📄 게임 데이터베이스 쿼리문 정리 (MySQL)

Hunter in the Gate — Database Programming Queries

1. 스키마 생성 / 기본 환경 구성

```
-- 1-1. 새 스키마 생성
CREATE SCHEMA `new_schema`;

-- 1-2. 헌터 인 더 게이트 데이터 스키마 선택
USE `hunter_data`;
```

2. 테이블 생성 실습 (PK, AUTO_INCREMENT, 복합키)

```
-- 2-1. 실습용 new_table (AUTO_INCREMENT + 복합 PK 실험)
CREATE TABLE `hunter_data`.`new_table` (
  `level` INT NOT NULL,
  `hp` INT NOT NULL AUTO_INCREMENT,
  `exp` INT NOT NULL,
  PRIMARY KEY (`level`, `hp`, `exp`)
);

-- 2-2. 실습용 player 테이블 생성 (복합 PK 실습)
CREATE TABLE `hunter_data`.`player` (
  `level` INT NOT NULL,
  `hp` INT NOT NULL AUTO_INCREMENT,
  `exp` INT NOT NULL,
  PRIMARY KEY (`level`, `exp`)
);
```

3. 카드 데이터 정제 및 최종 carddata 테이블 확정

```sql
-- 3-1. 임시 테이블(data_utf8_clean)을 게임용 carddata 테이블로 정제
ALTER TABLE `hunter_data`.`data_utf8_clean`
  CHANGE COLUMN `카드ID`             `카드ID`             VARCHAR(50)  NOT NULL FIRST,
  CHANGE COLUMN `카드명`             `카드명`             VARCHAR(100) NOT NULL,
  CHANGE COLUMN `카드타입`           `카드타입`           VARCHAR(50)  NOT NULL,
  CHANGE COLUMN `희귀도`             `희귀도`             VARCHAR(20)  NOT NULL,
  CHANGE COLUMN `코스트`             `코스트`             INT          NOT NULL,
  CHANGE COLUMN `레벨`               `레벨`               INT          NOT NULL,
  CHANGE COLUMN `카드효과ID`         `카드효과ID`         VARCHAR(50)  NOT NULL,
  CHANGE COLUMN `설명`               `설명`               TEXT         NOT NULL,
  CHANGE COLUMN `카드 이미지 파일명` `카드 이미지 파일명` VARCHAR(191) NOT NULL,
  ADD PRIMARY KEY (`카드ID`),
  RENAME TO `hunter_data`.`carddata`;
```

4. 인벤토리(inventory) 테이블 초안 설계

```sql
-- 4-1. 플레이어 보유 카드 리스트를 저장하는 테이블 초안
CREATE TABLE `hunter_data`.`inventory` (
  `InventoryID` INT NOT NULL,
  `보유카드 ID` VARCHAR(500) NOT NULL,
  PRIMARY KEY (`InventoryID`)
);
```

5. 스키마 전환 및 데이터 조회

```sql
-- 5-1. 스키마 전환(실습)
USE `game_data`;

-- 5-2. 다시 헌터 인 더 게이트 스키마로 복귀
USE `hunter_data`;

-- 5-3. carddata 테이블 조회
SELECT * FROM `hunter_data`.`carddata`;
```

6. 트랜잭션 실습 (ROLLBACK / COMMIT)

```sql
-- 6-1. 트랜잭션 실습을 위한 스키마 선택
USE `hunter_data`;

-- 6-2. [실습 1] 삭제 후 ROLLBACK 테스트
START TRANSACTION;
SELECT * FROM `cards`;          -- 삭제 전 상태 확인
DELETE FROM `cards` WHERE id = 2;
ROLLBACK;                       -- 삭제 취소 → id = 2는 유지됨

-- 6-3. [실습 2] 삭제 후 COMMIT 테스트
START TRANSACTION;
SELECT * FROM `cards`;          -- 현재 상태 확인
DELETE FROM `cards` WHERE id = 1;
COMMIT;                         -- 삭제 확정 → id = 1 영구 삭제

-- 6-4. [실습 3] COMMIT 후 ROLLBACK 테스트
ROLLBACK;                       -- 커밋된 작업은 되돌릴 수 없음
```