch sử gửi xe ví dụ như của tài khoản người dùng có id là A008, ta thực thi:

```
CALL GetParkingHistory('A008');
```

Kết quả như sau:

Ticket ID	Check-in Time	Check-out Time	Duration	License Plate	Parking Lot	Slot Number
T003	2024-11-29 19:21:00	2024-11-29 22:13:00	2 giờ 52 phút	LP003	Lot D	4
T007	2024-04-22 10:08:00	2024-04-22 13:43:00	3 giờ 35 phút	LP005	Lot E	22
T006	2023-06-05 08:56:00	2023-06-05 12:31:00	3 giờ 35 phút	LP005	Lot E	5

TRUY VẤN LỊCH SỬ THANH TOÁN

Đây là một Store Procedure để lấy thông tin lịch sử thanh toán của tài khoản người dùng hiện tại. Các trường thông tin của lịch sử là: mã thanh toán, thời gian thanh toán, tổng số tiền thanh toán, tên ngân hàng và số tài khoản ngân hàng.

```
DELIMITER //
CREATE PROCEDURE GetPaymentHistory(IN customerAccountID VARCHAR(10))
BEGIN
    SELECT
        p.TRANSACTION_ID AS `Transaction ID`,
        p.PAYMENT_TIME AS `Payment Time`,
        p.TOTAL_AMOUNT AS `Amount`,
        b.BANK_NAME AS `Bank Name`,
        b.BANK_ACC_NUMBER AS `Bank Account Number`
    FROM PAYMENT p
    JOIN BANK_ACCOUNT b ON p.BANK_NUM = b.BANK_ACC_NUMBER
    JOIN CUSTOMER_ACCOUNT ca ON b.CUS_ACC_ID = ca.CUS_ACC_ID
    WHERE ca.CUS_ACC_ID = customerAccountID AND p.PAY_STATUS = 'Paid'
    ORDER BY p.PAYMENT_TIME DESC;
END //
DELIMITER ;
```

Để tiến hành truy xuất lịch sử thanh toán ví dụ như của tài khoản người dùng có id là A008, ta thực thi:

```
CALL GetPaymentHistory('A008');
```

Kết quả như sau:

	Transaction ID	Payment Time	Amount	Bank Name	Bank Account Number
▶	3	2023-07-03 00:00:00	8000.00	Bank C	BA003
	6	2023-07-03 00:00:00	12000.00	Bank A	BA006
	7	2023-07-03 00:00:00	18000.00	Bank A	BA006

B. THỐNG KÊ

TRUY VẤN THỐNG KÊ

Đây là một Store Procedure để lấy thông tin thống kê của từng bãi xe theo năm. Các trường thông tin bao gồm: tên bãi xe, tổng số xe, tổng số vé xe và tổng số tiền thu được.

```
DELIMITER //
CREATE PROCEDURE GetParkingLotStatisticsByYear(IN year INT)
BEGIN
    SELECT
        p.PARKINGLOT_NAME AS `Parking Lot Name`,
        COUNT(DISTINCT m.LICENSE_PLATE) AS `Number of Vehicles`,
        COUNT(t.TICKET_ID) AS `Number of Tickets`,
        IFNULL(SUM(pay.TOTAL_AMOUNT), 0) AS `Total Payment Amount`
    FROM PARKING_LOT p
    LEFT JOIN TICKET t ON p.PARKINGLOT_ID = t.PARKINGLOT_ID AND YEAR(t.CHECKIN_TIME) = year AND t.TICKET_STATUS = 'Expired'
    LEFT JOIN MOTORBIKE m ON t.LICENSE_PLATE = m.LICENSE_PLATE
    LEFT JOIN PAYMENT pay ON t.TICKET_ID = pay.TIC_ID AND pay.PAY_STATUS = 'Paid' AND YEAR(pay.PAYMENT_TIME) = year
    GROUP BY p.PARKINGLOT_NAME
    ORDER BY p.PARKINGLOT_NAME;
END //
DELIMITER ;
```

Để tiến hành truy xuất thống kê ví dụ như của năm 2023 và năm 2024, ta thực thi:

```
CALL GetParkingLotStatisticsByYear(2023);
CALL GetParkingLotStatisticsByYear(2024);
```

Kết quả như sau:

	Parking Lot Name	Number of Vehicles	Number of Tickets	Total Payment Amount
▶	Lot A	1	1	6000.00
	Lot B	1	1	12000.00
	Lot C	1	1	4000.00
	Lot D	0	0	0.00
	Lot E	2	2	22000.00

	Parking Lot Name	Number of Vehicles	Number of Tickets	Total Payment Amount
▶	Lot A	0	0	0.00
	Lot B	0	0	0.00
	Lot C	0	0	0.00
	Lot D	1	1	8000.00
	Lot E	1	1	18000.00

C. KHÁCH HÀNG

TRUY VẤN THÔNG TIN CÁ NHÂN

Đây là một câu lệnh truy vấn các thông tin cá nhân của khách hàng bao gồm: mã khách hàng, số định danh, tên, ngày sinh, mail, giới tính, địa chỉ và số điện thoại.

```
SELECT
    c.CUS_ID,
    p.SSN,
    CONCAT(p.FIRSTNAME, ' ', p.MIDNAME, ' ', p.LASTNAME) AS `Full Name`,
    p.DOB,
    p.EMAIL,
    p.SEX,
    p.ADDRESS,
    ph.PHONE_NUM AS `Phone Number`
FROM CUSTOMER_ACCOUNT ca
JOIN CUSTOMER c ON ca.CUS_SSN = c.CUS_SSN
JOIN PERSON p ON c.CUS_SSN = p.SSN
LEFT JOIN PHONE ph ON p.SSN = ph.OWNER_SSN
WHERE ca.CUS_ACC_ID = 'A009';
```

Như trên ta thực hiện truy vấn thông tin cá nhân của tài khoản có id là A009.

Kết quả như sau:

	CUS_ID	SSN	Full Name	DOB	EMAIL	SEX	ADDRESS	Phone Number
▶	C004	999999999999	Grace I Taylor	1978-09-09	grace.taylor@example.com	F	606 Ash St	3333901234

TRUY VẤN THÔNG TIN TÀI KHOẢN

Đây là một câu lệnh truy vấn thông tin tài khoản của khách hàng bao gồm: mã tài khoản, tên đăng nhập, mật khẩu, QR code và ngày tạo tài khoản.

```
SELECT
    ca.CUS_ACC_ID AS `Customer Account ID`,
    acc.USERNAME AS `Username`,
    acc.PASSWORD AS `Password`,
    ca.QR_CODE AS `QR Code`,
    ca.CREATION_DATE AS `Creation Date`
FROM CUSTOMER_ACCOUNT ca
JOIN ACCOUNT acc ON ca.CUS_ACC_ID = acc.ACCOUNT_ID
WHERE ca.CUS_ACC_ID = 'A009';
```

Như trên ta thực hiện truy vấn thông tin tài khoản có id là A009.

Kết quả như sau:

	Customer Account ID	Username	Password	QR Code	Creation Date
▶	A009	user9	pass9	QR004	2023-04-01

TRUY VẤN DANH SÁCH THÔNG BÁO

Đây là câu lệnh truy vấn các thông báo cảnh báo được gửi từ quản lý. Gồm các thông tin như: tiêu đề, nội dung, thời gian, loại, mức độ nghiêm trọng ... và tên quản lý đã gửi thông báo.

```
SELECT
    n.*,
    wn.PRIORITY,
    CONCAT(p.FIRSTNAME, ' ', p.MIDNAME, ' ', p.LASTNAME) AS `Send Manager`
FROM WARNING_NOTIFICATION wn
JOIN NOTIFICATION n ON wn.NOTIFICATION_ID = n.NOTI_ID
JOIN MANAGER m ON wn.MNG_SSN = m.MNG_SSN
JOIN PERSON p ON m.MNG_SSN = p.SSN
JOIN CUSTOMER c ON n.CUS_SSN = c.CUS_SSN
JOIN CUSTOMER_ACCOUNT ca ON c.CUS_SSN = ca.CUS_SSN
WHERE ca.CUS_ACC_ID = 'A009';
```

Như trên ta thực hiện truy vấn thông báo cảnh cáo của tài khoản có id là A009.

Kết quả như sau:

	NOTI_ID	NOTI_TITLE	NOTI_MESSAGE	NOTI_TIME	NOTI_TYPE	NOTI_STATUS	CUS_SSN	PRIORITY	Send Manager
▶	N004	Electric Issue	Short circuit at A2! Please careful	2023-06-04 11:00:00	Warning	viewed	999999999999	High	Bob D Brown

Đây là câu lệnh truy vấn các thông báo thanh toán. Gồm các thông tin như: tiêu đề, nội dung, thời gian, loại, trạng thái thanh toán,... và thời gian tới hạn thanh toán.

```
SELECT
    pn.*,
    n.NOTI_TITLE AS `Notification Title`,
    n.NOTI_MESSAGE AS `Notification Message`,
    n.NOTI_TIME AS `Notification Time`,
    n.NOTI_TYPE AS `Notification Type`,
    n.NOTI_STATUS AS `Notification Status`
FROM PAYMENT_NOTIFICATION pn
JOIN NOTIFICATION n ON pn.NOTIFICATION_ID = n.NOTI_ID
JOIN CUSTOMER c ON n.CUS_SSN = c.CUS_SSN
JOIN CUSTOMER_ACCOUNT ca ON c.CUS_SSN = ca.CUS_SSN
WHERE ca.CUS_ACC_ID = 'A006';
```

Như trên ta thực hiện truy vấn thông báo cảnh cáo của tài khoản có id là A006.

Kết quả như sau:

NOTIFICATION_ID	TRANSACTION_STATUS	PAYMENT_DUE_TIME	TRANSACTION_ID	Notification Title	Notification Message	Notification Time	Notification Type	Notification Status
N001	Paid	24h	1	Payment Completed	Your payment is completed. Do you have any q...	2023-06-01 08:00:00	Payment	Unread

TRUY VẤN DANH SÁCH PHẢN HỒI ĐÃ GỬI

Đây là câu lệnh truy vấn danh sách các phản hồi mà khách hàng đã gửi từ tài khoản của mình

```
SELECT r.*
FROM RESPONSE r
JOIN CUSTOMER c ON r.CU_SSN = c.CUS_SSN
JOIN CUSTOMER_ACCOUNT ca ON c.CUS_SSN = ca.CUS_SSN
WHERE ca.CUS_ACC_ID = 'A009';
```

Như trên ta thực hiện truy vấn danh sách với id là A009.

Kết quả như sau:

RESPONSE_ID	STATUS_RE	TYPE_RE	TITLE_RE	MESSAGE	MNG_SSN	CU_SSN	TIME_RE	RATING	CAUSE	EVIDENCE
R004	Resolved	Issue	Issue D	Message D	444444444444	999999999999	2023-04-01 13:00:00	NULL	Parking issue A	NULL

TẠO TÀI KHOẢN

```
DELIMITER //
CREATE PROCEDURE RegisterNewCustomer(
    IN newAccountID VARCHAR(10),
    IN newUsername VARCHAR(50),
    IN newPassword VARCHAR(50),
    IN ssn VARCHAR(12),
    IN firstName VARCHAR(50),
    IN midName VARCHAR(50),
    IN lastName VARCHAR(50),
    IN dob DATE,
    IN email VARCHAR(100),
    IN sex CHAR(1),
    IN address VARCHAR(255),
    IN phoneNum VARCHAR(10)
)
BEGIN
    -- Kiểm tra xem tài khoản đã tồn tại chưa
    IF (SELECT COUNT(*) FROM ACCOUNT WHERE ACCOUNT_ID = newAccountID) > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Tài khoản đã tồn tại';
    ELSE
        -- Thêm thông tin vào bảng PERSON
        INSERT INTO PERSON (SSN, FIRSTNAME, MIDNAME, LASTNAME, DOB, EMAIL, SEX, ADDRESS)
        VALUES (ssn, firstName, midName, lastName, dob, email, sex, address);
```

```
        -- Thêm tài khoản mới vào bảng ACCOUNT
        INSERT INTO ACCOUNT (ACCOUNT_ID, USERNAME, PASSWORD)
        VALUES (newAccountID, newUsername, newPassword);

        -- Thêm số điện thoại vào bảng PHONE
        INSERT INTO PHONE (OWNER_SSN, PHONE_NUM)
        VALUES (ssn, phoneNum);
    END IF;
END //

DELIMITER ;
```

Đây là một Store Procedure dùng để tạo tài khoản người dùng. Khi tạo thông tin cá nhân được tạo ở bảng PERSON và CUSTOMER, thông tin tài khoản được tạo ở bảng ACCOUNT và CUSTOMER_ACCOUNT.

Để thêm mẫu một tài khoản mới, ta thực thi như sau:

```
CALL RegisterNewCustomer('A014', 'user14', 'pass14', '123456789012', 'John', 'M', 'Doe',
'1990-01-01', 'john.doe@example.com', 'M', '123 Main St', '0912345678');
select *from ACCOUNT;
select*from PERSON;
```

Kết quả đạt được:

SSN	FIRSTNAME	MIDNAME	LASTNAME	DOB	EMAIL	SEX	ADDRESS
000000000000	Hank	J	Anderson	1981-10-10	hank.anderson@example.com	M	707 Elm St
111111111111	John	A	Doe	1980-01-01	john.doe@example.com	M	123 Main St
123456789012	John	M	Doe	1990-01-01	john.doe@example.com	M	123 Main St
222222222222	Jane	B	Smith	1990-02-02	jane.smith@example.com	F	456 Elm St
333333333333	Alice	C	Johnson	1985-03-03	alice.johnson@example.com	F	789 Oak St
444444444444	Bob	D	Brown	1975-04-04	bob.brown@example.com	M	101 Pine St
555555555555	Charlie	E	Williams	1995-05-05	charlie.williams@example.com	M	202 Cedar St
666666666666	David	F	Miller	1988-06-06	david.miller@example.com	M	303 Maple St

ACCOUNT_ID	USERNAME	PASSWORD
A005	user5	pass5
A006	user6	pass6
A007	user7	pass7
A008	user8	pass8
A009	user9	pass9
A010	user10	pass10
A014	user14	pass14
NULL	NULL	NULL

CUS_ID	CUS_SSN
C005	000000000000
C467	123456789012
C001	666666666666
C002	777777777777
C003	888888888888

CUS_ACC_ID	CUS_SSN	CREATION_DATE	QR_CODE
A008	888888888888	2023-03-01	QR003
A009	999999999999	2023-04-01	QR004
A010	000000000000	2023-05-01	QR005
A014	123456789012	2024-12-11	QR447
NULL	NULL	NULL	NULL

ĐỔI MẬT KHẨU

Đây là một Store Procedure dùng để đổi mật khẩu tài khoản khách hàng dựa trên hai thông tin đầu vào là Id tài khoản và mật khẩu mới.

```
DELIMITER //
CREATE PROCEDURE changePassword(
    IN customerAccountID VARCHAR(10),
    IN newPassword VARCHAR(50)
)
BEGIN
    -- Cập nhật mật khẩu mới cho tài khoản
    UPDATE ACCOUNT
    SET PASSWORD = newPassword
    WHERE ACCOUNT_ID = customerAccountID;
    -- Thông báo thành công
    SELECT 'Password changed successfully.' AS message;
END //
DELIMITER ;
```

Để đổi mật khẩu của tài khoản có id là A003, ta thực thi như sau:

```
CALL changePassword('A003', 'new_secure_password');
select *from ACCOUNT;
```

Kết quả đạt được:

ACCOUNT_ID	USERNAME	PASSWORD
A001	user1	pass1
A002	user2	pass2
A003	user3	new_secure_password
A004	user4	pass4
A005	user5	pass5

D. QUẢN LÝ

TRUY VẤN THÔNG TIN CÁ NHÂN

Đây là một câu lệnh truy vấn các thông tin cá nhân của quản lý bao gồm: mã quản lý, thời gian bắt đầu làm việc, số định danh, tên, ngày sinh, mail, giới tính, địa chỉ, số điện thoại và công việc.

```
SELECT
    m.MANAGER_ID,
    m.START_WORKING_DAY,
    p.SSN,
    CONCAT(p.FIRSTNAME, ' ', p.MIDNAME, ' ', p.LASTNAME) AS `Full Name`,
    p.DOB,
    p.EMAIL,
    p.SEX,
    p.ADDRESS,
    ph.PHONE_NUM AS `Phone Number`,
    CONCAT("start manage ", pl.PARKINGLOT_NAME, ' from ',  m.MNG_START_TIME) AS `Work`
FROM MANAGER_ACCOUNT ma
JOIN MANAGER m ON ma.MNG_SSN = m.MNG_SSN
JOIN PERSON p ON m.MNG_SSN = p.SSN
LEFT JOIN PHONE ph ON p.SSN = ph.OWNER_SSN
LEFT JOIN PARKING_LOT pl ON m.MNG_SSN = pl.MNG_SSN
WHERE ma.MNG_ACC_ID = 'A003';
```

Như trên ta thực hiện truy vấn thông tin cá nhân của tài khoản có id là A003.

Kết quả như sau:

MANAGER_ID	START_WORKING_DAY	SSN	Full Name	DOB	EMAIL	SEX	ADDRESS	Phone Number	Work
M003	2020-03-01	333333333333	Alice C Johnson	1985-03-03	alice.johnson@example.com	F	789 Oak St	3456789012	start manage Lot C from 2020-03-01 08:00:00

TRUY VẤN DANH SÁCH PHẢN HỒI

Đây là một câu lệnh truy vấn danh sách tất cả các phản hồi của khách hàng. Gồm các thông tin như: mã phản hồi, trạng thái, tiêu đề, loại, nội dung, thời gian, … tên người gửi và mã tài khoản gửi

```
SELECT
    r.*,
    CONCAT(p.FIRSTNAME, ' ', p.MIDNAME, ' ', p.LASTNAME) AS `Sender Name`,
    ca.CUS_ACC_ID AS `Customer Account ID`
FROM RESPONSE r
JOIN CUSTOMER c ON r.CU_SSN = c.CUS_SSN
JOIN PERSON p ON c.CUS_SSN = p.SSN
JOIN CUSTOMER_ACCOUNT ca ON c.CUS_SSN = ca.CUS_SSN;
```

Kết quả như sau:

RESPONSE_ID	STATUS_RE	TYPE_RE	TITLE_RE	MESSAGE	MNG_SSN	CU_SSN	TIME_RE	RATING	CAUSE	EVIDENCE	Sender Name	Customer Account ID
R001	Resolved	Complaint	Issue A	Message A	111111111111	666666666666	2023-01-01 10:00:00	NULL	NULL	Photo evidence A	David F Miller	A006
R002	Pending	Feedback	Issue B	Message B	NULL	777777777777	2023-02-01 11:00:00	2	NULL	NULL	Eve G Wilson	A007
R003	In Progress	Complaint	Issue C	Message C	333333333333	888888888888	2023-03-01 12:00:00	NULL	NULL	Photo evidence E	Frank H Moore	A008
R004	Resolved	Issue	Issue D	Message D	444444444444	999999999999	2023-04-01 13:00:00	NULL	Parking issue A	NULL	Grace I Taylor	A009
R005	Resolved	Feedback	Issue E	Message E	555555555555	000000000000	2023-05-01 14:00:00	5	NULL	NULL	Hank J Anderson	A010

E. VÉ

THÊM VÉ MỚI

Khi khách hàng gửi xe vào một bãi giữ xe, trạng thái sẽ là ‘Active’ chỉ ra rằng vé đang có hiệu lực, chỗ trống sẽ được hệ thống tự chọn dựa trên danh sách chỗ sau đó tiến hành tạo vé cùng với mã code và id.

Bên dưới là một Store Procedure để tạo vé gửi xe

```
DELIMITER //
CREATE PROCEDURE CreateParkingTicket1( IN licensePlate VARCHAR(20), IN parkingLotID VARCHAR(10))
BEGIN
    DECLARE ticketID VARCHAR(10);
    DECLARE tickerCode VARCHAR(20);
    DECLARE ticketStatus VARCHAR(50);
    DECLARE emptySlot INT;

    SET ticketID = CONCAT('T', LPAD(FLOOR(RAND() * 1000), 3, '0'));
    SET tickerCode = CONCAT('CODE', LPAD(FLOOR(RAND() * 10000), 4, '0'));
    SET ticketStatus = 'Active';
    -- Tìm vị trí trống trong bãi xe
    SELECT POSITION_NUMBER INTO emptySlot
    FROM EMPTY_SLOT
    WHERE PKL_ID = parkingLotID AND ISEMPTY = true
    LIMIT 1;
    -- Nếu không có vị trí trống, thông báo lỗi
    IF emptySlot IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Không có vị trí trống trong bãi xe.';
    END IF;
    -- Chèn phiếu gửi xe vào bảng TICKET
    INSERT INTO TICKET (TICKET_ID, TICKER_CODE, TICKET_STATUS, LICENSE_PLATE, PARKINGLOT_ID, EMPTY_NUM, CHECKIN_TIME)
    VALUES (ticketID, tickerCode, ticketStatus, licensePlate, parkingLotID, emptySlot, NOW());
END //
DELIMITER ;
```

Để tiến hành thêm vé gửi xe ví dụ như xe LP002 tại bãi xe P002, ta thực thi:

```
CALL CreateParkingTicket1('LP002', 'P002');
```

Chú ý các giá trị trên phải tồn tại trong bảng tương ứng

Kết quả như sau:

TICKET_ID	TICKER_CODE	TICKET_STATUS	LICENSE_PLATE	PARKINGLOT_ID	EMPTY_NUM	CHECKOUT_TIME	CHECKIN_TIME
T003	CODE3	Expired	LP003	P004	4	2024-11-29 22:13:00	2024-11-29 19:21:00
T004	CODE4	Expired	LP004	P002	2	2023-06-04 18:00:00	2023-06-04 08:00:00
T005	CODE5	Expired	LP006	P005	6	2023-06-05 18:00:00	2023-06-05 08:00:00
T006	CODE6	Expired	LP005	P005	5	2023-06-05 12:31:00	2023-06-05 08:56:00
T007	CODE7	Expired	LP005	P005	22	2024-04-22 13:43:00	2024-04-22 10:08:00
T008	CODE8	Active	LP004	P005	5	NULL	2024-12-10 18:00:00
T477	CODE3941	Active	LP002	P002	14	NULL	2024-12-11 07:52:41
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Một dòng mới xuất hiện ở cuối bảng chứng tỏ vé được tạo thành công.

F. THANH TOÁN

TÍNH TỔNG TIỀN

Đây là một Store Procedure để tính toán tổng số tiền gửi xe phải trả theo vé. Nó nhận vào mã vé và trả kết quả là số tiền. Đầu tiên nó lấy các trường CHECKIN_TIME và CHECKOUT_TIME từ bảng TICKET rồi thực hiện tính toán theo công thức:

- Từ 6h - 18h: 4000 đồng/xe/4 giờ/lượt
- Từ 18h - 6h sáng hôm sau: 6000 đồng/xe/4 giờ/lượt.
- Quá thời gian 4 giờ sẽ thu theo các lượt 4 giờ tiếp theo.

```
DELIMITER //
CREATE PROCEDURE CalculateParkingFee(IN ticketID VARCHAR(10), OUT totalFee INT)
BEGIN
    DECLARE checkin DATETIME;
    DECLARE checkout DATETIME;
    DECLARE totalHours INT;
    DECLARE totalMinutes INT;
    DECLARE dayHours INT;
    DECLARE nightHours INT;
    DECLARE dayFeePer4Hours INT DEFAULT 4000;
    DECLARE nightFeePer4Hours INT DEFAULT 6000;

    -- Lấy thời gian check-in và check-out từ bảng TICKET
    SELECT CHECKIN_TIME, CHECKOUT_TIME INTO checkin, checkout
    FROM TICKET
    WHERE TICKET_ID = ticketID;

    -- Tính tổng số phút
    SET totalMinutes = TIMESTAMPDIFF(MINUTE, checkin, checkout);

    -- Tính tổng số giờ, làm tròn lên nếu có phút lẻ
    SET totalHours = CEIL(totalMinutes / 60);

    -- Khởi tạo số giờ ban ngày và ban đêm
    SET dayHours = 0;
    SET nightHours = 0;
```

```
    -- Lặp qua từng giờ để phân loại giờ ban ngày và ban đêm
    WHILE totalHours > 0 DO
        IF HOUR(checkin) >= 6 AND HOUR(checkin) < 18 THEN
            SET dayHours = dayHours + 1;
        ELSE
            SET nightHours = nightHours + 1;
        END IF;
        SET checkin = ADDTIME(checkin, '01:00:00');
        SET totalHours = totalHours - 1;
    END WHILE;

    -- Tính phí ban ngày
    SET totalFee = (FLOOR(dayHours / 4) * dayFeePer4Hours);
    IF (dayHours % 4) > 0 THEN
        SET totalFee = totalFee + dayFeePer4Hours;
    END IF;

    -- Tính phí ban đêm
    SET totalFee = totalFee + (FLOOR(nightHours / 4) * nightFeePer4Hours);
    IF (nightHours % 4) > 0 THEN
        SET totalFee = totalFee + nightFeePer4Hours;
    END IF;
END //
DELIMITER ;
```

Để tính toán tổng phí gửi xe ví dụ như của vé có mã là T003, ta thực thi như sau:

```
CALL CalculateParkingFee("T003", @totalAmount);
SELECT @totalAmount AS TotalAmount;
```

Kết quả như sau:

	TotalAmount
▶	6000

CHECK-OUT

Đây là một Store Procedure dùng để thực hiện hoạt động check-out (khi quét vé ra về). Khi người dùng quét vé, vé sẽ được cập nhật lại là hết hiệu lực cùng thời gian check-out. Sau đó, hệ thống kiểm tra xem đây là xe gửi tháng hay lượt, hệ thống sẽ tính toán phí gửi xe và tạo thanh toán. Thanh toán lúc này là "Unpaid" chưa được trả và cần khách hàng chọn ngân hàng thanh toán. Tự động, hệ thống sẽ tạo và thêm thông báo thanh toán sau khi PAYMENT được thêm thành công.

```
DELIMITER //
CREATE PROCEDURE checkout(IN ticket VARCHAR(10))
BEGIN
    DECLARE totalAmount INT;
    DECLARE notifiID VARCHAR(10);
    DECLARE paymentID VARCHAR(10);
    DECLARE due_time VARCHAR(5);
    DECLARE exit handler FOR SQLEXCEPTION
    BEGIN
        -- Xử lý lỗi
        ROLLBACK;
        SELECT 'An error occurred. Transaction rolled back.';
    END;

    START TRANSACTION;
    SET notifiID = CONCAT('N', LPAD(FLOOR(RAND() * 1000), 3, '0'));
    SET paymentID = CONCAT('PM', LPAD(FLOOR(RAND() * 1000), 3, '0'));
    -- Cập nhật checkout time
    UPDATE TICKET
    SET CHECKOUT_TIME = NOW(), TICKET_STATUS = 'Expired'
    WHERE TICKET_ID = ticket;

    -- Kiểm tra nếu MONTHLY_PARK không null
    IF (SELECT MONTHLY_PART FROM MOTORBIKE m JOIN TICKET t ON m.LICENSE_PLATE = t.LICENSE_PLATE WHERE t.TICKET_ID = ticket) IS NULL THEN
        -- Tính phí gửi xe
        CALL CalculateParkingFee(ticket, @totalAmount);
        SET due_time = '10h';
    ELSE
        SET totalAmount=300000;
        SET due_time = '24h';
    END IF;
        -- Tạo payment
    INSERT INTO PAYMENT (TRANSACTION_ID, TIC_ID, PAYMENT_METHOD, PAYMENT_TIME, TOTAL_AMOUNT, PAY_STATUS)
    VALUES (paymentID, ticket, "Internet Banking", NOW(), @totalAmount, 'Unpaid');
    -- Tạo thông báo trong bảng NOTIFICATION
    INSERT INTO NOTIFICATION (NOTI_ID, NOTI_TITLE, NOTI_MESSAGE, NOTI_TIME, NOTI_TYPE, NOTI_STATUS, CUS_SSN) VALUES
    (notifiID, 'Payment Notification', CONCAT('Payment of ', totalAmount, ' is due.'), NOW(), 'Payment', 'Unpaid', (
    SELECT m.CUS_SSN FROM TICKET t JOIN MOTORBIKE m ON t.LICENSE_PLATE = m.LICENSE_PLATE WHERE t.TICKET_ID = ticket));
    -- Tạo thông báo thanh toán
    INSERT INTO PAYMENT_NOTIFICATION (NOTIFICATION_ID, TRANSACTION_STATUS, PAYMENT_DUE_TIME, TRANSACTION_ID) VALUES
    (notifiID, 'Unpaid', due_time, paymentID);

    COMMIT;

    SELECT 'Transaction completed successfully.';
END //
DELIMITER ;
```

Để tiến hành check-out ví dụ như với vé có mã là T008, ta thực thi như sau:

```
CALL checkout('T008');
SELECT* FROM TICKET;
SELECT * FROM PAYMENT;
SELECT * FROM PAYMENT_NOTIFICATION;
```

Kết quả như sau:

TICKET_ID	TICKER_CODE	TICKET_STATUS	LICENSE_PLATE	PARKINGLOT_ID	EMPTY_NUM	CHECKOUT_TIME	CHECKIN_TIME
T002	CODE2	Expired	LP002	P003	3	2023-06-02 18:00:00	2023-06-02 08:00:00
T003	CODE3	Expired	LP003	P004	4	2024-11-29 22:13:00	2024-11-29 19:21:00
T004	CODE4	Expired	LP004	P002	2	2023-06-04 18:00:00	2023-06-04 08:00:00
T005	CODE5	Expired	LP006	P005	6	2023-06-05 18:00:00	2023-06-05 08:00:00
T006	CODE6	Expired	LP005	P005	5	2023-06-05 12:31:00	2023-06-05 08:56:00
T007	CODE7	Expired	LP005	P005	22	2024-04-22 13:43:00	2024-04-22 10:08:00
T008	CODE8	Expired	LP004	P005	5	2024-12-11 09:54:05	2024-12-10 18:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ở bảng TICKET thông tin vé T008 được cập nhật thành công. Còn với bảng PAYMENT, một thanh toán mới được thêm vào với mã là PM620.

TRANSACTION_ID	TIC_ID	PAYMENT_METHOD	PAYMENT_TIME	TOTAL_AMOUNT	BANK_NUM	PAY_STATUS
PM002	T002	Cash	2023-07-02 00:00:00	4000.00	BA002	Paid
PM003	T003	Debit Card	2024-11-29 00:00:00	8000.00	BA003	Paid
PM004	T004	Credit Card	2023-07-04 00:00:00	12000.00	BA004	Paid
PM005	T005	Cash	2023-07-05 00:00:00	10000.00	BA005	Paid
PM006	T006	Debit Card	2023-07-03 00:00:00	12000.00	BA006	Paid
PM007	T007	Debit Card	2024-04-22 00:00:00	18000.00	BA006	Paid
PM620	T008	Internet Banking	2024-12-11 09:54:05	22000.00	NULL	Unpaid
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Kiểm tra bảng thông báo thanh toán ta cũng thấy một thông báo mới đã được thêm vào.

NOTIFICATION_ID	TRANSACTION_STATUS	PAYMENT_DUE_TIME	TRANSACTION_ID
N001	Paid	24h	PM001
N002	Paid	20m	PM002
N003	Paid	20m	PM003
N004	NULL	20m	PM004
N005	Paid	20m	PM005
N707	Unpaid	10h	PM620
NULL	NULL	NULL	NULL

G. THANH TOÁN

Đây là một Store Procedure dùng để thực hiện hoạt động thanh toán, nhận vào giá trị đầu vào là tài khoản ngân hàng và mã thanh toán. Khi người dùng thanh toán, hệ thống tiến hành trừ tiền tài khoản ngân hàng, sau đó thông tin sẽ được cập nhật lại là đã trả ‘Paid’ và số tài khoản thanh toán. Đồng thời thông báo thanh toán cũng được cập nhật.

```
DELIMITER //
CREATE PROCEDURE pay_for_parking(payment VARCHAR(10), bank VARCHAR(16))
BEGIN
    DECLARE totalAmount INT;
    DECLARE due_time VARCHAR(5);
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Xử lý lỗi
        ROLLBACK;
        SELECT 'An error occurred. Transaction rolled back.';
    END;
    START TRANSACTION;
    SELECT TOTAL_AMOUNT INTO totalAmount
    FROM PAYMENT
    WHERE TRANSACTION_ID = payment;

    -- Trừ tiền từ tài khoản nguồn
    UPDATE BANK_ACCOUNT
    SET BALANCE = CASE
            WHEN BALANCE - totalAmount < 0 THEN 0
            ELSE BALANCE - totalAmount
        END
    WHERE BANK_ACC_NUMBER = bank;


    -- Ghi lại giao dịch
    UPDATE PAYMENT
    SET BANK_NUM = bank, PAY_STATUS = 'Paid'
    WHERE TRANSACTION_ID = payment;

    -- Cập nhật thông báo thanh toán
    UPDATE PAYMENT_NOTIFICATION
    SET TRANSACTION_STATUS = 'Paid'
    WHERE TRANSACTION_ID = payment;
    COMMIT;

    SELECT 'Transaction completed successfully.';
END //
DELIMITER ;
```

Để thực hiện thanh toán ví dụ như cho khoản thanh toán số PM620 bằng tài khoản ngân hàng BA004, ta thực thi như sau:

```
CALL pay_for_parking('PM620','BA004');
SELECT * FROM PAYMENT;
SELECT * FROM PAYMENT_NOTIFICATION;
```

Kết quả như sau:

Ta thấy thông tin của thanh toán có mã là PM620 đã được cập nhật thành công.

TRANSACTION_ID	TIC_ID	PAYMENT_METHOD	PAYMENT_TIME	TOTAL_AMOUNT	BANK_NUM	PAY_STATUS
PM002	T002	Cash	2023-07-02 00:00:00	4000.00	BA002	Paid
PM003	T003	Debit Card	2024-11-29 00:00:00	8000.00	BA003	Paid
PM004	T004	Credit Card	2023-07-04 00:00:00	12000.00	BA004	Paid
PM005	T005	Cash	2023-07-05 00:00:00	10000.00	BA005	Paid
PM006	T006	Debit Card	2023-07-03 00:00:00	12000.00	BA006	Paid
PM007	T007	Debit Card	2024-04-22 00:00:00	18000.00	BA006	Paid
PM620	T008	Internet Banking	2024-12-11 09:54:05	22000.00	BA004	Paid
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Đồng thời thông báo thanh toán cũng được cập nhật.

NOTIFICATION_ID	TRANSACTION_STATUS	PAYMENT_DUE_TIME	TRANSACTION_ID
N001	Paid	24h	PM001
N002	Paid	20m	PM002
N003	Paid	20m	PM003
N004	NULL	20m	PM004
N005	Paid	20m	PM005
N707	Paid	10h	PM620

H. BÃI XE

SỐ CHỖ TRỐNG

Đây là một truy vấn để xác định số chỗ trống, số chỗ đã dùng của từng bãi xe.

```
SELECT
    es.PKL_ID AS `Parking Lot ID`,
    pl.PARKINGLOT_NAME AS `Parking Lot Name`,
    pl.CAPACITY AS `Capacity`,
    SUM(CASE WHEN es.ISEMPTY = FALSE THEN 1 ELSE 0 END) AS `Used Slots`,
    pl.CAPACITY - SUM(CASE WHEN es.ISEMPTY = FALSE THEN 1 ELSE 0 END) AS `Empty Slots`
FROM EMPTY_SLOT es
JOIN PARKING_LOT pl ON es.PKL_ID = pl.PARKINGLOT_ID
GROUP BY es.PKL_ID, pl.PARKINGLOT_NAME, pl.CAPACITY;
```

Kết quả sẽ cho ra:

	Parking Lot ID	Parking Lot Name	Capacity	Used Slots	Empty Slots
▶	P001	Lot A	100	1	99
	P002	Lot B	150	1	149
	P003	Lot C	200	1	199
	P004	Lot D	250	1	249
	P005	Lot E	300	3	297

CẬP NHẬT THÔNG TIN BÃI XE

Để cập nhật thông tin bãi giữ xe, ta cần phải thực hiện việc cập nhật thông qua khóa chính của bãi giữ xe, ở đây là PARKINGLOT_ID.

```
DELIMITER //
CREATE PROCEDURE UpdateParkingLotInfo(
    IN p_parkinglot_id VARCHAR(10),
    IN p_new_name VARCHAR(100),
    IN p_new_capacity INT,
    IN p_new_address VARCHAR(255)
)
BEGIN
    -- Kiểm tra nếu tham số p_parkinglot_id có giá trị NULL
    IF p_parkinglot_id IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ParkingLot ID cannot be NULL!';
    END IF;

    -- Cập nhật thông tin bãi giữ xe
    UPDATE PARKING_LOT
    SET
        PARKINGLOT_NAME = p_new_name,
        CAPACITY = p_new_capacity,
        ADDRESS = p_new_address
    WHERE PARKINGLOT_ID = p_parkinglot_id;

    -- Nếu không có dòng nào bị thay đổi, đưa ra cảnh báo
    IF ROW_COUNT() = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No matching ParkingLot ID found!';
    END IF;
END //
DELIMITER ;
```

Nếu muốn cập nhật thông tin một bãi giữ xe nào đó, ta chỉ cần thực thi lệnh:

```
12 •   CALL UpdateParkingLotInfo('P001', 'New Parking Lot A', 120, 'New Address 123');
```

Kết quả sẽ thay đổi sau khi thực hiện truy vấn, ở đây bãi giữ xe có mã là P001:

PARKINGLOT_ID	CAPACITY	ADDRESS	PARKINGLOT_NAME	MNG_SSN
P001	120	New Address 123	New Parking Lot A	111111111111
NULL	NULL	NULL	NULL	NULL

I. XE MÁY

Xe máy được thêm bởi một stored procedure AddMotorbike như sau:

```
95    DELIMITER //
96
97 •  CREATE PROCEDURE AddMotorbike(
98        IN p_license_plate VARCHAR(20),
99        IN p_color VARCHAR(20),
100       IN p_brand VARCHAR(50),
101       IN p_type VARCHAR(50),
102       IN p_cus_ssn VARCHAR(12)
103   )
104   BEGIN
105       -- Kiểm tra nếu khách hàng tồn tại
106       IF NOT EXISTS (SELECT 1 FROM CUSTOMER WHERE CUS_SSN = p_cus_ssn) THEN
107           SIGNAL SQLSTATE '45000'
108           SET MESSAGE_TEXT = 'Customer does not exist!';
109       END IF;
110
111       -- Kiểm tra nếu biển số xe đã tồn tại
112       IF EXISTS (SELECT 1 FROM MOTORBIKE WHERE LICENSE_PLATE = p_license_plate) THEN
113           SIGNAL SQLSTATE '45000'
114           SET MESSAGE_TEXT = 'License plate already exists!';
115       END IF;
116
117       -- Thêm xe máy vào bảng MOTORBIKE
118       INSERT INTO MOTORBIKE (LICENSE_PLATE, COLOR, BRAND, TYPE, CUS_SSN)
119       VALUES (p_license_plate, p_color, p_brand, p_type, p_cus_ssn);
120   END //
121
122   DELIMITER ;
```

Sau khi đã thêm thủ tục, khi muốn thêm một thông tin xe máy ta chỉ cần thực hiện lệnh gọi, ví dụ như sau:

```
CALL AddMotorbike('LP006', 'Black', 'Harley-Davidson', 'Cruiser', '666666666666');
```

Để kiểm tra xem xe đã được thêm vào hay chưa, ta sẽ kiểm tra thông qua khóa chính LICENSE_PLATE, ở đây có mã là ‘LP006’:

	LICENSE_PLATE	COLOR	BRAND	TYPE	CUS_SSN
▶	LP006	Black	Harley-Davidson	Cruiser	666666666666
*	NULL	NULL	NULL	NULL	NULL