Parking Lot Management System
Database Implementation

아주대학교 디지털미디어학과 박성범 (201721107)

1. Relational Data Model with Samples

VEHICLE

VNumber	VType	Last_enter_at	Owner	Email	Phone
24조2426	NORMAL	NULL	Kim	kim@mail.com	821038277712
382다7499	NORMAL	2023-04-07T15:58:32	Lee	NULL	NULL
86135	LOW_EMISSION	2023-04-07T11:04:16	Kim	kim@mail.com	821038277712
경기51거2824	LIGHT	NULL	Park	NULL	16175551283
89하6063	NORMAL	NULL	NULL	NULL	NULL

SLOT

SID	Floor	Section	SType	Occupied_by
1A01	1	A	NORMAL	NULL
2A01	2	A	NORMAL	NULL
2A02	2	A	IMPAIRMENT	382다7499
2B01	2	B	LIGHT	NULL
3B01	2	B	ELECTRIC	86135

TICKET

TID	Issued_at	Expired_at	Renewed_at	Registered_on	Owner	Discount_types
12eec58f-0b2a-4f4b-95c9-5fba021b4886	2020-12-11T10:43:36	2023-07-11T18:00:00	2022-06-11T17:12:42	382다7499	Lee	MERIT
1e627944-4af0-4251-bceb-440a53a86fc0	2023-04-01T14:01:53	2023-05-01T15:00:00	NULL	경기51거2824	Park	NONE
164fc1f6-d27c-4f99-a10b-3212093dcd62	2019-12-11T09:42:46	2020-01-11T10:00:00	NULL	경기51거2824	Park	MERIT, WOUND

ENTER_LOG

ENID	Vehicle	Enter_to	Enter_at
ddc78b3b-bb90-4eaf-b17d-9a931228a21d	382다7499	WEST01	2023-04-07T15:58:32
490919e7-0ccd-489a-b45a-029489e2b996	86135	EAST01	2023-04-07T11:04:16
8f8d4d40-e779-4af8-965a-0f95790bc9cd	24조2426	WEST01	2023-04-06T18:22:17
1a658e16-3c24-4352-837b-d211c20c7ec5	89하6063	EAST01	2023-04-06T17:58:43
d8a0c824-caa0-4a72-a011-8fb41f9003fa	경기51거2824	WEST02	2023-04-05T12:34:10
6ba2a247-5cb7-4159-a5d5-ac9051fc3000	86135	NORTH01	2023-04-05T12:32:56

EXIT_LOG

Enter	Exit_from	Exit_at
490919e7-0ccd-489a-b45a-029489e2b996	WEST02	2023-04-07T17:47:07
d8a0c824-caa0-4a72-a011-8fb41f9003fa	NORTH01	2023-04-05T15:08:11
6ba2a247-5cb7-4159-a5d5-ac9051fc3000	EAST01	2023-04-05T13:43:51

PAID_FOR_SETTLEMENT_LOG

PSID	Vehicle	Method	Amount	PSPaid_at
14782282-be2e-4765-a874-e8a7994c14cc	24조2426	CARD	3600	2023-04-05T15:00:08
ac3383e9-1825-442b-9caf-b13c93fbfaed	89하6063	CARD	12300	2023-04-05T10:35:19

PAID_FOR_TICKET_LOG

PTID	Ticket	PTType	Method	Amount	PTPaid_at
34d29749-9178-4848-977c-df0be33b9ba2	1e627944-4af0-4251-bceb-440a53a86fc0	NEW	CASH	35000	2023-04-01T14:01:53
22700c6d-0b65-4f72-b1ea-65bab0cd91b8	12eec58f-0b2a-4f4b-95c9-5fba021b4886	RENEW	CASH	20000	2022-06-11T17:12:42
7365483b-aef8-4621-a3c5-d8e952d0591d	164fc1f6-d27c-4f99-a10b-3212093dcd62	NEW	CARD	35000	2019-12-11T09:42:46

2. Functional Dependency and Normalization

2.1. 1NF

TICKET에서 ticket_discount_types(요금 감면 유형)은 쉼표로 구분된 multi-valued 어트리뷰트로서 1NF를 위반하므로, 아래와 같이 테이블을 분리해야 한다.

TICKET

TID	Issued_at	Expired_at	Renewed_at	Registred_on	Owner

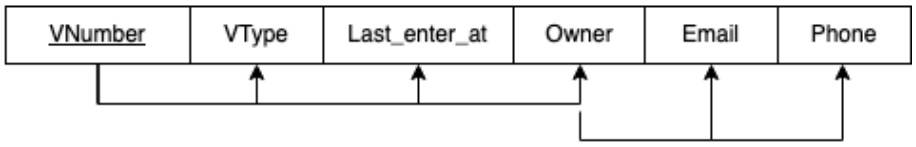
TICKET_DISCOUNT_TYPES

Ticket	TType

2.2. 2NF

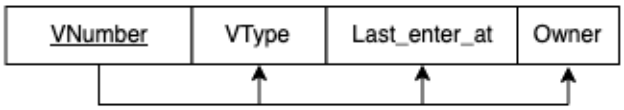
VEHICLE에서 Owner(차주)가 VNumber(차량번호)에 종속적인 동시에, Email과 Phone이 Owner에 종속적이므로 2NF를 위반한다.

VEHICLE

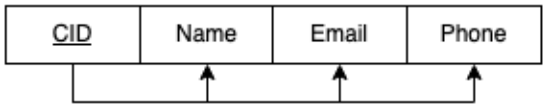


따라서 아래와 같이 테이블을 분리해야 2NF 조건을 만족할 수 있다. 이때 VEHICLE의 Owner는 CUSTOMER의 CID를 참조하는 FKey이다.

VEHICLE

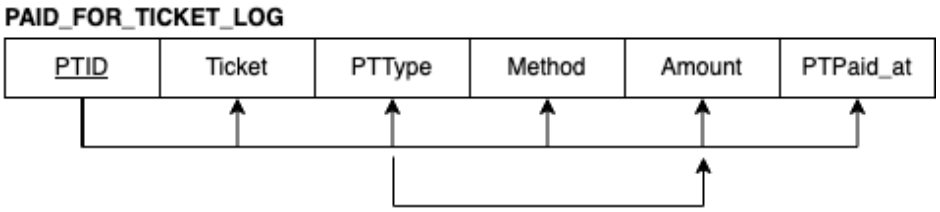


CUSTOMER

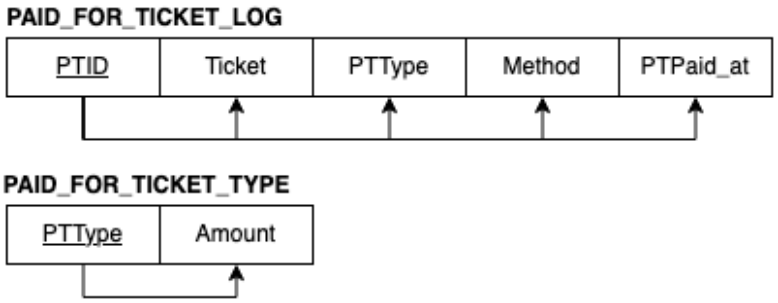


2.3. 3NF

PAID_FOR_TICKET_LOG에서 Amount(결제금액)은 PTType(결제유형)에 종속적이며, PTType은 PTID(로그 아이디)에 종속적이다. 따라서 PTID와 Amount 사이에 transitive FD가 존재한다.

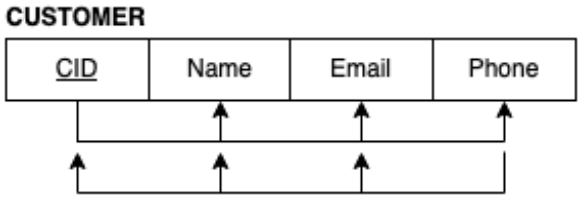


따라서 아래와 같이 테이블을 분리해야 3NF 조건을 만족할 수 있다.

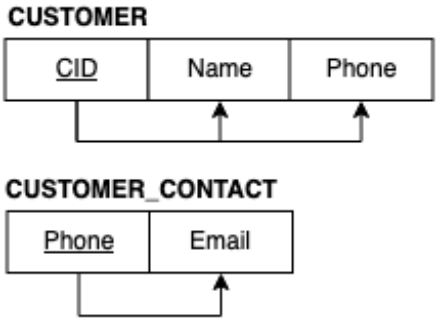


2.4. BCNF

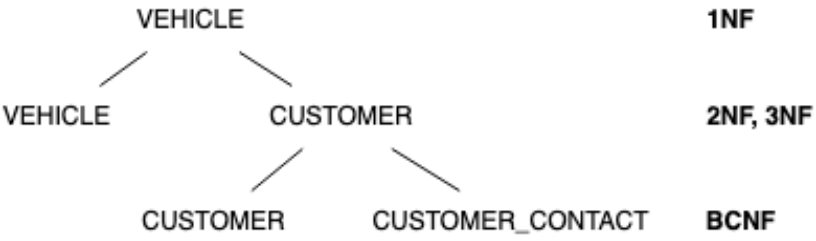
CUSTOMER에는 CID뿐 아니라 Phone(전화번호)도 결정자 역할을 하므로 BCNF를 위반한다.



따라서 아래와 같이 테이블을 분리해야 BCNF 조건을 만족할 수 있다. 이때 CUSTOMER의 Phone은 CUSTOMER_CONTACT의 Phone을 참조하는 FKey이다.



결과적으로 VEHICLE에는 아래와 같은 normalization 과정이 이뤄졌다.



3. Implementation

3.1. Create

VEHICLE

```
CREATE TABLE vehicles (
    vnumber VARCHAR(10) NOT NULL,
    vtype VARCHAR(16) NOT NULL DEFAULT "NORMAL",
    last_enter_at TIMESTAMP DEFAULT NOW(),
    owner INT,
    PRIMARY KEY (vnumber)
);
```

SLOT

```
CREATE TABLE slots (
    sid VARCHAR(16) NOT NULL,
    floor INT,
    section VARCHAR(16),
    stype VARCHAR(16) NOT NULL DEFAULT "NORMAL",
    occupied_by VARCHAR(10),
    PRIMARY KEY (sid),
    UNIQUE (occupied_by),
    FOREIGN KEY (occupied_by) REFERENCES vehicles(vnumber)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

CUSTOMER_CONTACT

```
CREATE TABLE customer_contacts (
    phone VARCHAR(16) NOT NULL,
    email VARCHAR(255),
    PRIMARY KEY (phone)
);
```

CUSTOMER

```
CREATE TABLE customers (
    cid INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(10),
    phone VARCHAR(16),
    PRIMARY KEY (cid),
    FOREIGN KEY (phone) REFERENCES customer_contacts(phone)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

TICKET

```
CREATE TABLE tickets (
    tid VARCHAR(36) NOT NULL DEFAULT UUID(),
    issued_at TIMESTAMP NOT NULL DEFAULT NOW(),
    expired_at TIMESTAMP NOT NULL DEFAULT (NOW() + INTERVAL '1' MONTH),
    renewed_at TIMESTAMP,
    discount_types VARCHAR(16) NOT NULL DEFAULT "NONE",
    registered_on VARCHAR(10) NOT NULL,
    owner INT NOT NULL,
    PRIMARY KEY (tid),
    FOREIGN KEY (registered_on) REFERENCES vehicles(vnumber)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (owner) REFERENCES customers(cid)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

TICKET_DISCOUNT_TYPE

```
CREATE TABLE ticket_discount_types (
    ticket VARCHAR(36) NOT NULL DEFAULT UUID(),
    ttype VARCHAR(16) NOT NULL,
    PRIMARY KEY (ticket, ttype),
    FOREIGN KEY (ticket) REFERENCES tickets(tid)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

ENTER_LOG

```
CREATE TABLE enter_logs (
    enid VARCHAR(36) NOT NULL DEFAULT UUID(),
    vehicle VARCHAR(10) NOT NULL,
    enter_to VARCHAR(16),
    enter_at TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (enid),
    FOREIGN KEY (vehicle) REFERENCES vehicles(vnumber)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

EXIT_LOG

```
CREATE TABLE exit_logs (
    enter VARCHAR(36) NOT NULL,
    exit_from VARCHAR(16),
    exit_at TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (enter),
    FOREIGN KEY (enter) REFERENCES enter_logs(enid)
        ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

PAID_FOR_SETTLEMENT_LOG

```
CREATE TABLE paid_for_settlement_logs (
    psid VARCHAR(36) NOT NULL DEFAULT UUID(),
    vehicle VARCHAR(10) NOT NULL,
    method VARCHAR(16) NOT NULL,
    amount INT NOT NULL,
    pspaid_at TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (psid),
    FOREIGN KEY (vehicle) REFERENCES vehicles(vnumber)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

PAID_FOR_TICKET_TYPE

```
CREATE TABLE paid_for_ticket_types (
    pttype VARCHAR(16) NOT NULL,
    amount INT NOT NULL,
    PRIMARY KEY (pttype, amount)
);
```

PAID_FOR_TICKET_LOG

```
CREATE TABLE paid_for_ticket_logs (
    ptid VARCHAR(36) NOT NULL DEFAULT UUID(),
    ticket VARCHAR(36) NOT NULL,
    pttype VARCHAR(16) NOT NULL,
    method VARCHAR(16) NOT NULL,
    ptpaid_at TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (ptid),
    FOREIGN KEY (pttype) REFERENCES paid_for_ticket_types(pttype)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

VEHICLE_OWNER (FKey)

```
ALTER TABLE vehicles ADD FOREIGN KEY (owner) REFERENCES customers(cid)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

3.2. Constraints

CHECK_VEHICLE_TYPE

```
CREATE TRIGGER check_vehicle_type
BEFORE UPDATE ON slots
FOR EACH ROW
BEGIN
    DECLARE vehicle_type VARCHAR(16);
    SELECT vtype INTO vehicle_type FROM vehicles
        WHERE vnumber = NEW.occupied_by;
    IF NEW.stype = "LIGHT" THEN
        IF vehicle_type <> "LIGHT" THEN
            SIGNAL SQLSTATE "45000";
        END IF;
    ELSEIF NEW.stype = "ELECTRIC" THEN
        IF vehicle_type <> "LOW_EMISSION" THEN
            SIGNAL SQLSTATE "45000";
        END IF;
    END IF;
END;
```

차량은 주차 자리 유형의 조건을 만족해야 해당 자리에 주차할 수 있다. 일반(NORMAL) 자리에는 모든 차량이 주차 가능하다. 또한 차량 유형으로는 장애인(IMPAIRMENT) 자리에 주차 가능한지 판단할 수 없으므로 조건에서 제외한다. 따라서 경형(LIGHT) 자리에 주차한 차량이 경형이 아닌 경우와 전기(ELECTRIC) 자리에 주차한 차량이 저공해(LOW_EMISSION)가 아닌 경우 업데이트를 제한한다.

MONTHLY_QUATER

```
CREATE TRIGGER monthly_quater
BEFORE INSERT ON tickets
FOR EACH ROW
BEGIN
    DECLARE slot_count INT;
    DECLARE issued INT;
    SELECT COUNT(*) INTO slot_count FROM slots;
    SELECT COUNT(*) INTO issued FROM tickets WHERE expired_at > NOW();
    IF (slot_count * 0.25) < issued THEN
        SIGNAL SQLSTATE "45000";
    END IF;
END;
```

만료되지 않은 기발급 정기권이 전체 주차 자리의 25% 이하일 때만 신규 정기권을 발급할 수 있다.

PREVENT_OVERPRICE

```
CREATE TRIGGER prevent_overprice
BEFORE INSERT ON paid_for_settlement_logs
FOR EACH ROW
BEGIN
    DECLARE last_enter_at TIMESTAMP;
    SELECT last_enter_at INTO last_enter_at FROM vehicles
        WHERE vnumber = NEW.vehicle;
    IF TIMESTAMPDIFF(MINUTE, last_enter_at, NOW()) > 1440 THEN
        SET NEW.amount = CEIL(a / 1440) * 20000
            + CEIL(TIMESTAMPDIFF(MINUTE, last_enter_at, NOW())
                % 1440) / 10 * 500;
    ELSE
        IF NEW.amount > 20000 THEN
           SET NEW.amount = 20000;
        END IF;
    END IF;
END;
```

출차 정산 요금은 10분당 500원을 부과하되, 24시간 단위로 최대 20000원을 넘지 못하게 한다. 즉, 24시간 이하로 주차한 경우에는 사용자가 결제한 금액을 그대로 반영하되 최대 20000원이 부과되며, 24시간을 초과한 경우에는 24시간 단위 최대 요금이 적용된다. 다시 말해서, 24시간 주차한 경우에는 20000원, 48시간 주차한 경우에는 40000원을 부과하며, 48시간 10분 주차한 경우에는 40500원을 부과한다.

PREVENT_TO_DELETE_OCCUPIED_SLOT

```
CREATE TRIGGER prevent_to_delete_occupied_slot
BEFORE DELETE ON slots
FOR EACH ROW
BEGIN
    IF OLD.occupied_by IS NOT NULL THEN
        SIGNAL SQLSTATE "45000";
    END IF;
END;
```

자리에 주차된 차량이 있다면 해당 자리를 삭제하지 못한다.

PREVENT_DUPLICATE_TICKET

```
CREATE TRIGGER prevent_duplicate_ticket
BEFORE INSERT ON tickets
FOR EACH ROW
BEGIN
    IF EXISTS (SELECT * FROM tickets
                WHERE owner = NEW.owner
                    AND registered_on = NEW.registered_on
                    AND expired_at > NOW()) THEN
        SIGNAL SQLSTATE "45000";
    END IF;
END;
```

정기권 발급 시 해당 고객, 차량에 대해 아직 만료되지 않은 정기권이 있다면 중복 발급할 수 없다.

3.3. Insert and Delete

정기권 구입

```
INSERT INTO customer_contacts (phone, email)
    VALUES ("82848836209", "yoon@email.com");
INSERT INTO customers (name, phone) VALUES ("Yoon", "82848836209");
INSERT INTO vehicles (owner)
    VALUES (SELECT cid FROM customers WHERE phone = "82848836209");
INSERT INTO tickets (registered_on, owner)
    VALUES ("서울72차8738",
        (SELECT cid FROM customers WHERE phone = "82848836209"));
INSERT INTO paid_for_ticket_logs (ticket, pttype, method)
    VALUES ((SELECT tid FROM tickets WHERE owner =
        (SELECT cid FROM customers WHERE phone = "82848836209")),
        "NEW", "CARD");
```

customers (BEFORE)

	cid	name	phone
1	1	Lee	8277482545
2	2	Bae	NULL
3	3	Park	16175551283
4	10	Kim	821038277712
5	11	Choi	NULL

customer_contacts (BEFORE)

	phone	email
1	16175551283	NULL
2	821038277712	kim@mail.com
3	8277482545	lee@email.com

customers (AFTER)

	cid	name	phone
1	1	Lee	8277482545
2	2	Bae	NULL
3	3	Park	16175551283
4	10	Kim	821038277712
5	11	Choi	NULL
6	12	Yoon	82848836209

customer_contacts (AFTER)

	phone	email
1	16175551283	NULL
2	821038277712	kim@mail.com
3	8277482545	lee@email.com
4	82848836209	yoon@email.com

vehicles (BEFORE)

	vnumber	vtype	last_enter_at	owner
1	12너54843	LIGHT	NULL	NULL
2	24조2426	NORMAL	NULL	3
3	382다7499	NORMAL	NULL	NULL
4	86135	LOW_EMISSION	NULL	2
5	89하6063	NORMAL	NULL	1
6	경기51거2824	NORMAL	NULL	1
7	서울72차8738	NORMAL	NULL	NULL

vehicles (AFTER)

	vnumber	vtype	last_enter_at	owner
1	12너54843	LIGHT	NULL	NULL
2	24조2426	NORMAL	NULL	3
3	382다7499	NORMAL	NULL	NULL
4	86135	LOW_EMISSION	NULL	2
5	89하6063	NORMAL	NULL	1
6	경기51거2824	NORMAL	NULL	1
7	서울72차8738	NORMAL	NULL	12

tickets (BEFORE)

	tid	issued_at	expired_at	renewed_at	discount_types	registered_on	owner
1	15a51b73-1603-4dbb-a002-2adb2337	2022-05-02 13:36:41	2023-06-02 13:32:41	2023-06-02 13:36:11	NONE	경기51거2824	1
2	18148960-3e15-47d0-9ff0-0991265c	2023-06-02 13:36:18	2023-07-02 13:36:18	NULL	NONE	86135	2
3	34539459-742b-42ab-8621-6288f1eb	2023-06-02 13:33:54	2023-07-02 13:33:54	NULL	NONE	89하6063	1

tickets (AFTER)

	tid	issued_at	expired_at	renewed_at	discount_types	registered_on	owner
1	15a51b73-1603-4dbb-a002-2adb2337	2022-05-02 13:36:41	2023-06-02 13:32:41	2023-06-02 13:36:11	NONE	경기51거2824	1
2	18148960-3e15-47d0-9ff0-0991265c	2023-06-02 13:36:18	2023-07-02 13:36:18	NULL	NONE	86135	2
3	34539459-742b-42ab-8621-6288f1eb	2023-06-02 13:33:54	2023-07-02 13:33:54	NULL	NONE	89하6063	1
4	cbbbe942-2304-44d3-bc82-19bb2481	2023-06-03 13:54:15	2023-06-03 23:03:25	NULL	NONE	서울72차8738	3

paid_for_ticket_logs (BEFORE)

	ptid	ticket	pttype	method	ptpaid_at
1	09ca15c7-afa…	15a51b73-160…	NEW	CARD	2022-05-02 13:36:41
2	55dfadd7-b34…	34539459-742…	NEW	CARD	2023-06-02 13:33:54
3	a49f7b3a-832…	18148960-3e1…	NEW	CARD	2023-06-02 13:36:18
4	bbcdb8c6-4be…	15a51b73-160…	RENEW	CASH	2023-06-02 13:36:11

paid_for_ticket_logs (AFTER)

	ptid	ticket	pttype	method	ptpaid_at
1	09ca15c7-af…	15a51b73-16…	NEW	CARD	2022-05-02 13:36:41
2	55dfadd7-b3…	34539459-74…	NEW	CARD	2023-06-02 13:33:54
3	637b46bf-c6…	cbbbe942-23…	NEW	CARD	2023-06-03 13:54:50
4	a49f7b3a-83…	18148960-3e…	NEW	CARD	2023-06-02 13:36:18
5	bbcdb8c6-4b…	15a51b73-16…	RENEW	CASH	2023-06-02 13:36:11

정기권 환불

```
UPDATE tickets SET expired_at = NOW()
    WHERE tid = "cbbbe942-2304-44d3-bc82-19bb2481";
INSERT INTO paid_for_ticket_logs (ticket, pttype, method)
    VALUES ("cbbbe942-2304-44d3-bc82-19bb2481", "WITHDRAW_NEW", "CARD");
```

정기권 레코드를 실제로 제거하는 대신, 즉시 만료시킨다. 해당 정기권에 대응하는 결제 로그 역시 제거하기 보다는, 환불 로그를 추가하는 편이 더 합리적일 것이다. 이때 NEW의 금액(amount)이 35000이라면, WITHDRAW_NEW의 금액은 -35000으로 설정해둔다.

```
SELECT amount FROM paid_for_ticket_types
WHERE pttype = "WITHDRAW_NEW"; -- returns -35000
```

만약 고객이 시스템에서 자신의 고객 정보를 지우길 요구한다면, 해당 고객의 이름을 NULL로 바꾸고, customer_contacts 레코드를 삭제해 익명화한다.

```
UPDATE customers SET name = NULL;
DELETE FROM customer_contacts WHERE phone = OLD.phone
```

tickets (BEFORE)

	tid	issued_at	expired_at	renewed_at	discount_types	registered_on	owner
1	15a51b73-1603-4dbb-…	2022-05-02 13:36:41	2023-06-02 13:32:41	2023-06-02 13:36:11	NONE	경기51거2824	1
2	18148960-3e15-47d0-…	2023-06-02 13:36:18	2023-07-02 13:36:18	NULL	NONE	86135	2
3	34539459-742b-42ab-…	2023-06-02 13:33:54	2023-07-02 13:33:54	NULL	NONE	89하6063	1
4	cbbbe942-2304-44d3-…	2023-06-03 13:54:15	2023-07-03 13:54:15	NULL	NONE	24조2426	3

tickets (AFTER)

	tid	issued_at	expired_at	renewed_at	discount_types	registered_on	owner
1	15a51b73-1603-4dbb-…	2022-05-02 13:36:41	2023-06-02 13:32:41	2023-06-02 13:36:11	NONE	경기51거2824	1
2	18148960-3e15-47d0-…	2023-06-02 13:36:18	2023-07-02 13:36:18	NULL	NONE	86135	2
3	34539459-742b-42ab-…	2023-06-02 13:33:54	2023-07-02 13:33:54	NULL	NONE	89하6063	1
4	cbbbe942-2304-44d3-…	2023-06-03 13:54:15	2023-06-03 23:03:25	NULL	NONE	24조2426	3

paid_for_ticket_logs (BEFORE)

	ptid	ticket	pttype	method	ptpaid_at
1	09ca15c7-af…	15a51b73-16…	NEW	CARD	2022-05-02 13:36:41
2	55dfadd7-b3…	34539459-74…	NEW	CARD	2023-06-02 13:33:54
3	637b46bf-c6…	cbbbe942-23…	NEW	CARD	2023-06-03 13:54:50
4	a49f7b3a-83…	18148960-3e…	NEW	CARD	2023-06-02 13:36:18
5	bbcdb8c6-4b…	15a51b73-16…	RENEW	CASH	2023-06-02 13:36:11

paid_for_ticket_logs (AFTER)

	ptid	ticket	pttype	method	ptpaid_at
1	09ca15c7-af…	15a51b73-16…	NEW	CARD	2022-05-02 13:36:41
2	55dfadd7-b3…	34539459-74…	NEW	CARD	2023-06-02 13:33:54
3	bbcdb8c6-4b…	15a51b73-16…	RENEW	CASH	2023-06-02 13:36:11
4	a49f7b3a-83…	18148960-3e…	NEW	CARD	2023-06-02 13:36:18
5	637b46bf-c6…	cbbbe942-23…	NEW	CARD	2023-06-03 13:54:50
6	62938b46-a1…	cbbbe942-23…	WITHDRAW_NEW	CARD	2023-06-03 23:03:25

주차 자리 추가

```
INSERT INTO slots (sid, floor, section) VALUES ("3A01", 3, "A");
```

slots (BEFORE)

	sid	floor	section	stype	occupied_by
1	1a01	1	a	NORMAL	24조2426
2	1a02	1	a	NORMAL	NULL
3	1a03	1	a	NORMAL	NULL
4	1a04	1	a	NORMAL	NULL
5	1b01	1	b	LIGHT	58너5954
6	1b02	1	b	LIGHT	382다7499
7	1b03	1	b	NORMAL	NULL
8	1b04	1	b	NORMAL	NULL
9	2a01	2	a	NORMAL	NULL
10	2a02	2	a	NORMAL	NULL
11	2a03	2	a	IMPAIRMENT	89하6063
12	2a04	2	a	IMPAIRMENT	NULL
13	2b01	2	b	NORMAL	NULL
14	2b02	2	b	NORMAL	NULL
15	2b03	2	b	ELECTRIC	86135
16	2b04	2	b	ELECTRIC	NULL

slots (AFTER)

	sid	floor	section	stype	occupied_by
1	1a01	1	a	NORMAL	24조2426
2	1a02	1	a	NORMAL	NULL
3	1a03	1	a	NORMAL	NULL
4	1a04	1	a	NORMAL	NULL
5	1b01	1	b	LIGHT	58너5954
6	1b02	1	b	LIGHT	382다7499
7	1b03	1	b	NORMAL	NULL
8	1b04	1	b	NORMAL	NULL
9	2a01	2	a	NORMAL	NULL
10	2a02	2	a	NORMAL	NULL
11	2a03	2	a	IMPAIRMENT	89하6063
12	2a04	2	a	IMPAIRMENT	NULL
13	2b01	2	b	NORMAL	NULL
14	2b02	2	b	NORMAL	NULL
15	2b03	2	b	ELECTRIC	86135
16	2b04	2	b	ELECTRIC	NULL
17	3A01	3	A	NORMAL	NULL

주차 자리 제거

```
DELETE FROM slots WHERE sid = "3A01";
```

slots (BEFORE)

	sid	floor	section	stype	occupied_by
1	1a01	1	a	NORMAL	24조2426
2	1a02	1	a	NORMAL	NULL
3	1a03	1	a	NORMAL	NULL
4	1a04	1	a	NORMAL	NULL
5	1b01	1	b	LIGHT	58너5954
6	1b02	1	b	LIGHT	382다7499
7	1b03	1	b	NORMAL	NULL
8	1b04	1	b	NORMAL	NULL
9	2a01	2	a	NORMAL	NULL
10	2a02	2	a	NORMAL	NULL
11	2a03	2	a	IMPAIRMENT	89하6063
12	2a04	2	a	IMPAIRMENT	NULL
13	2b01	2	b	NORMAL	NULL
14	2b02	2	b	NORMAL	NULL
15	2b03	2	b	ELECTRIC	86135
16	2b04	2	b	ELECTRIC	NULL
17	3A01	3	A	NORMAL	NULL

slots (AFTER)

	sid	floor	section	stype	occupied_by
1	1a01	1	a	NORMAL	24조2426
2	1a02	1	a	NORMAL	NULL
3	1a03	1	a	NORMAL	NULL
4	1a04	1	a	NORMAL	NULL
5	1b01	1	b	LIGHT	58너5954
6	1b02	1	b	LIGHT	382다7499
7	1b03	1	b	NORMAL	NULL
8	1b04	1	b	NORMAL	NULL
9	2a01	2	a	NORMAL	NULL
10	2a02	2	a	NORMAL	NULL
11	2a03	2	a	IMPAIRMENT	89하6063
12	2a04	2	a	IMPAIRMENT	NULL
13	2b01	2	b	NORMAL	NULL
14	2b02	2	b	NORMAL	NULL
15	2b03	2	b	ELECTRIC	86135
16	2b04	2	b	ELECTRIC	NULL

앞서 정의한 제약으로 인해 차량이 주차되어 있는 자리는 제거할 수 없다.

차량 입출차

```
INSERT vehicles (vnumber, vtype) VALUES ("382다7499", "LIGHT");
INSERT INTO enter_logs (vehicle, enter_to)
    VALUES ("382다7499", "WEST02");
```

차량이 입차하면 차량 정보와 입차 로그를 추가한다. 데이터베이스에 이미 차량 정보가 있는 경우에는 아래와 같이 업데이트를 수행한다.

```
UPDATE vehicles SET last_enter_at = NOW()
        WHERE vnumber = "382다7499";
INSERT INTO enter_logs (vehicle, enter_to)
        VALUES ("382다7499", "WEST02");
```

반대로 차량을 출차하는 경우에는 vehicles 테이블에 항상 차량 정보가 있음을 보장할 수 있으므로, 업데이트만을 수행하고 해당 차량의 가장 최근 입차 로그에 대응하는 출차 로그를 추가한다.

```
UPDATE vehicles SET last_enter_at = NULL WHERE vnumber = "382다7499";
INSERT INTO exit_logs (enter, exit_from)
    VALUES ( ( SELECT enid FROM enter_logs
        WHERE vehicle = "382다7499" ORDER BY enter_at DESC LIMIT 1 ),
        "SOUTH01" );
```

3.4. Retrieval

차량이 주차되어 있는 자리 검색

```
SELECT sid, floor, section FROM slots INNER JOIN vehicles
    ON occupied_by = vnumber WHERE vnumber = "24조2426";
```

주어진 차량 번호를 이용해 주차 자리를 찾는다.

주차 가능한 자리 집계

```
SELECT floor, COUNT(*) FROM slots
    WHERE occupied_by IS NULL GROUP BY floor;
```

층별로 주차 가능한 자리가 얼마나 있는지 집계한다.

Top-3 VIP 검색

```
SELECT vehicle,
    SEC_TO_TIME(SUM(t.diffsec) / SUM(t.diffdate)) AS average
FROM (
    SELECT vehicle,
        DATEDIFF(COALESCE(exit_at, NOW()), enter_at) + 1
            AS diffdate,
        SUM(TIME_TO_SEC(TIMEDIFF(COALESCE(exit_at, NOW()), enter_at)))
            AS diffsec
    FROM enter_logs
    LEFT JOIN exit_logs ON enid = enter
    GROUP BY DATE(enter_at)
) AS t
GROUP BY t.vehicle
ORDER BY SUM(t.diffsec) / SUM(diffdate) DESC
LIMIT 3;
```

일 평균 주차 시간이 가장 긴 상위 3대의 차량을 찾는다. 기본적으로 enter_at(입차 시각)과 그에 대응하는 exit_at(출차 시각) 사이 기간으로 집계하며, 출차하지 않은 차량에 대해서는 현재 시각까지의 주차 시간을 집계한다. 만약 기간이 자정을 넘겨 날짜가 바뀐 경우에는 다른 날짜로 집계한다. 즉, 1일 22시에 주차한 차량이 2일 새벽 2시에 출차하는 경우, 주차 기간은 4시간이고, 일수는 이틀이 된다. 따라서 해당 차량의 일 평균 주차 시간은 4 / 2 = 2시간으로 계산한다.

7일 이내에 만료되는 정기권을 가진 고객 검색

```
SELECT DISTINCT cid, name, email, phone
FROM tickets INNER JOIN customers ON owner = cid
WHERE expired_at > NOW() AND expired_at <= NOW() + INTERVAL 7 DAY;
```

특정 기간의 수익 집계

```
SELECT (
    SELECT SUM(amount)
    FROM paid_for_settlement_logs
    WHERE pspaid_at
        BETWEEN "2023-06-01 15:30:00" AND "2023-06-20 08:10:00"
) AS settlement_income, (
    SELECT SUM(typ.amount)
    FROM paid_for_ticket_logs AS p
        INNER JOIN paid_for_ticket_types AS typ
            ON p.pttype = typ.pttype
    WHERE p.ptpaid_at
        BETWEEN "2023-06-01 15:30:00" AND "2023-06-20 08:10:00"
) AS ticket_income;
```

특정 기간 동안 정기권 결제로 발생한 총 수입과 정산 결제로 발생한 총 수입을 집계한다.

출입구별 혼잡도 집계

```
SELECT COALESCE(enter_to, exit_from) AS entrance,
    COALESCE(enter_count, 0) + COALESCE(exit_count, 0) AS total
FROM (
    (SELECT enter_to, COUNT(*) AS enter_count FROM enter_logs
        GROUP BY enter_to) AS enter_counts
    LEFT JOIN
        (SELECT exit_from, COUNT(*) AS exit_count FROM exit_logs
            GROUP BY exit_from) AS exit_counts
    ON enter_counts.enter_to = exit_counts.exit_from
) UNION (
    SELECT COALESCE(enter_to, exit_from) AS entrance,
        COALESCE(enter_count, 0) + COALESCE(exit_count, 0) AS total
    FROM
        (SELECT enter_to, COUNT(*) AS enter_count FROM enter_logs
             GROUP BY enter_to) AS enter_counts
    RIGHT JOIN
        (SELECT exit_from, COUNT(*) AS exit_count FROM exit_logs
             GROUP BY exit_from) AS exit_counts
    ON enter_counts.enter_to = exit_counts.exit_from
) ORDER BY total DESC;
```

각 출입구별 입출차 건수를 구하고, 입출차 건수의 총합이 많은 출입구 순으로 집계한다. MySQL이 FULL OUTER JOIN을 지원하지 않아 LEFT JOIN과 RIGHT JOIN한 결과를 UNION했다.