CHAPTER 11

보안과 권한 관리

Database Security & Privilege Management

박상돈 교수 대전대학교 컴퓨터공학과

AxGS Lab · AI × Game Systems

CONTENTS

오늘 배울 내용

01

보안

데이터베이스를 위협에서 보호하는 세 가지 방법

02

권한 관리

GRANT · REVOKE · ROLE 로 접근을 통제하는 방법

01

보안

Database Security

데이터베이스 보안이란?

DB의 보안을 유지하여 데이터를 안전하게 보호하는 방법

① 물리적 환경 보안

화재 · 홍수 · 정전 같은 물리적 손실의 위험에서 데이터를 보호합니다.

장비실 출입 통제, 이중화, 백업 등이 여기에 해당합니다.

② 권한 관리 보안

권한이 허락되지 않은 사용자로부터 보호합니다.

로그인한 사람이라도 허가된 데이터에만 접근하도록 통제합니다. — 오늘의 핵심

③ 운영 관리 보안

권한이 허락된 사용자로부터도 DB를 보호합니다.

실수·악용을 막기 위한 운영 절차와 감사(audit) 가 필요합니다.

데이터베이스 보안의 유형

그림 11-1 — 보호 대상에 따라 세 갈래로 나뉩니다

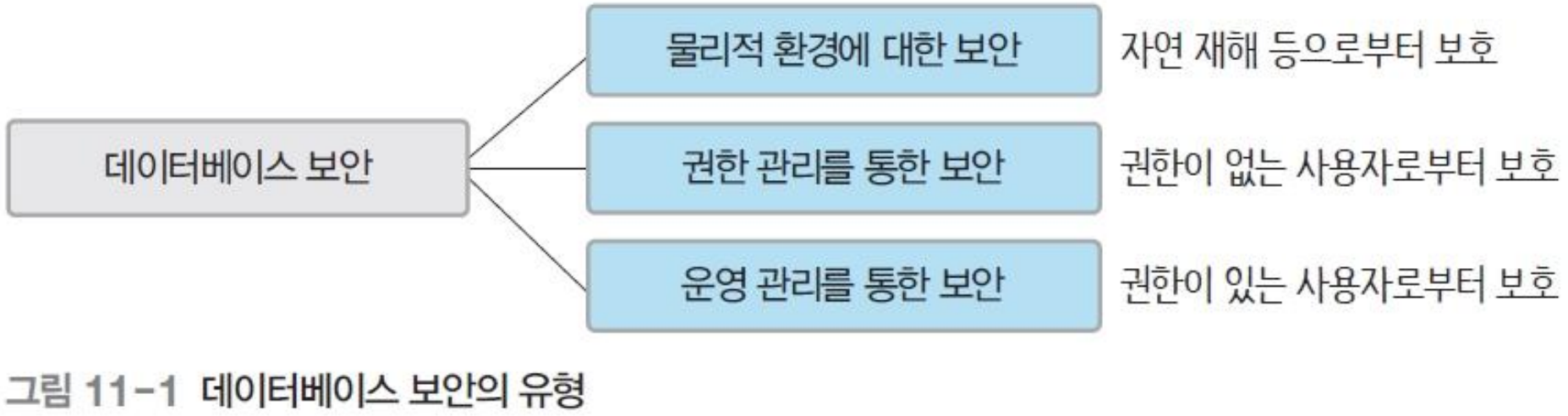


그림 11-1 데이터베이스 보안의 유형

- **물리적 환경 → 자연 재해 등으로부터 보호**
- **권한 관리 → 권한이 없는 사용자로부터 보호**
- **운영 관리 → 권한이 있는 사용자로부터 보호**

보안과 무결성 — 미술관에 비유하면

그림 11-2 — 운영자가 아니면 작품을 만질 수 없습니다

그림 11-2 보안과 무결성 유지

미술관 비유로 이해하기

- **전시된 그림 = 데이터베이스의 데이터**
  - 관람객은 볼 수 있지만 만질 수 없음 → 권한 통제
- **"만지지 마세요" = 무결성 보호 장치**
  - 허가받지 않은 손길이 데이터를 훼손하지 못하게 막는 것
- **운영자(관리자)만 작품을 옮기거나 교체 가능**
  - = DBA·소유자만 구조를 바꾸거나 권한을 줄 수 있음

02

권한 관리

Privilege Management

권한 관리 ① 접근 제어 (Access Control)

그림 11-3 — 로그인에 성공해야만 DB에 접근할 수 있습니다

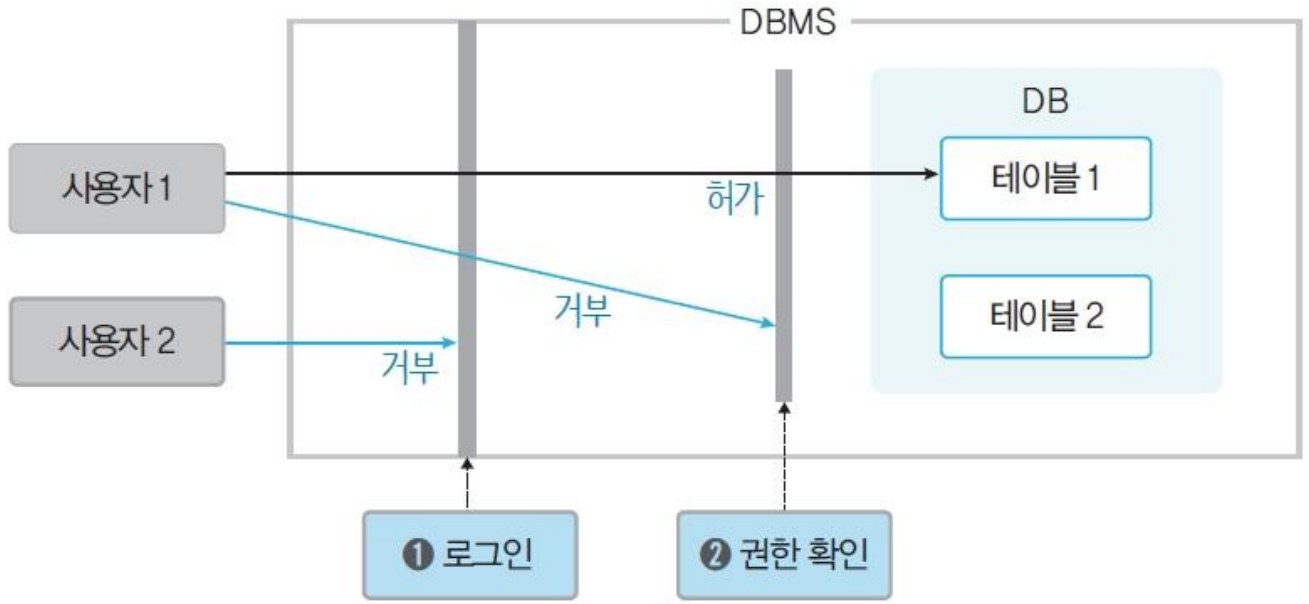


그림 11-3 로그인과 데이터베이스 접근 권한

그림 11-3 로그인과 데이터베이스 접근 권한

DBA 의 역할

- 새 사용자의 계정·암호 생성
- 기존 계정 변경 및 제거
- 전체 사용자 계정 관리

DBMS 의 역할

- 사용자별 DB 사용 범위 제한
- 수행 가능한 작업(읽기·쓰기 등) 제한
- ❶ 로그인 → ❷ 권한 확인

권한 관리 ② 객체 권한 부여

내가 만든 객체가 아니면 함부로 쓸 수 없습니다

기본 원칙

DB 접근이 허락된 사용자라도, 자신이 생성하지 않은 객체(테이블·뷰 등)는 사용할 수 없습니다.

즉 '로그인했다'는 것과 '그 데이터를 쓸 수 있다'는 것은 완전히 다른 이야기입니다.

권한의 전달

객체의 소유자(수요자)는 필요에 따라 다른 사용자에게 자신이 소유한 객체에 대한 사용 권한을 부여할 수 있습니다.

→ 이 '부여'를 명령어로 표현한 것이 바로 GRANT 문이며, 회수는 REVOKE 문입니다.

다음 슬라이드부터: GRANT (부여) → REVOKE (취소) → ROLE (역할로 묶기)

권한 관리의 전체 그림

그림 11-4 — 오늘 배울 명령어들이 한눈에 보입니다

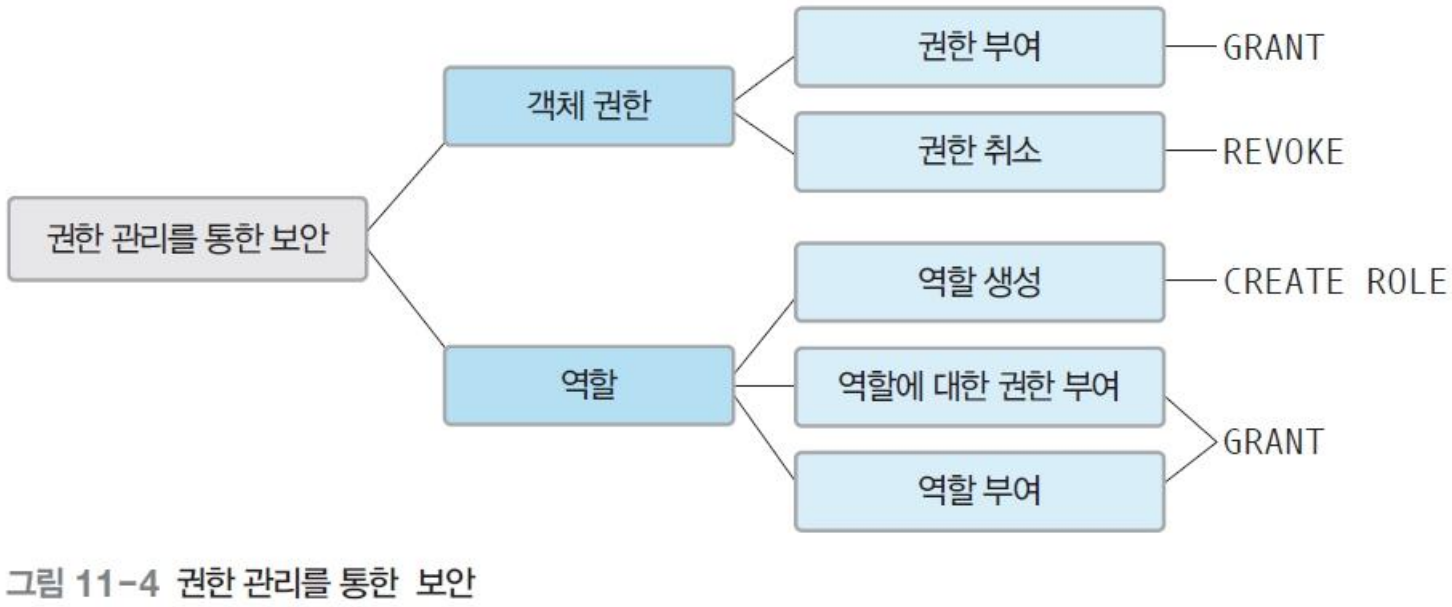


그림 11-4 권한 관리를 통한 보안

그림 11-4 권한 관리를 통한 보안

객체 권한

- 권한 부여 → GRANT
- 권한 취소 → REVOKE

역할(ROLE)

- 역할 생성 → CREATE ROLE
- 역할에 권한 부여 → GRANT
- 역할 부여 → GRANT

권한의 부여 — GRANT 문

객체의 소유자가 다른 사용자에게 사용 권한을 부여합니다

SYNTAX

```
GRANT 권한 ON 객체 TO 사용자 [WITH GRANT OPTION];
```

부여할 수 있는 권한

SELECT	INSERT	UPDATE	DELETE	REFERENCES
검색	삽입	수정	삭제	외래키 정의

- **PUBLIC : 모든 사용자에게 한꺼번에 권한을 부여**
- **WITH GRANT OPTION : 부여받은 권한을 또 다른 사용자에게 넘길 수 있는 옵션**

GRANT 예제 ① — 기본 부여와 PUBLIC

특정 사용자에게, 또는 모든 사용자에게 권한을 줍니다

예제 11-1

고객 테이블에 대한 검색 권한을 사용자 Hong에게 부여해보자.

```
GRANT SELECT ON 고객 TO Hong;
```

예제 11-2

고객 테이블에 대한 삽입·삭제 권한을 모든 사용자에게 부여해보자
.

```
GRANT INSERT, DELETE ON 고객 TO PUBLIC;
```

GRANT 예제 ② — 일부 속성과 권한 이양

속성을 콕 집어 주거나, 권한을 넘길 자격까지 함께 줍니다

예제 11-3

고객 테이블 속성 중 '등급·적립금' 수정 권한만 사용자 Park에게 부여해보자.

```
GRANT UPDATE(등급, 적립금) ON 고객 TO Park;
```

예제 11-4

고객 테이블 검색 권한을 WITH GRANT OPTION 을 포함해 사용자 Lee에게 부여해보자.

```
GRANT SELECT ON 고객 TO Lee
  WITH GRANT OPTION;
```

WITH GRANT OPTION 을 받은 Lee는, 자기가 받은 검색 권한을 또 다른 사람에게 부여할 수 있습니다.

시스템 권한의 부여 — 예제 11-5 · 11-6

테이블·뷰를 '만들 수 있는' 권한은 객체를 지정하지 않습니다

예제 11-5

테이블을 생성할 수 있는 시스템 권한을 사용자 Song에게 부여해보보자.

```
GRANT CREATE TABLE TO Song;
```

예제 11-6

뷰를 생성할 수 있는 시스템 권한을 사용자 Shin에게 부여해보자.

```
GRANT CREATE VIEW TO Shin;
```

포인트 시스템 권한(생성 권한)은 **DBA**가 부여하며, ON 절(예: ON 고객)이 필요 없습니다 — 특정 객체가 아닌 '기능'을 주기 때문입니다.

권한의 취소 — REVOKE 문

부여한 권한을 회수합니다. 두 가지 옵션이 핵심입니다

SYNTAX

```
REVOKE 권한 ON 객체 FROM 사용자 CASCADE | RESTRICT;
```

CASCADE (연쇄 취소)

연쇄적으로 부여된 권한까지 함께 취소합니다.

내가 준 권한을, 받은 사람이 또 다른 사람에게 넘겨준 경우 — 그 줄기 전체를 한 번에 회수합니다.

→ '뿌리째 뽑기'

RESTRICT (제한)

이미 다른 사람에게 권한이 넘어가 있으면, 취소를 거부(막음)합니다.

다른 곳에 영향이 없을 때만 안전하게 취소되도록 보호하는 옵션입니다.

→ '영향 있으면 멈춤'

권한이 연쇄적으로 전달되는 경우

그림 11-5 — WITH GRANT OPTION 으로 권한이 줄줄이 이어집니다

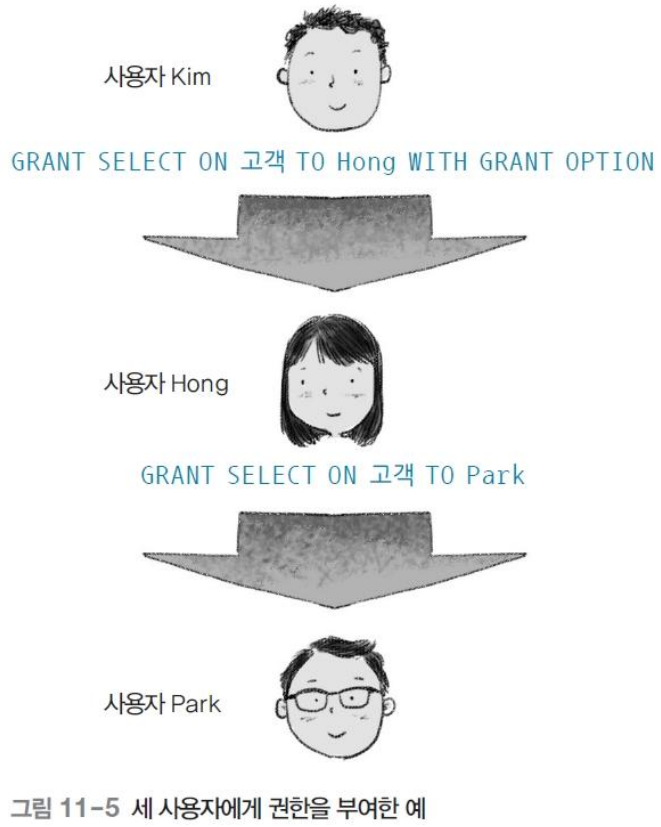


그림 11-5 세 사용자에게 권한을 부여한 예

그림 11-5 세 사용자에게 권한을 부여한 예

권한이 어떻게 흘러갔나?

- **Kim → Hong : 검색 권한 (WITH GRANT OPTION)**
  - 옵션이 있으므로 Hong도 남에게 줄 수 있음
- **Hong → Park : 검색 권한 전달**
  - 이제 Park도 고객 테이블을 검색할 수 있음
- **여기서 Kim이 Hong의 권한을 취소하면?**
  - CASCADE → Park 권한까지 함께 사라짐
  - RESTRICT → Park가 있으므로 취소 거부됨

REVOKE 예제 — CASCADE vs RESTRICT

같은 상황, 옵션 하나로 결과가 완전히 달라집니다

예제 11-7

Kim이 Hong에게 준 검색 권한을 취소하면서, Hong이 남에게 준 검색 권한도 함께 취소하자.

```
REVOKE SELECT ON 고객 FROM Hong CASCADE;
```

예제 11-8

Hong이 다른 사용자에게 권한을 준 적이 없을 때만, Kim이 Hong의 검색 권한을 취소하자.

```
REVOKE SELECT ON 고객 FROM Hong RESTRICT;
```

CASCADE 는 Park의 권한까지 지우고, **RESTRICT** 는 Park가 있으면 아예 취소를 거부합니다.

사용자별 권한 정리하기

표 11-1 — 소유자는 누가 무엇을 할 수 있는지 정리해 둬야 합니다

표 11-1 고객 테이블에 대한 각 사용자의 권한 목록

사용자 \ 권한	고객 테이블에 대한 권한
Kim	소유자
Hong	INSERT / DELETE / SELECT
Park	INSERT / DELETE / UPDATE(등급, 적립금)
Lee	INSERT / DELETE / SELECT(WITH GRANT OPTION)

표 11-1 고객 테이블에 대한 각 사용자의 권한 목록

왜 정리가 필요할까

권한이 여러 사람에게 흩어지면 누가 무엇을 할 수 있는지 추적하기 어렵습니다.

- Kim = 소유자
- Hong = 삽입·삭제·검색
- Park = 삽입·삭제·일부 수정
- Lee = 검색(+이양 가능)

소유자는 이 표를 관리하며 보안 사고를 예방합니다.

시스템 권한의 취소 — 예제 11-9

생성 권한도 REVOKE 로 회수합니다 (객체 지정 없음)

예제 11-9

Hong에게 부여한 '테이블 생성' 시스템 권한을 취소해보자.

```
REVOKE CREATE TABLE FROM Hong;
```

객체 권한 vs 시스템 권한

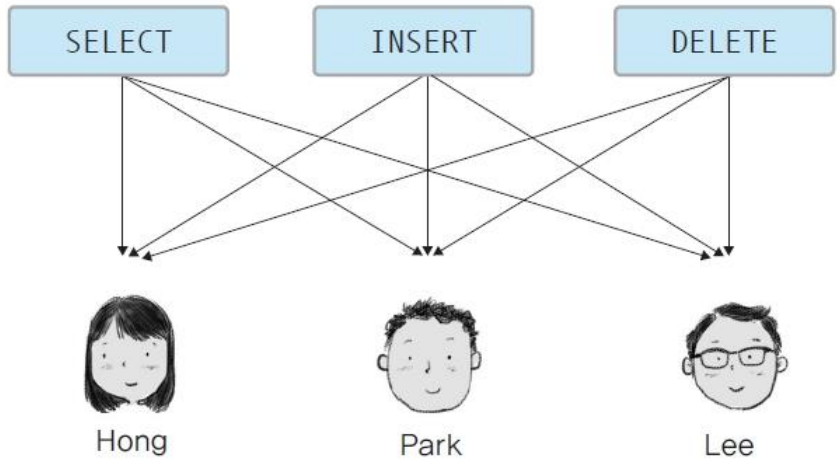
객체 권한 취소:
REVOKE ... ON 고객 FROM ...
→ 특정 테이블을 가리킴

시스템 권한 취소:
REVOKE CREATE TABLE FROM ...
→ ON 절이 없음

'무엇을 만질지'가 아니라 '무슨 기능을 쓸지'를 다루기 때문입니다.

왜 역할(ROLE)이 필요할까?

그림 11-6 — 같은 권한을 여러 명에게 일일이 주면 너무 번거롭습니다




```
GRANT SELECT ON 고객 TO Hong;
GRANT INSERT ON 고객 TO Hong;
GRANT DELETE ON 고객 TO Hong;

GRANT SELECT ON 고객 TO Park;
GRANT INSERT ON 고객 TO Park;
GRANT DELETE ON 고객 TO Park;

GRANT SELECT ON 고객 TO Lee;
GRANT INSERT ON 고객 TO Lee;
GRANT DELETE ON 고객 TO Lee;
```

그림 11-6 3개의 권한을 사용자 세 명에게 부여하는 예

그림 11-6 3개의 권한을 사용자 세 명에게 부여하는 예

이게 왜 문제일까

- 3개 권한 × 3명 = GRANT 문 9개
- 사람이 늘면 명령문이 폭발적으로 증가
- 권한을 바꾸면 모든 사람에게 다시 실행
- → 관리가 복잡하고 실수가 생기기 쉬움

해결책 권한을 '바구니(역할)' 하나에 담아 그 바구니만 나눠주자!

권한과 역할 — 바구니에 담는다

그림 11-7 — 여러 권한을 묶은 꾸러미가 바로 '역할'입니다

그림 11-7 권한과 역할

그림 11-7 권한과 역할

ROLE(역할) 의 개념

- **역할 = 여러 권한을 묶어 담는 '바구니'**
  - DBA가 사용자를 위해 빈 바구니를 만들어 줌
- **바구니에 SELECT·INSERT·DELETE 등을 담음**
  - 사용자는 권한 하나하나가 아니라 바구니째 받음
- **권한 부여 작업 = 객체 소유자가 담당**
- **역할을 사용자에게 주는 것 = DBA가 담당**

역할 만들기 — CREATE ROLE

먼저 권한을 담을 빈 바구니부터 만듭니다

SYNTAX

```
CREATE ROLE 롤이름;
```

예제 11-10

role_1 이라는 이름의 역할을 생성해보자.

```
▶ CREATE ROLE role_1;
```

이 시점의 role_1 은 아직 텅 빈 바구니입니다. 다음 단계에서 권한을 담고(GRANT … TO role_1), 그 다음에 사용자에게 통째로 건네줍니다(GRANT role_1 TO 사용자).

역할에 권한 담기 — GRANT … TO 롤

빈 바구니에 실제 권한들을 넣습니다

SYNTAX

```
GRANT 권한 ON 객체 TO 롤이름;
```

예제 11-11

고객 테이블의 검색·삽입·삭제 권한을 [예제 11-10]에서 만든 role_1 역할에 담아보자.

```
▶ GRANT SELECT, INSERT, DELETE ON 고객 TO role_1;
```

핵심 객체와 관련된 권한을 역할에 담는 일은 **객체의 소유자**가 담당합니다. (자기 객체에 대한 권한이니까요)

역할을 사용자에게 — GRANT 롤 TO 사용자

권한이 가득 찬 바구니를 통째로 건네줍니다

SYNTAX

```
GRANT 롤이름 TO 사용자;
```

예제 11-12

검색·삽입·삭제 권한을 담고 있는 role_1 역할을 사용자 Hong에게 부여해보자.

```
GRANT role_1 TO Hong;
```

핵심 역할을 사용자에게 부여하는 일은 **DBA**가 담당합니다. 이제 Hong은 세 개의 권한을 한 번에 갖게 됩니다.

역할로 한 번에 부여하기 — 종합

그림 11-8 — 9개의 GRANT가 단 3줄로 줄었습니다

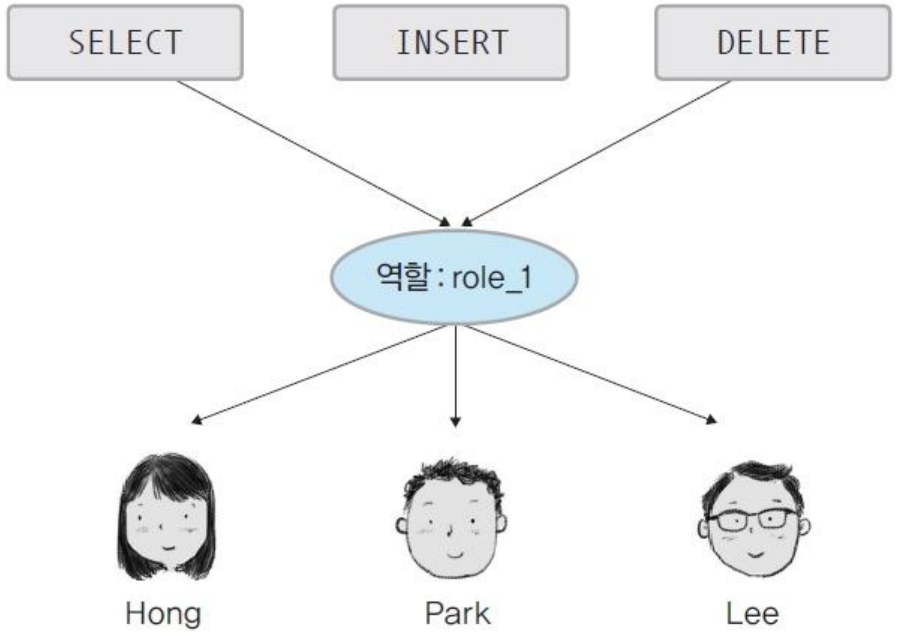


그림 11-8 역할을 이용해 3개의 권한을 세 명의 사용자에게 부여하는 예

```
GRANT role_1 TO Hong;
GRANT role_1 TO Park;
GRANT role_1 TO Lee;
```

그림 11-8 역할을 이용해 3개의 권한을 세 명에게 부여하는 예

무엇이 좋아졌나

- SELECT·INSERT·DELETE → role_1 에 한 번만 담음
- role_1 을 Hong·Park·Lee 에게 부여
- GRANT 문 9개 → 3개로 감소
- 권한을 바꿀 땐 역할만 수정하면 끝
- 사람이 늘어도 '바구니 하나'만 더 주면 됨

역할의 회수와 제거

필요 없어지면 거두고(REVOKE), 아예 없앱니다(DROP)

예제 11-13

사용자 Hong에게 부여한 role_1 역할을 취소해보자.

```
REVOKE role_1 FROM Hong;
```

문법: REVOKE 롤이름 FROM 사용자;

예제 11-14

[예제 11-10]에서 만든 role_1 역할 자체를 제거해보자.

```
DROP ROLE role_1;
```

문법: DROP ROLE 롤이름;

정리 — 두 동작은 다릅니다

REVOKE role_1 FROM Hong → Hong에게서 역할만 떼어냄. 역할 자체(role_1)와 다른 사용자에 대한 부여는 그대로 남습니다.

DROP ROLE role_1 → 역할 자체를 시스템에서 삭제. 이 역할로 받았던 모든 권한이 한꺼번에 사라집니다. (역할 제거는 DBA가 담당)

Wrap-Up — 오늘 배운 것 정리

보안의 큰 그림과 권한 명령어 3종 세트

GRANT — 부여

객체 소유자가 권한을 줌.
GRANT 권한 ON 객체
TO 사용자;
PUBLIC · WITH GRANT OPTION

REVOKE — 취소

준 권한을 회수함.
REVOKE 권한 ON 객체
FROM 사용자;
CASCADE · RESTRICT

ROLE — 묶기

권한을 바구니에 담아 관리.
CREATE ROLE → GRANT 담기
→ GRANT 롤 TO 사용자
REVOKE · DROP ROLE

한 문장 요약

보안은 데이터를 물리적·권한·운영 세 갈래로 지키는 일이고, 그중 **권한 관리**는 GRANT로 주고 REVOKE로 거두며, 반복이 많아지면 ROLE로 묶어 깔끔하게 다스리는 기술입니다.

감사합니다.

11장 보안과 권한 관리를 마칩니다.

Next Class Chapter 12 · 데이터베이스 응용 기술 (빅데이터 · 데이터 과학 개론)

박상돈 교수 · 대전대학교 컴퓨터공학과 · AxGS Lab