- **5. SQL**
  - **1. SQL 개요 & 2. MySQL 서버 연결 및 SQL 에디터 구성**
    - **1. SQL 개요**
      - **SQL(Structured Query Language)**
        - **구조화된 질의 언어, 사람과 DBMS 간의 원활한 의사소통을 위한 언어**
        - **절차적 언어(procedural language)가 아니라 선언형 언어(declarative language)**
      - **유형**
        - **데이터 정의 언어 (DDL, Data Definition Language)**
          - **데이터베이스 내의 개체를 생성 및 삭제하고 그 구조를 조작하는 SQL문**
          - **- 테이블 스키마 정의, 테이블 삭제, 테이블 스키마 변경 등 - 제약조건 관련**
        - **데이터 조작 언어 (DML, Data Manipulation Language)**
          - **레코드 검색, 추가, 삭제 등의 실 데이터 조작과 관련된 SQL문**
          - **INSERT, UPDATE, DELETE, SELECT**
        - **데이터 제어 언어 (DCL, Data Control Language)**
          - **DBMS 동작 설정 및 DBMS 접근에 대한 사용자 권한을 관리하는 SQL 명령어 집합**
          - **GRANT, REVOKE, BEGIN 등**
    - **2. MySQL 서버 연결 및 SQL 에디터 구성 [123]**
      - **1. MySQL Connections**
        - **Connection Method**
        - **Hostname**
        - **Port**
        - **Username**
        - **Password**
      - **2. Workbench SQL 에디터[126]**
        - **네비게이터(Navigator) 패널**
          - **관리(MANAGEMENT)**
          - **인스턴스(INSTANCE)**
          - **스키마(SCHEMAS)**
        - **인포메이션(information) 패널**
        - **SQL 쿼리 패널**
        - **결과(Output 패널)**
  - **3. 데이터 정의 언어 [128]**
    - **개요**
      - **CREATE**
      - **ALTER**
      - **DROP**
    - **1. 스키마 정의**
      - **DDL문으로 생성 가능한 가장 상위 단계의 개체는 스키마(schema)**
      - **스키마(schema)**
        - **MySQL의 데이터베이스**
        - **테이블, 뷰, 인덱스 등의 데이터베이스 객체를 저장하기 위한 영역**
      - **스키마 생성 방법**
        - **CREATE SCHEMA 스키마이름**
        - **MySQL 워크벤치의 Forward Engineer**
      - **스키마 삭제 방법**
        - **DROP SCHEMA 스키마이름**
    - **2. 테이블 정의**
      - **(1) 테이블 생성**
        - **CREATE TABLE 테이블이름 (컬럼1 데이터타입1 [제약조건1], 컬럼2 데이터타입2 [제약조건2], : 컬럼n 데이터타입n [제약조건n] [PRIMARY KEY 컬럼] [UNIQUE 컬럼] [FOREIGN KEY 컬럼 REFERENCES 테이블(컬럼)])**
      - **(2) 데이터 타입(data type)**

- 데이터 타입(data type)
  - 컬럼에 저장되는 값의 종류
- 1) CHAR(n)과 VARCHAR(n)
  - 최대 n개로 구성된 문자열을 저장할 수 있는 데이터 타입
  - CHAR(n)
    - 선언된 컬럼의 길이가 고정되어 빈 공간은 공백문자로 채워짐
    - 별도의 실제 문자열 길이를 관리하지 않기 때문에 VARCHAR에 비해서 1~2 바이트가 절약된다
    - 삽입되는 데이터가 선언된 길이보다 작다면 공간 낭비가 발생한다
    - 모든 레코드에서 컬럼의 길이가 동일하기 때문에 데이터의 수정 및 검색의 속도가 빠르다
  - VARCHAR(n)
    - 각 컬럼 값의 길이에 맞춰 컬럼의 길이가 유지되기 때문에 공간을 효과적으로 사용할 수 있다.
    - CHAR에 비해 수정 및 검색 속도가 떨어진다.: 가변적인 길이를 위해 (문자열, 문자열 길이) 형태로 별도의 문자열 길이에 대한 정보를 관리해야 하기 때문에 데이터의 수정 발생 시 공간에 대한 별도의 연산 및 전체 레코드의 길이 수정 등 동반되는 연산이 있다.
  - CHAR과 VARCHAR 비교
- 2) INT와 FLOAT
  - INT (정수)
  - FLOAT (부동소수)
    - DOUBLE
- DECIMAL(n[,m])과 NUMERIC(n[,m])
  - DECIMAL과 NUMERIC은 동일한 데이터 타입으로 고정 소수점 표현을 지원
  - DECIMAL (n [,m])
    - DECIMAL(10,2)은 전체 10자리 중 8자리는 정수, 2자리는 소수점 이하의 수를 저장하기 위한 타입
  - NUMERIC(n[,m])
- DATETIME과 TIMESTAMP
  - 'CCYY-MM-DD hh:mm:ss' 형식의 날짜와 시간을 동시에 저장
  - YEAR, DATE, TIME 타입도 사용 가능
  - DATETIME
    - 8바이트 저장공간
    - '1000-01-01 00:00:00'부터 '9999-12-31 23:59:59'까지의 시간값 가질 수 있음
  - TIMESTAMP
    - 4바이트 저장공간
    - 표준시(UTC)를 기준으로 '1970-01-01 00:00:01'부터 '2038-01-09 03:14: 07'까지의 범위를 가짐
  - 차이점
    - TIMESTAMP의 날짜와 시간이 MySQL 서버의 타임존(timezone)에 따라 변경되는 반면, DATETIME은 항상 일정하다. 따라서 사용자와 MySQL 서버가 서로 다른 타임존에 위치할 경우 예상하지 않은 시간값이 사용될 수 있어 되도록이면 DATETIME 사용이 권장된다.
- ENUM과 SET
  - ENUM
    - 열거된 리스트에서 값을 선택해야 하는 문자열 집합
    - 성별 ENUM ('남자', '여자')
  - SET
    - 열거된 리스트 중 0개 또는 이상의 값의 집합을 할당할 수 있는 집합
- (3) 테이블 수정
  - ALTER TABLE 테이블이름 ALTER TABLE 테이블 이름 [ADD COLUMN 컬럼 데이터타입 [제약조건]] [DROP COLUMN 컬럼, …] [CHANGE COLUMN 수정전컬럼 수정후컬럼] [MODIFY COLUMN 컬럼 새로운_데이터타입]
  - ADD COLUMN
  - DROP COLUMN
  - CHANGE COLUMN
  - MODIFY COLUMN
- (4) 테이블 삭제

- **DROP TABLE** 테이블이름
  - 이름이 '테이블이름'인 테이블을 스키마에서 제거
  - 테이블이 삭제됨과 동시에 테이블 내의 모든 레코드 삭제, 테이블에 대한 정의가 시스템 카탈로그에서 제거

데이터 정의 언어 (2) [138]

- **3.** 제약 조건의 사용
  - 테이블 생성 시 정의된 제약조건은 새로운 레코드의 입력 혹은 기존 레코드의 수정 및 삭제가 되는 경우 **DBMS**가 사전에 정의한 제약조건을 준수하는지 검사한 후, 실행여부를 결정
  - 스키마 정의
    - **(1) PRIMARY KEY** 절
      - **PRIMARY KEY** 절은 기본키를 지정하는 제약조건으로 하나 이상의 컬럼을 기본키로 설정
      - 모든 레코드는 해당 컬럼에 대해 고유(unique)한 값을 가짐
      - **NULL** 값을 가질 수 없게 됨
    - **(2) NOT NULL** 절
      - **NN** 절
      - 해당 컬럼에 대해 반드시 어떤 값이 저장되어야 함
    - **(3) UNIQUE** 절
      - 어떤 레코드들도 해당 컬럼에 대해 다른 레코드와 동일한 값을 가질 수 없다.
    - **(4) AUTO_INCREMENT** 절
      - 사용자의 직접 입력 없이도 레코드가 추가될 때 자동적으로 기존 레코드의 값을 참고하여 순차적으로 해당 컬럼에 **1**을 증가시킨 값을 저장함
    - **(5) DEFAULT** 절
      - 해당 컬럼에 어떤 값이 입력되지 않으면 자동으로 지정된 값이 입력
    - **(6) FOREIGN KEY** 절
      - 다른 (테이블의) 컬럼을 참조하는 외래키를 정의
      - 참조를 하는 테이블의 컬럼과 참조가 되는 테이블의 컬럼이 필요함
        - 참조하는 컬럼의 값이 참조되는 컬럼에 존재하지 않는 값일 경우, 참조 무결성 제약조건에 의해 **DBMS** 실행을 거절한다
      - **FOREIGN KEY** (참조하는 컬럼이름) **REFERENCES** 참조되는 테이블이름 (참조되는 컬럼이름)
    - **(7) CHECK** 절
      - **CHECK**(조건) 또는 **CONSTRAINT id CHECK** (조건)
      - 컬럼값이 특정 조건을 준수하여 데이터의 무결성을 유지하기 위해 **CHECK** 절을 제공
      - **MySQL** 서버 **8.X** 이하의 버전은 **CHECK** 절을 지원하지 않음
      - 예제
        - **'CHECK**(나이**>18)'**
        - **CHECK** (이수구분 **IN('**전공필수**', '**일반선택**', '**교양**'))**

- **5.** 뷰의 사용 **[198]**
  - 개요
    - **뷰(view)**는 하나 이상의 원본 테이블로부터 유도되어 일반 테이블처럼 조작할 수 있는 가상 테이블**(virtual table)**이다.
    - 물리적으로 저장되는 원본 테이블과는 다르게 뷰는 물리적으로 저장되지 않는다.
    - 뷰의 장점
      - 데이터의 독립성 효과
        - 동적 기본 테이블 반영하되, 기본 테이블의 구조가 바뀌어도 뷰를 이용한 작업은 그 정의만 바뀌면 되므로 응용 프로그램 속에서 뷰를 이용한 검색 등의 작업은 영향을 받지 않는다.
      - 데이터의 보안 효과
        - 테이블에서 특정 사용자의 접근을 막아야 하는 컬럼이 있을 경우, 그 컬럼을 제외하고 뷰를 생성한 원본 테이블에는 접근 권한을 불허하고 뷰에만 접근을 허용하는 방법으로 보안 효과를 높일 수 있다.
      - 다양한 구조의 테이블 사용 효과
        - 다양한 사용자의 요구사항에 맞는 뷰를 제공함으로써 다양한 구조를 가진 테이블의 사용 효과를 낼 수 있다.
      - 작업의 단순화
        - 복잡한 질의를 뷰로 정의한 뒤 필요 시에 간단한 검색문 만으로 사용할 수 있기 때문에 작업 효율을 높일 수 있다.
      - 데이터의 무결성 효과

- 뷰의 생성 시 **WITH CHECK OPTION**을 이용한 경우, 뷰의 생성과 위배되는 갱신 작업은 실행이 거부되기 때문에 데이터 무결성 의 효과가 있다.

1. 뷰의 생성
   - **CREATE VIEW** 뷰이름 **AS** SELECT문 **[WITH CHECK OPTION]**
   - **MySQL Workbench**
2. 뷰의 수정
   - **CREATE VIEW** 뷰이름 **AS** SELECT문 **[WITH CHECK OPTION]**
3. 뷰의 삭제
   - **DROP VIEW** 이름
4. 뷰와 관련된 데이터 조작
   - 1) 뷰를 이용한 데이터 검색
     - **SELECT * FROM** 뷰 이름 **[WHERE** 조건**]**
   - 2) 뷰를 이용한 데이터 삽입
     - **INSERT**문이 뷰가 아닌 기본 테이블에서 실행되어 삽입이 이루어짐
     - **INSERT INTO** 컴퓨터과학과_학생 (학생번호, 학생이름, 성별, 생년월일, 나이, 전화번호, 학과이름, 이수학점) **VALUES ('201934-901129', '홍길동', '남', '1993-03-29', 27, '010-0001-0001', '컴퓨터과학과', 0);**
       - 기본 테이블 2개에 나눠 입력됨.
       - 테이블의 제약 조건으로 인해 실패할 수 있음
     - **WITH CHECK OPTION**
       - 레코드의 삽입 및 수정 후 해당 레코드가 뷰를 통한 검색 결과에 포함될 수 없는 형태의 질의문은 처리되는 것을 방지함
       - **WITH CHECK OPTION**은 뷰(view)를 통해 **INSERT·UPDATE** 작업을 할 때, 그 변경 결과가 뷰 정의의 **WHERE** 절 조건을 항상 만족하도록 강제하는 옵션
       - **CREATE VIEW** 생활과학과_교수 **AS (** SELECT * FROM 교수 WHERE 소속학과 = '생활과학과') **WITH CHECK OPTION; SELECT * FROM** 생활과학과_교수;
         - 생활과학과 교수가 아니면 오류 발생
         - 테이블 무결성을 위해 효율적 제약조건이 됨