5. ADVANCED SQL

Accessing SQL from Programming Language

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27863/

PROGRAMMING LANGUAGE 사용 이유

- SQL

  장점 : 사용하기 용이한 declarative(선언형) query language

  단점 : 모든 질의를 다 표현 X – 무엇을 수행하고 어떤 데이터를 필요로 하는지만 기술

  – 데이터를 어떻게 가져오는지 기술 X

  : 비선언적 동작 지원 X – 보고서 출력, 사용자와의 상호작용, 질의 결과 GUI로 전달 등

- SQL로 다 표현할 수 없는 질의를 작성하기 위해 프로그래밍 언어 사용함

DB 서버 접근법 (1)

1. Dynamic SQL (동적 SQL)

- 프로그래밍 언어 직접 사용 → 직접 JAVA API 호출 O, 컴파일러 필요 X

  (프로그램에서 함수, 메소드 사용 → DB 서버 접속, 통신 가능)

- 과정

  : 런타임에 SQL 쿼리를 문자열로 만들고 전달

  → 결과를 프로그램 변수로 한 번에 1개 튜플씩 가져옴

- SQL 구문을 런타임에 해석 → 오류 : 런타임에 발견

- JDBC, ODBC (응용프로그래밍 인터페이스 예시)

DB 서버 접근법 (2)

2. Embedded SQL (내장 SQL)

- 응용 프로그램 내에 포함되어 함께 실행되도록 호스트프로그램에 삽입된 SQL
- 호스트 언어로 작성된 프로그램 : DB에 저장된 데이터에 접근/갱신 위해 Embedded SQL의 문법 사용하기도 함
- 컴파일 전 전처리기(preprocessor)를 통해 우선 처리되어야 함
- 전처리 이후, 호스트 언어의 컴파일러로 컴파일
- SQL 구문의 오류 (ex. 데이터타입 오류)는 컴파일 시 발견됨

JDBC와 ODBC

1. JDBC : Dynamic SQL의 API

   - Java 언어를 위한 응용프로그래밍 인터페이스
   - DB 서버와 상호작용하는 프로그램을 위한 응용프로그램 인터페이스
   - 응용프로그램과 데이터베이스를 연결해주는 역할

2. ODBC : C, C++, C#, Visual Basic과 같은 언어를 사용할 때 사용되는 API

JDBC

- SQL을 C, Java와 같은 범용 프로그래밍 언어와 함께 사용할 시 고려할 점

  1) 각자 다른 방식으로 데이터를 다룸

     SQL : 기본적인 데이터 형태 = 릴레이션

     릴레이션에 대한 동작 수행 → 결과 릴레이션 반환

     프로그래밍 언어 : 변수를 사용한 작업

- 응용프로그램에서 JDBC API를 호출하는 경우

  1) DB 서버와 연결할 때

  2) DB 서버와 SQL 명령을 보낼 때

  3) 쿼리 결과 튜플을 하나씩 결과 프로그램의 변수에 저장하기 위해 가져올 때

JDBC

: SQL을 제공하는 DB 시스템과 데이터를 주고받기 위한 Java API

- 인터페이스로서 응용프로그램과 DB를 연결해주는 역할
- DBMS의 종류와 관계없이 SQL 문 수행, 결과처리 O
- 데이터를 질의, 갱신, 쿼리결과 검색을 위한 다양한 기능 제공
- DB와의 통신 방식

    1) DB에 접속 (커넥션 열기)

    2) "statement" 객체 생성 – 쿼리 문을 실행하기 위해

    3) Statement 객체를 사용하여 쿼리 실행 → 쿼리결과 가져오기

    4) 예외 처리 (오류 처리)