Give Your Money

데이터베이스 프로젝트

2019102203 윤성배, 2019102236 최현영

2023.06.17

목차

프로젝트 필요성

문제 정의

- 신체적인 활동을 통해 체력을 증진시키고, 근력을 강화하며 유연성과 몸의 균형을 향상시킬 수 있다.
- 일상 생활에서의 스트레스를 해소하며 긍정적인 에너지를 얻을 수 있다.
- 바디 프로필 유행에 따른 몸의 가시적인 변화를 몸소 느끼고 싶어하는 욕구를 충족.
- 다양한 사람들과 소통하고 교류하며 친분을 쌓기 위함.

⇒ 이러한 이유로 수요가 많아지고 있는 현재, **헬스장을 운영하는 입장에서 DBMS를 통한 데이터 통합 관리 덕분에 직원의 업무 프로세스 개선과 효율성 향상 및 서비스 양질에 긍정적인 영향을 미칠 것으로 사료됨.**

프로젝트 필요성

DBMS를 통한 데이터 관리의 필요성

- 갈수록 많아지는 회원들의 정보를 보다 효율적으로 관리
- 회원 정보에 기반한 다양한 서비스를 제공
- 운동기구 및 편의시설과 같은 사내 물품을 효율적으로 운영
- 직원의 관리를 자동화 함으로써 업무 효율의 증대

Give Your Money만의 특별함

- 요구사항은 **이윤 극대화**다

  → 헬스장이 이윤을 남길 수 있는 요소는 PT와 등록이다

  → 회원들의 정보를 통해 PT와 등록을 유도할 수 있는 이벤트를 한다

  - 전업주부가 많다면 오후에 단체 PT를 하는 이벤트
  - 학생이 많다면 친구 데리고 올 시 할인 또는 개월 수 늘려주는 이벤트
  - 회사원이 많다면 저녁에 사람들이 몰리는 시간대를 예상

    → 그 시간대에 회원들에게 다가가서 사람이 없는 기구로 유도

    → 회원들이 불편함 해소, PT 유도도 가능

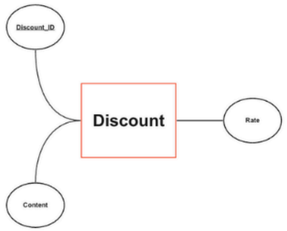
- 또 다른 요구사항은 운영 자동화다
  - **매일 자동으로 당번 최신화**
  - 데이터베이스 작업 부분 자동화

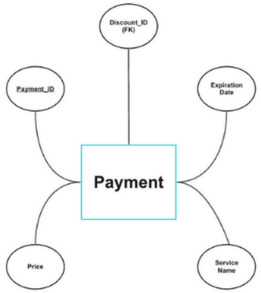
프로젝트 완성 후의 기대 효과

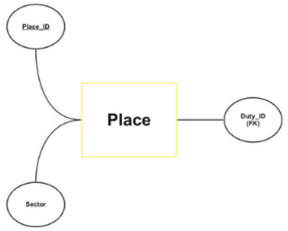
- 데이터 일관성 유지 → 헬스장의 데이터를 효율적으로 관리
- 당번 관리 자동화 → 업무 효율이 증대됨.

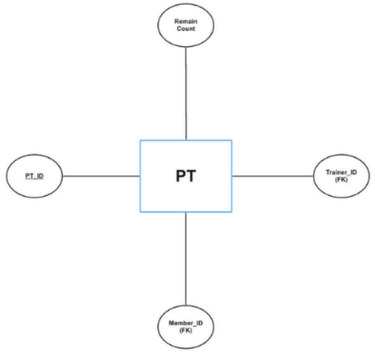
  ⇒ 회원에게 질 좋은 서비스를 제공하는데 집중
- 연령대 혹은 직업별로 특색화된 서비스를 제공 → 수익 창출 효과 기대
- 회원의 운동기록 및 성과, 목표량에 기반한 맞춤형 서비스를 제공

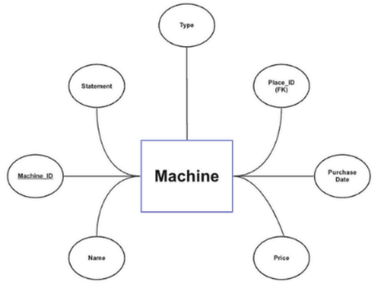
실세계 모델들의 마인드맵 도식화

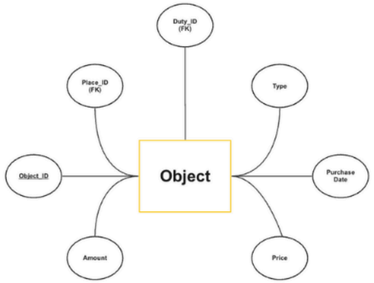


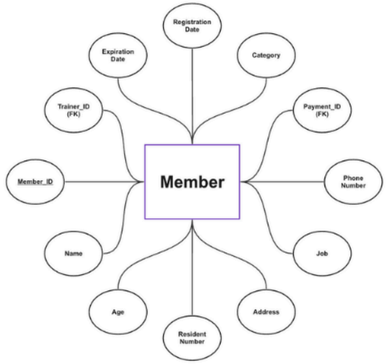


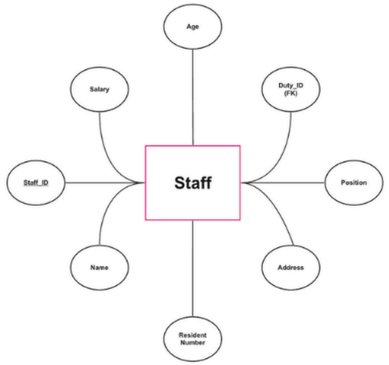


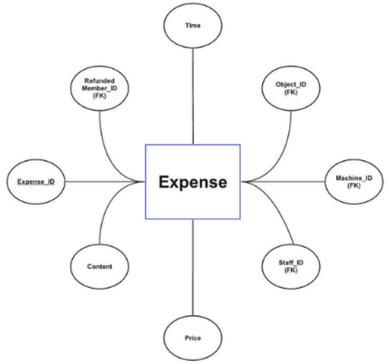




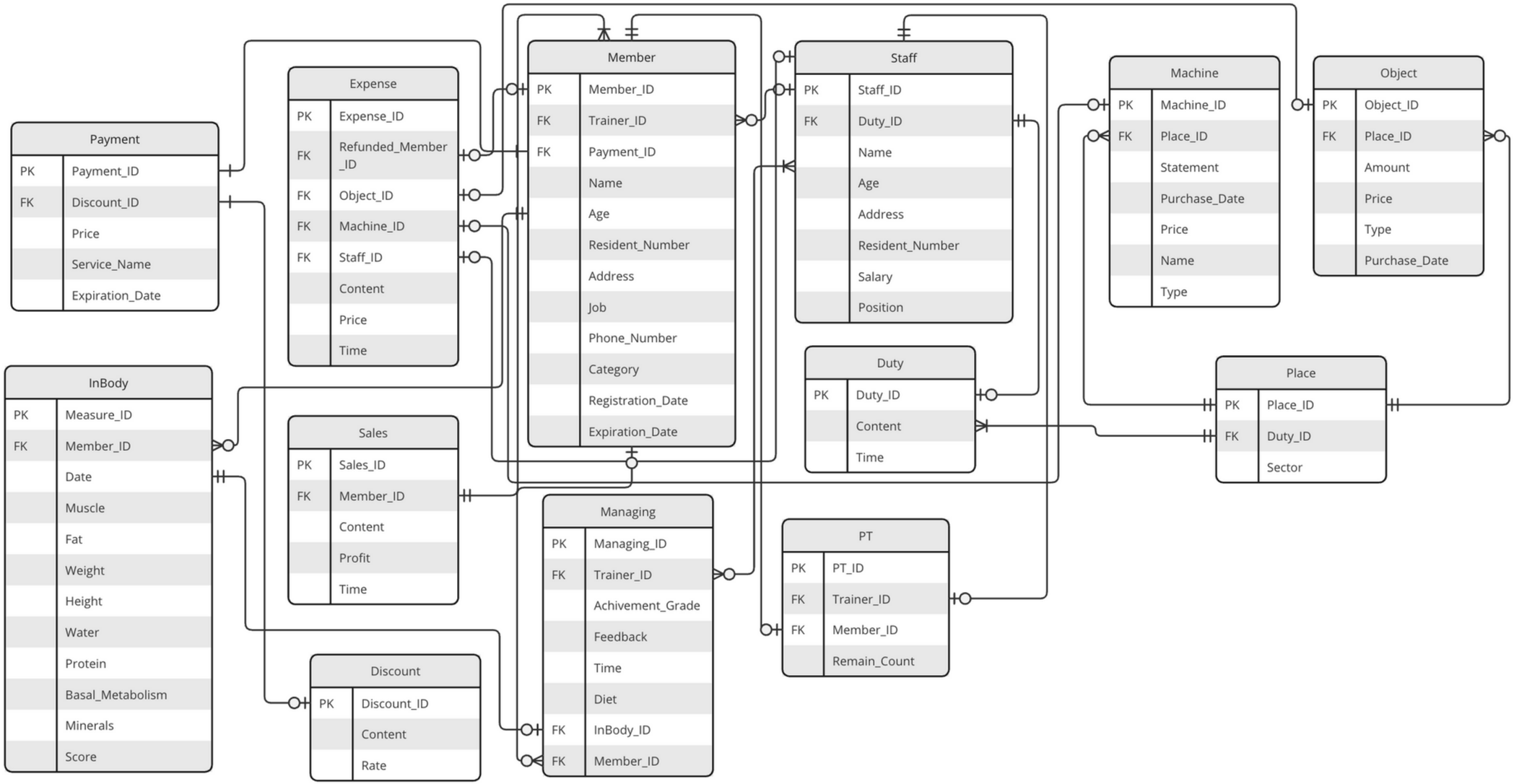








Give Your Money



RDBMS 선정

- MySQL
- 오픈소스 RDBMS.

  ⇒ 성능이 보장되며 신뢰성있고, 널리 사용되고 있는 대중성을 고려
- 다중 사용자를 지원
- 다양한 프로그래밍 언어를 위한 API를 지원

  ⇒ 서버 구축 시 언어에 대한 호환성을 체크 여부를 하지 않아도 됨

Payment

```sql
CREATE TABLE payment (
  payment_id INT PRIMARY KEY AUTO_INCREMENT,
  price INT NOT NULL,
  duration INT NOT NULL
);

-- 제약조건
ALTER TABLE payment
ADD CONSTRAINT CHECK (price > 0 AND duration > 0);
```

Expense

```sql
CREATE TABLE Expense (
  Expense_ID INT PRIMARY KEY AUTO_INCREMENT,
  Object_ID INT,
  Machine_Id INT,
  Staff_ID INT,
  Content VARCHAR(500) NOT NULL,
  Price INT NOT NULL,
  Time TIMESTAMP NOT NULL,
  CONSTRAINT FK_Object_ID FOREIGN KEY (Object_ID) REFERENCES Object(Object_ID),
  CONSTRAINT FK_Machine_Id FOREIGN KEY (Machine_Id) REFERENCES Machine(Machine_Id),
  CONSTRAINT FK_Staff_ID FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID)
);

-- 제약조건
ALTER TABLE expense
ADD CONSTRAINT CHECK (price > 0);
```

Staff

```sql
CREATE TABLE Staff (
  Staff_ID INT AUTO_INCREMENT PRIMARY KEY,
  Duty_ID INT,
  Name VARCHAR(5) NOT NULL,
  Age INT NOT NULL,
  Address VARCHAR(100) NOT NULL,
  Sex CHAR(2) NOT NULL,
  Salary INT NOT NULL,
  Position CHAR(10) NOT NULL,
  FOREIGN KEY (Duty_ID) REFERENCES Duty(Duty_ID)
);

-- 제약조건
ALTER TABLE staff
ADD CONSTRAINT CHECK (sex in ('M', 'F')),
ADD CONSTRAINT CHECK (position in ('Trainer', 'Boss', 'Manager'));
```

Duty

```sql
CREATE TABLE duty (
  duty_id INT PRIMARY KEY AUTO_INCREMENT,
  place VARCHAR(100) NOT NULL,
  time INT NOT NULL
);

-- 제약조건
ALTER TABLE duty
ADD CONSTRAINT CHECK (place in
  ('free weight', 'pilates', 'stretching', 'machine', 'dressing room'));
```

Object

```
CREATE TABLE object (
  object_id INT PRIMARY KEY AUTO_INCREMENT,
  place VARCHAR(100) NOT NULL,
  amount INT NOT NULL,
  price INT NOT NULL,
  purchase_date TIMESTAMP not null
)

-- 제약조건
ALTER TABLE object
ADD CONSTRAINT CHECK (amount >= 0 and price > 0),
ADD CONSTRAINT CHECK (place in
('free weight', 'pilates', 'stretching', 'machine', 'dressing room'));
```

Managing

```
CREATE TABLE managing (
  managing_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT NOT NULL,
  trainer_id INT NOT NULL,
  grade VARCHAR(100) NOT NULL,
  feedback VARCHAR(100) NOT NULL,
  date TIMESTAMP NOT NULL,
  FOREIGN KEY (member_id) REFERENCES member(member_id),
  FOREIGN KEY (trainer_id) REFERENCES staff(staff_id),
  FOREIGN KEY (managing_id) REFERENCES inbody(inbody_id) ON DELETE CASCADE;
);
```

Member

```
CREATE TABLE Member (
  Member_ID INT AUTO_INCREMENT PRIMARY KEY,
  Trainer_ID INT,
  Payment_ID INT NOT NULL,
  Name VARCHAR(5) NOT NULL,
  Age INT NOT NULL,
  Sex VARCHAR(2) NOT NULL,
  Address VARCHAR(50) NOT NULL,
  Job VARCHAR(20) NOT NULL,
  Phone_Number VARCHAR(12) NOT NULL UNIQUE,
  Category CHAR(10),
  Registration_Date TIMESTAMP NOT NULL,
  Expiration_Date TIMESTAMP NOT NULL,
  FOREIGN KEY (Trainer_ID) REFERENCES Staff(Staff_ID),
  FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID)
);

-- 제약조건
ALTER TABLE member
ADD CONSTRAINT
FOREIGN KEY (trainer_id)
REFERENCES staff(staff_id)
ON DELETE SET NULL;

ALTER TABLE member
ADD CONSTRAINT CHECK (category in ('PT', 'Pilates', 'Free')),
ADD CONSTRAINT CHECK (sex in ('M', 'F'));
```

Discount

```
CREATE TABLE discount (
  discount_id INT PRIMARY KEY AUTO_INCREMENT,
  rate FLOAT(3,2) NOT NULL
);

-- 제약조건
ALTER TABLE discount
ADD CONSTRAINT CHECK (rate > 0.0);
```

Sales

```
CREATE TABLE Sales (
  Sales_ID INT PRIMARY KEY,
  Member_ID INT,
  Discount_ID INT,
  Content VARCHAR(50) NOT NULL,
  Profit INT not null,
  Time TIMESTAMP NOT NULL,
  CONSTRAINT FK_Member_ID FOREIGN KEY (Member_ID)
    REFERENCES Member(Member_ID),
  CONSTRAINT FK_Discount_ID FOREIGN KEY (Discount_ID)
    REFERENCES Discount(Discount_ID)
);
```

Machine

```
CREATE TABLE machine (
  machine_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  price INT NOT NULL,
  purchase_date TIMESTAMP NOT NULL,
  state VARCHAR(100) NOT NULL,
  type VARCHAR(100) NOT NULL,
  place VARCHAR(100) NOT NULL
);

-- 제약조건
ALTER TABLE machine
ADD CONSTRAINT CHECK (price > 0),
ADD CONSTRAINT CHECK (place in
  ('free weight', 'pilates', 'stretching', 'machine', 'dressing room')),
ADD CONSTRAINT CHECK (type in
  ('aerobic', 'chest', 'back', 'arm', 'leg', 'shoulder')),
ADD CONSTRAINT CHECK (state in ('good', 'soso', 'bad'));
```

Inbody

```
CREATE TABLE inbody (
  Inbody_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT NOT NULL,
  date TIMESTAMP NOT NULL,
  muscle FLOAT(3,1) NOT NULL,
  fat FLOAT(3,1) NOT NULL,
  weight FLOAT(4,1) NOT NULL,
  height FLOAT(4,1) NOT NULL,
  water FLOAT(3,1) NOT NULL,
  protein FLOAT(3,1) NOT NULL,
  basal_metabolism INT NOT NULL,
  minerals FLOAT(3,2) NOT NULL,
  FOREIGN KEY (member_id) REFERENCES member(member_id)
);

-- 제약조건
ALTER TABLE inbody
ADD CONSTRAINT inbody_cascade FOREIGN KEY (member_id)
    REFERENCES member(member_id)
    ON DELETE CASCADE;
```

PT

```
CREATE TABLE pt (
  pt_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT NOT NULL,
  remain_count INT NOT NULL,
  FOREIGN KEY (member_id) REFERENCES member(member_id) ON DELETE CASCADE
);

-- 제약조건
ALTER TABLE pt
ADD CONSTRAINT CHECK (remain_count >= 0),
```

인덱스

- 가장 중요한, 가장 많이 쓰이는 질의가 무엇일까?
  - 회원의 Inbody 검색

```
CREATE INDEX idx_private_inbody ON inbody (member_id, date DESC)
```

  - 이벤트 관리자의 회원 검색

```
CREATE INDEX idx_member_name ON member (name);
```

  - 운동기구 관리자의 기구 검색

```
CREATE INDEX idx_machine ON machine (name);
```

만료 여부 확인

- 매일 정각마다 회원들이 만료 여부 확인
- 만료됐다면 요금제ID, 등록 일자, 만료 일자를 NULL로 교체

```
DELIMITER //

CREATE PROCEDURE check_member_expiration()
BEGIN
DECLARE currentDate TIMESTAMP;
SET currentDate = CURRENT_TIMESTAMP;

UPDATE member
SET expiration_date = NULL, payment_id = NULL, registration_date = NULL
WHERE expiration_date < currentDate;

END //

DELIMITER ;

CREATE EVENT daily_member_expiration_check
ON SCHEDULE
   EVERY 1 DAY
   STARTS CURDATE() + INTERVAL 1 DAY
DO
CALL check_member_expiration();
```

당번 자동 갱신

- 정각마다 자동 갱신
- 할 일의 수는 직원의 수와 같으며 할 일이 변경되거나 직원이 추가, 삭제되지 않는다고 가정
- 할 일 목록
    1. Pilates Zone: 물품 정리, 바닥+거울 청소
    2. Stretching Zone: 물품 정리, 바닥+거울 청소
    3. Free weight Zone: 원판 정리, 거울 청소
    4. Machine Zone: 기구 정리, 바닥 청소
    5. Dressing room Zone: 빨래, 샤워실 물청소

```
DELIMITER //

CREATE EVENT renew_duty_event
ON SCHEDULE
    EVERY 1 DAY
    STARTS CURDATE() + INTERVAL 1 DAY
DO
    BEGIN
        DECLARE max INT;
        SET max = (SELECT COUNT(*) FROM duty);
        UPDATE staff
        SET duty_id = CASE
            WHEN (duty_id + 1) > max THEN 1
            ELSE (duty_id + 1)
        END;
    END //

DELIMITER ;
```

할인 적용

- INSERT된 Member(회원) 테이블에 Discount_Id 애트리뷰트가 NULL이 아니라면, 할인을 받은 회원이다.

  따라서, IF 구문을 통해 Discount_Id에 해당하는 Rate(할인율) 값을 통해 계산하고 Sales(매출)에 계산된

  최종 지불 가격을 장부에 기록한다.

```
DELIMITER //

CREATE TRIGGER apply_discount
AFTER INSERT ON member
FOR EACH ROW
BEGIN
DECLARE result INT;
SET result = (SELECT price FROM payment WHERE payment_id = NEW.payment_id);
IF NEW.discount_id IS NOT NULL THEN
  SET result = result * (1 - (SELECT rate FROM discount
                              WHERE discount_id = NEW.discount_id));
END IF;

INSERT INTO sales(member_id, content, profit, time)
VALUES (NEW.member_id, 'registration', result, CURRENT_TIMESTAMP());
END //

DELIMITER ;
```

회원 신규 등록

- INSERT된 Member(회원)테이블의 registratoin_date(등록일)과 Payment의 Duration(이용기간) 애트리뷰트를 계산하여 Expiratoin_Date(만료 기간)에 자동으로 기입한다.

```
DELIMITER //
CREATE TRIGGER trg_expiration_date
Before INSERT ON member
FOR EACH ROW
BEGIN
declare duration_date int;
SELECT duration INTO duration_date FROM Payment
                                   WHERE payment_id = new.Payment_ID;
set new.expiration_date = timestamp(DATE_ADD(new.registration_date,
                          interval duration_date MONTH));
end //
DELIMITER ;
```

환불

- Member(회원) 테이블의 투플이 삭제되었다고 한다면, 탈퇴한 회원에게 남은 기간만큼 비례해서 환불한다.
- 환불된 금액은 Expense(지출)의 내역에 자동으로 생성된다.
- 1 개월은 30일로 가정한다.

$$originMoney * (remainDay / (originDay * 30))$$

```
DELIMITER //
CREATE TRIGGER before_delete_member
BEFORE DELETE ON member
FOR EACH ROW
BEGIN
DECLARE origin_money INT;
DECLARE tmp1 TIMESTAMP;
DECLARE remain_day INT;
DECLARE origin_day INT;
DECLARE result_money INT;

SELECT profit INTO origin_money FROM sales WHERE member_id = OLD.member_id;
SELECT expiration_date INTO tmp1 FROM member WHERE member_id = OLD.member_id;
SET remain_day = DATEDIFF(tmp1, CURRENT_TIMESTAMP());

IF (remain_day > 0) THEN
SELECT duration INTO origin_day FROM payment WHERE payment_id = old.payment_id;
SET result_money = CAST(origin_money * (CAST(remain_day AS DECIMAL(4, 1)) / (origin_day * 30)) AS UNSIGNED);
INSERT INTO expense (content, price, time) VALUES ('refund', result_money, CURRENT_TIMESTAMP());
END IF;
END //
DELIMITER ;
```

향후 계획

- 당번 최신화 알고리즘 개선
  - 현재, 직원수와 당번의 할 일의 갯수가 동일해야만 작동하는 방식
- Object 및 Machine 테이블에 대한 트리거 검토 및 추가
- 현재 root 권한의 유저만 접근
  - 직원 별도의 사용자를 생성하여 권한 부여

    → 무분별한 데이터 접근 차단 및 보안 강화