03.SQL 활용

1. SQL 데이터 정의문

1.1 테이블 생성 CREATE문

- CREATE TABLE문의 형식

```
CREATE TABLE 테이블_이름❶
  ({ 열_이름 ❷데이터_유형 ❸[NULL|NOT NULL] ❹[DEFAULT 기본_값], }+
     [PRIMARY KEY (열_이름_리스트),]❺
   { [UNIQUE (열_이름_리스트),] }*❻
   { [FOREIGN KEY (열_이름_리스트) REFERENCES 테이블_이름 (열_이름_리스트),] }*❼
  ) ;
```

- 데이터 유형

분류	데이터 유형	저장 유형
문자형	CHAR(길이)	고정길이 문자열, 남은 공간은 공백으로 채움
	VARCHAR(길이)	가변길이 문자열, 남은 공간이 없음
	TEXT	긴 문자열(255자 이상)
정수형	INT	소수점이 없는 기본 정수형
	SMALLINT	소수점이 없는 작은 정수형
실수형	DECIMAL(전체길이,소수점이하길이)	고정 소수점을 포함한 실수
	FLOAT(전체길이,소수점이하길이)	단정도 부동 소수점 실수
	DOUBLE(전체길이,소수점이하길이)	배정도 부동 소수점 실수
날짜형	DATE	날짜(연도-월-일)
	DATETIME	날짜시간(연도-월-일 시:분:초)
	TIME	시간(시:분:초)
	YEAR	날짜(연도)

CREATE TABLE문의 예제

- '과목2' 테이블 생성

예제 6-1 '과목번호', '이름', '강의실', '개설학과', '시수' 열로 구성된 '과목2' 테이블을 생성하시오. '과목번호'를 기본키로 설정하시오.

```
CREATE TABLE 과목2
     ( 과목번호 CHAR(4) NOT NULL PRIMARY KEY,❶
       이름 VARCHAR(20) NOT NULL,❷
       강의실 CHAR(5) NOT NULL,
       개설학과 VARCHAR(20) NOT NULL,
       시수 INT NOT NULL ) ;❸
```

CREATE TABLE문의 예제

- '학생2' 테이블 생성

예제 6-2 '학번', '이름', '주소', '학년', '나이', '성별', '휴대폰번호', '소속학과' 열로 구성된 '학생2' 테이블을 생성하시오. '학번'을 기본키로, '휴대폰번호'를 대체키로 지정하고 '주소' 열에는 기본 값을 설정하시오.

```
CREATE TABLE 학생2
    ( 학번 CHAR(4) NOT NULL,
      이름 VARCHAR(20) NOT NULL,
      주소 VARCHAR(50) DEFAULT '미정',❹
      학년 INT NOT NULL,
      나이 INT NULL,
      성별 CHAR(1) NOT NULL,
      휴대폰번호 CHAR(13) NULL,
      소속학과 VARCHAR(20) NULL,
      PRIMARY KEY (학번),❺
      UNIQUE (휴대폰번호) ) ;❻
```

- DESC 명령문

```
DESC 학생2 ;
```

CREATE TABLE문의 예제

- '수강2' 테이블 생성

예제 6-3 '학번', '과목번호', '신청날짜', '중간성적', '기말성적', '평가학점' 열로 구성된 '수강2' 테이블을 생성하시오. '과목', '과목번호'의 조합을 기본키로, '학번'과 '과목번호'를 외래키로 설정하시오.

```
CREATE TABLE 수강2
     ( 학번 CHAR(6) NOT NULL,
       과목번호 CHAR(4) NOT NULL,
       신청날짜 DATE NOT NULL,
       중간성적 INT NULL DEFAULT 0,
       기말성적 INT NULL DEFAULT 0,
       평가학점 CHAR(1) NULL,
       PRIMARY KEY(학번, 과목번호),❽
       FOREIGN KEY(학번) REFERENCES 학생2(학번),❾
       FOREIGN KEY(과목번호) REFERENCES 과목2(과목번호) ) ;❿
```

무결성 제약 조건의 동작 확인

■ '과목2' 테이블의 행 입력

```
INSERT INTO 과목2(과목번호, 이름, 강의실, 개설학과)
VALUES ('c111', 'database', A-123, '산업공학') ;
-- ❶ CHAR(5) 유형 오류('강의실'열의 문자형 유형에 불일치함)
INSERT INTO 과목2(과목번호, 이름, 강의실, 개설학과, 시수)
VALUES ('c111', 'database', 'A-123', '산업공학') ;
-- ❷ NULL 제약 조건 오류('시수'열이 NULL을 허용하지 않음)
INSERT INTO 과목2(과목번호, 이름, 강의실, 개설학과, 시수)
VALUES ('c111', 'database', 'A-123', '산업공학', 3) ;
-- ❸ 정상 수행
```

```
INSERT INTO 학생2(학번, 이름, 학년, 나이, 성별, 휴대폰번호, 소속학과)
VALUES ('s111', '박태환', 4, NULL, '남', '010-1111-1111', '산업공학') ;
-- ❹ 정상 수행
INSERT INTO 학생2(학번, 이름, 학년, 나이, 성별, 휴대폰번호, 소속학과)
VALUES ('s222', '박태환', 2, NULL, '남', '010-1111-1111', '산업공학') ;
-- ❺ UNIQUE 제약 조건 오류('휴대폰번호'열 값이 중복됨)
INSERT INTO 학생2(학번, 이름, 학년, 나이, 성별, 휴대폰번호, 소속학과)
VALUES ('s222', '박태환', 2, NULL, '남', '010-2222-2222', '산업공학') ;
-- ❻ 정상 수행
```

무결성 제약 조건의 동작 확인

■ '수강2' 테이블의 행 입력

```
INSERT INTO 수강2(학번, 과목번호, 신청날짜)
VALUES ('s111', 'c111', '2019-12-31') ;
-- ❼ 정상 수행
INSERT INTO 수강2(학번, 과목번호, 신청날짜, 중간성적, 기말성적, 평가학점)
VALUES ('s111', 'c222', '2019-12-31', 93, 98, 'A') ;
-- ❽ 외래키 제약 조건 오류(입력 과목번호 값이 '과목2' 테이블에 존재하지 않음)
INSERT INTO 수강2(학번, 과목번호, 신청날짜, 중간성적, 기말성적, 평가학점)
VALUES ('s111', 'c111', '2019-12-31', 93, 98, 'A') ;
-- ❾ 기본키 제약 조건 오류(기본키 '학번'과 '과목번호'열의 조합이 중복 값이 존재함)
INSERT INTO 수강2(학번, 과목번호, 신청날짜, 중간성적, 기말성적, 평가학점)
VALUES ('s222', 'c111', '2019-12-31', 93, 98, 'A') ;
-- ❿ 정상 수행
```

1.2 테이블 수정 ALTER문

- ALTER TABLE문의 형식

```
ALTER TABLE 테이블_이름❶
  { [ADD|MODIFY] 열_이름 데이터_유형 [NULL|NOT NULL] [DEFAULT 기본_값] }❷
  | { ADD CONSTRAINT 제약조건_이름 제약조건_상세내용 }❸
  | { DROP COLUMN 열_이름 }❹
  | { DROP CONSTRAINT 제약조건_이름 } ;❺
```

예제 6-4 '학생2' 테이블에 새로운 '등록날짜' 열을 추가하시오.

```
ALTER TABLE 학생2
    ADD 등록날짜 DATETIME NOT NULL DEFAULT '2019-12-30' ;❻

SELECT * FROM 학생2 ;
```

예제 6-5 '학생2' 테이블의 '등록날짜' 열을 삭제하시오.

```
ALTER TABLE 학생2
    DROP COLUMN 등록날짜 ;❼

SELECT * FROM 학생2 ;
```

1.3 테이블 삭제 DROP문

- DROP TABLE문의 형식

```
DROP TABLE 테이블_이름 ;
```

예제 6-6 '과목2' 테이블을 삭제하시오.

```
DROP TABLE 과목2 ;      -- 삭제 불가
```

```
DROP  TABLE 수강2 ;     -- 참조(자식) 테이블 삭제
DROP TABLE 과목2 ;      -- 삭제 가능
```

```
SELECT * FROM 과목2 ;
DESC 과목2 ;
```

2. SQL 데이터 제어문

2.1 사용자 및 권한 관리

- 계정 생성: CREATE USER
- CREATE USER 명령문의 형식

```
CREATE USER 사용자_계정 IDENTIFIED BY '비밀번호' ;
```

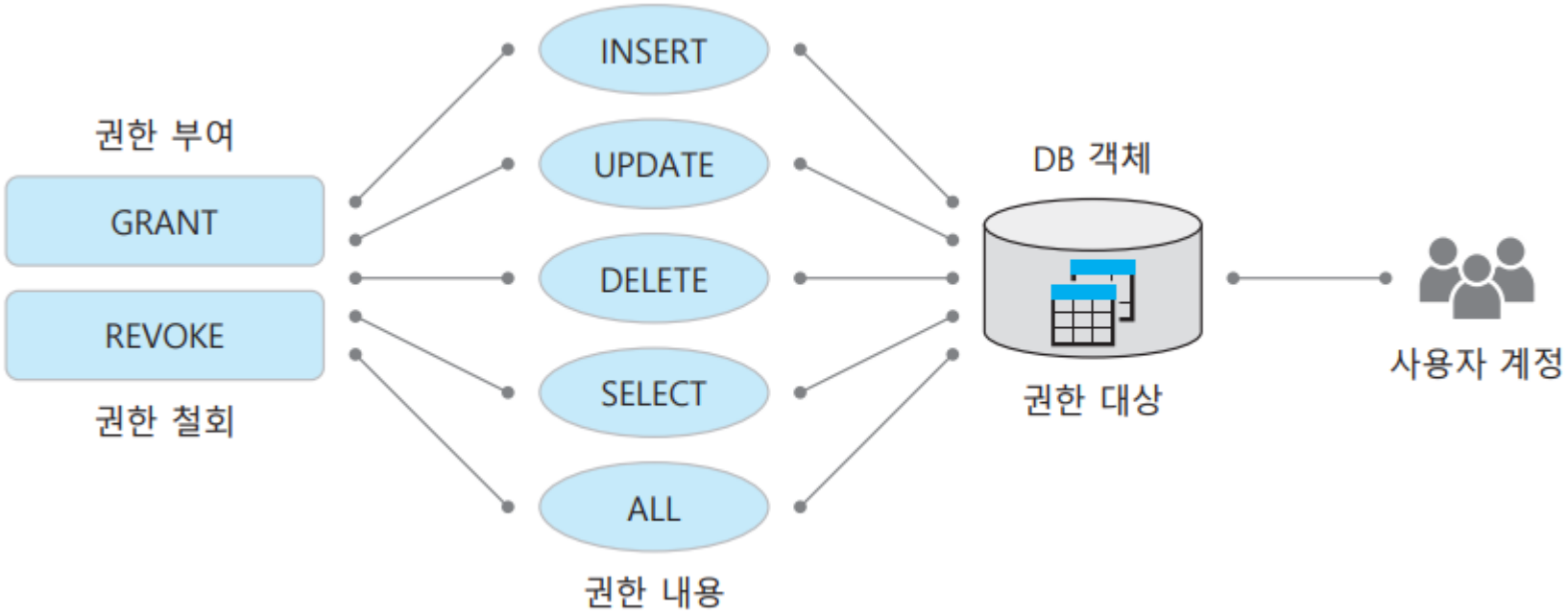
- CREATE USER 명령문의 예제

```
CREATE USER 'user1'@'127.1.1.1' IDENTIFIED BY '1111' ;❶
CREATE USER 'user2'@'localhost' IDENTIFIED BY '2222' ;❷
CREATE USER 'user3'@'192.182.10.2' IDENTIFIED BY '3333' ;❸
CREATE USER 'user4'@'%' IDENTIFIED BY '4444' ;❹
```

- 생성된 사용자 계정 정보 확인

```
SELECT host, user FROM mysql.user ;
```

사용자 및 권한 관리


권한 부여
GRANT
REVOKE
권한 철회
INSERT
UPDATE
DELETE
SELECT
ALL
권한 내용
DB 객체
권한 대상
사용자 계정

권한 부여: GRANT

- GRANT문의 형식

```
GRANT 권한_내용 ON 권한_대상 TO 사용자_계정 ;❺
GRANT 권한_내용 ON 권한_대상 TO 사용자_계정 WITH GRANT OPTION ;❻
```

- GRANT문의 예제

```
GRANT INSERT, UPDATE, DELETE ON univDB.* TO 'user1'@'127.1.1.1' ;❼
GRANT ALL ON *.* TO 'user4'@'%' WITH GRANT OPTION ;❽
GRANT SELECT ON univDB.학생 TO 'user2'@'localhost' ;❾
```

  - 사용자 계정의 권한 확인

```
SHOW GRANTS FOR 'user1'@'127.1.1.1' ;  -- user1 사용자의 권한 표시
```

```
SHOW GRANTS ;                          -- 현재 접속 사용자의 권한 표시
```

권한 철회: REVOKE

- REVOKE문의 형식

```
REVOKE 권한_내용 ON 권한_대상 FROM 사용자_계정 ;❿
```

- REVOKE문의 예제

```
REVOKE DELETE ON univDB.* FROM 'user1'@'127.1.1.1' ;⓫
```

계정 삭제: DROP USER

- DROP USER문의 형식

```
DROP USER 사용자_계정
```

- DROP USER문의 예제

```
DROP USER 'user1'@'127.1.1.1' ;
```

SQL 데이터 제어문

2.2 사용자 계정 생성

- 1) ‘manager’ 관리자계정 생성

```
CREATE USER 'manager'@'%' IDENTIFIED BY '1234' ;
```

- 2) ‘manager’ 관리자 계정에 권한을 부여

```
GRANT ALL ON *.* TO 'manager'@'%' WITH GRANT OPTION ;
```

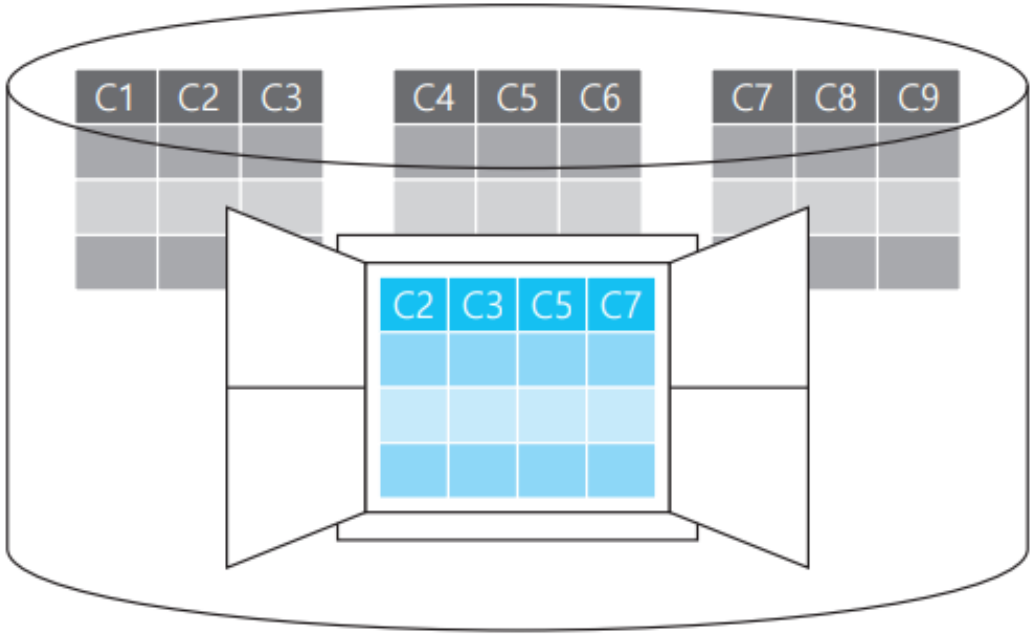
- 3) ‘manager’ 관리자 계정으로 접속
- 4) 접속한 사용자 아이디를 확인

```
SELECT user( ) ;        -- 현재 MySQL 사용자 표시
```

3. 뷰

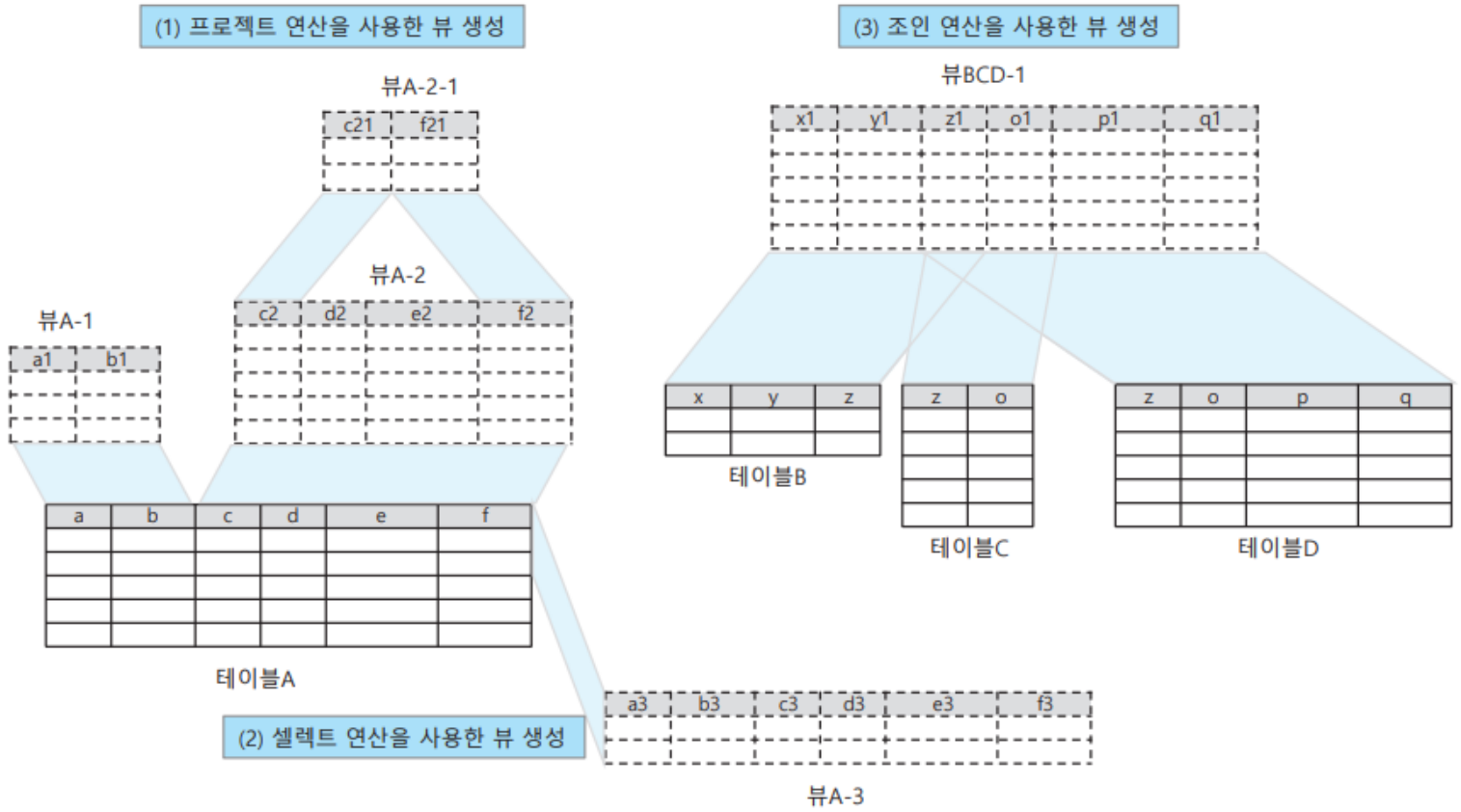
3.1 뷰 개념

- 데이터베이스를 바라보는 창문(window)



뷰의 정의

- 뷰(view)
  - 실제 데이터를 저장하지 않는 가상 테이블(virtual table)
  - 뷰에 대해 사용자가 질의를 요청할 때 비로소 DBMS는 뷰 정의를 참조하여 질의를 수행하고 그 결과를 사용자에게 반환
  - 주로 기반 테이블로부터 정의되지만 또 다른 뷰를 기반으로도 정의될 수도 있음



뷰의 장점

1) 편의성
: 복잡한 질의문 작성이 쉽고 간단해진다.

2) 보안성
: 데이터 보안 유지가 쉽다.

3) 재사용성
: 반복되는 질의문 작성에 효율적이다.

4) 독립성
: 스키마 변경에도 뷰 질의문은 변경할 필요가 없다.

3.2 뷰 생성

- CREATE VIEW 명령문의 형식

```
CREATE VIEW 뷰_이름 [ (열_리스트) ]❶
AS SELECT_검색문❷
```

예제 6-7 3학년 혹은 4학년 학생의 학생이름, 나이, 성, 학년으로 구성된 뷰를 'V1_고학년학생'이라는 이름으로 생성하시오.

```
CREATE VIEW V1_고학년학생(학생이름, 나이, 성, 학년)
AS SELECT 이름, 나이, 성별, 학년
    FROM 학생
    WHERE 학년 >= 3 AND 학년 <= 4
```

- 뷰의 생성 확인

```
SELECT *
FROM V1_고학년학생 ;      -- V1_고학년학생 뷰의 데이터 확인
```

뷰 생성

예제 6-8 각 과목별 과목번호, 강의실, 수강 인원수로 구성된 뷰를 'V2_과목수강현황'이라는 이름으로 생성하시오(과목별로 강의실이 동일하다고 가정한다).

```
CREATE VIEW V2_과목수강현황(과목번호, 강의실, 수강인원수)
AS SELECT 과목.과목번호, 강의실, COUNT(과목.과목번호)
    FROM 과목 JOIN 수강 ON 과목.과목번호 = 수강.과목번호
    GROUP BY 과목.과목번호 ;

SELECT * FROM V2_과목수강현황 ;
```

예제 6-9 'V1_고학년학생' 뷰를 기반으로 여학생만으로 구성된 뷰를 'V3_고학년여학생' 이름으로 생성하시오.

```
CREATE VIEW V3_고학년여학생
AS SELECT *
    FROM V1_고학년학생
    WHERE 성 = '여'

SELECT * FROM V3_고학년여학생 ;
```

3.3 뷰 활용– 뷰를 통한 검색

예제 6-10 생성된 뷰들을 통해 고학년여학생 정보만 검색하시오.

❶
```
SELECT *
  FROM V1_고학년학생
  WHERE 성 = '여' ;
```
⇒
❸
```
SELECT 이름 AS '학생이름', 나이, 성별 AS '성', 학년
  FROM 학생
  WHERE 학년 >= 3 AND 학년 <= 4 AND 성별 = '여' ;
```

❷
```
SELECT *
  FROM V3_고학년여학생
```
⇒
❸
```
SELECT 이름 AS '학생이름', 나이, 성별 AS '성', 학년
  FROM 학생
  WHERE 학년 >= 3 AND 학년 <= 4 AND 성별 = '여' ;
```

예제 6-11 'V2_과목수강현황' 뷰에서 수강생 인원이 가장 많은 과목과 가장 적은 과목에 대한 과목번호, 강의실, 수강인원수 정보를 검색하시오.

```
SELECT *
FROM V2_과목수강현황
WHERE 수강인원수 = ( SELECT MAX(수강인원수) FROM V2_과목수강현황 ) OR
      수강인원수 = ( SELECT MIN(수강인원수) FROM V2_과목수강현황 ) ;
```

뷰를 통한 데이터 변경

- 뷰 검색은 제한 없이 가능한 반면 뷰 변경은 특정한 경우로 한정
- 뷰 변경이 제한되는 이유
  - 뷰를 정의하는 'SELECT_검색문'의 조건이 다양하기 때문
  - 뷰가 참조하는 기반 테이블을 변경해야 하므로 내부 처리가 어려운 경우는 변경이 허용되지 않음

갱신이 불가능한 뷰

다음은 뷰를 통한 변경이 불가능한 경우로 입력, 수정, 삭제가 제한된다.

- 기반 테이블의 기본키를 포함하지 않은 뷰
- 집계 함수(또는 계산) 결과를 포함하는 뷰
- DISTINCT 키워드를 적용한 결과를 포함하는 뷰
- GROUP BY 절을 적용한 결과를 포함하는 뷰
- 다수의 기반 테이블의 조인을 통해 생성된 뷰
- 뷰에 포함되지 않은 기반 테이블의 열이 NOT NULL인 경우

3.4 뷰 삭제

- DROP VIEW 명령문의 형식

```
DROP VIEW 뷰_이름 ;
```

- DROP VIEW 적용 예

예제 6-12 'V1_고학년학생' 뷰를 삭제하시오.

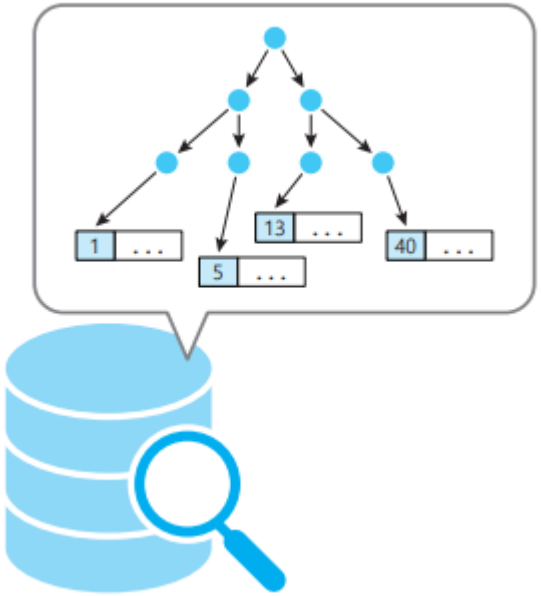
```
DROP VIEW V1_고학년학생 ;

SHOW TABLES ;
```

4. 인덱스

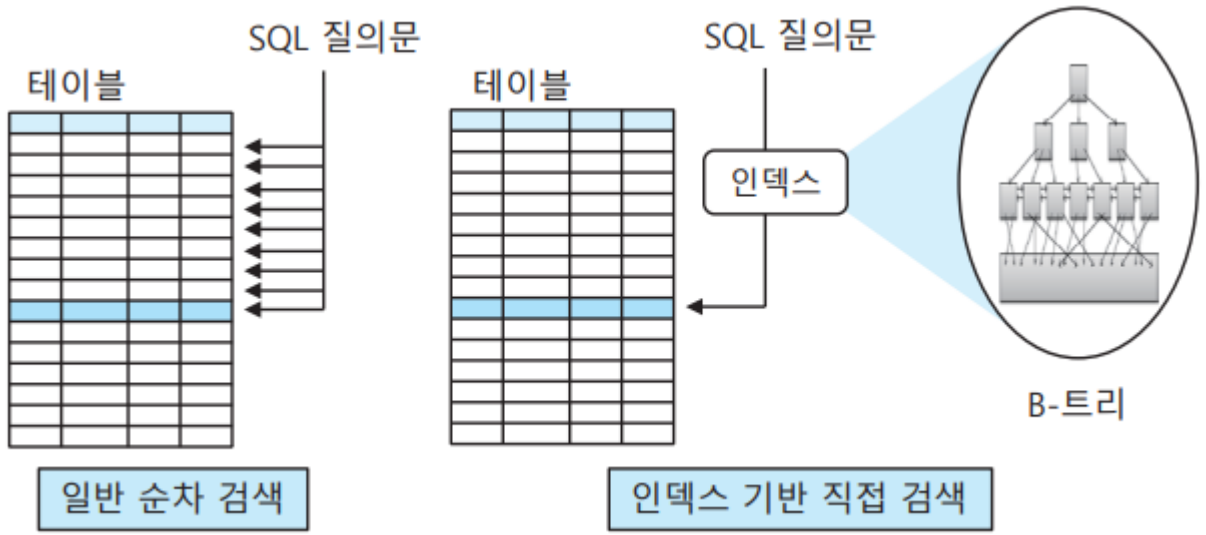
4.1 인덱스 개념

- 인덱스(index)
  - 테이블 안의 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터베이스 객체
  - 책 페이지를 목차나 색인을 통해 쉽게 찾는 것과 같은 원리



인덱스

- 인덱스의 필요성
  - SQL을 실행할 때, 디스크 접근 횟수를 줄여 검색 속도를 높이기 위함
    - 대부분 DBMS는 B-트리(Balanced tree) 구조의 인덱스를 지원
    - 장점: 루트 노드로부터 리프 노드까지의 탐색 길이가 같아서 모든 데이터에 대한 일정 수준의 검색 시간을 보장



4.2 인덱스 생성

- CREATE INDEX 명령문의 형식

```
CREATE ❶[REVERSE] ❷[UNIQUE] INDEX [인덱스_이름]
 ON 테이블_이름 ({열_이름 ❸[ASC|DESC]},+) ;
```

예제 6-13 '수강' 테이블의 '학번', '과목번호' 열을 대상으로 인덱스 'idx_수강'을 생성하시오.

```
CREATE INDEX idx_수강
ON 수강 (학번,과목번호) ;
```

```
SHOW INDEX FROM 수강 ;
```

예제 6-14 '과목' 테이블의 이름 열을 대상으로 유일한 값을 갖는 인덱스 'idx_과목'을 생성하시오.

```
CREATE UNIQUE INDEX idx_과목
  ON 과목 (이름 ASC) ;
```

4.3 인덱스 삭제

- 인덱스 삭제를 위한 DROP INDEX 명령문의 형식

```
DROP INDEX 인덱스_이름 ON 테이블_이름 ;
```

- DROP INDEX 적용 예

예제 6-15 인덱스 'idx_수강'을 삭제하고 '과목' 테이블의 인덱스 'idx_과목'을 삭제하시오.

```
DROP INDEX idx_수강 ON 수강 ;❶
ALTER TABLE 과목 DROP INDEX idx_과목 ;❷
```

인덱스 적용 고려사항

- 불필요한 인덱스는 오히려 성능을 떨어뜨리고 저장 공간만 낭비할 수 있음
  - 인덱스는 꼭 필요한 만큼만 효과적으로 생성해야 함
  - 【인덱스 생성이 필요한 경우】
    - 기본키와 외래키의 경우, 인덱스 생성이 바람직함. 대부분의 DBMS는 기본키에 대해서 자동으로 인덱스를 생성함
    - WHERE 절 조건식에 자주 사용되는 테이블 열의 경우, 인덱스 생성이 바람직함
    - 조인 조건식에 자주 사용되는 테이블 열도 인덱스 생성이 바람직함
    - 하나의 테이블에 3~5개 정도의 인덱스가 효과적임
    - 가변길이 문자형이나 실수형, 날짜형 열보다는 정수형, 고정길이 문자형 열에 인덱스를 생성하는 것이 바람직함
    - ORDER BY절이나 GROUP BY절에 자주 사용되는 열의 경우, 인덱스 생성을 고려할 수 있음
  - 【인덱스 생성이 불필요한 경우】
    - 갱신이 빈번한 테이블 열의 경우, 인덱스가 바람직하지 않음
    - 집계 함수, 내장 함수를 적용하여 열 값을 변형하는 경우, 인덱스가 효과적이지 않음
    - '성별' 같은 열처럼 도메인이 작아서 열의 선택도(selectivity)가 높을 경우, 인덱스가 바람직하지 않음
    - 범위 검색을 하는 경우, 인덱스가 바람직하지 않음
    - 테이블의 행 개수가 별로 없는 경우, 인덱스가 바람직하지 않음