데이터베이스응용 2강

하 석 재
CEO, 2HCUBE
sjha72@gmail.com

교과목정보

- 과목명 : 데이터베이스응용(온라인, 가반/나반)
- 강좌번호: 2150028101/2150028102
- 학점 : 3학점
- 비고 : 공학인증
- 선수과목: 데이터베이스

교과목 개요 및 목표

교과목 개요

- 관계형 데이터베이스의 활용 데이터베이스 설계와 실제 서비스 모델링 SQL, 상용 DBMS 도구 실습, 웹을 활용한 실습 및 과제

공학인증역량

- 이론 및 알고리즘의 검증 능력 문제해결을 위한 도구 활용 능력 설계 능력

출석규정

- 출석 1/3 이상 결석할 경우 자동적으로 F부여(교육부 가이드라인)
- 1학기동안 개인적인 사유로 결석 1회 인정
- 스마트캠퍼스출석 활용

평가기준

- 출석 : 10%
- 중간고사 : 35% -> 과제대체나 기말고사로 통합가능(별도공지)
- 기말고사 : 35%
- 과제 : 20%

수업일정

Week	Contents	
1	과정소개, 오픈소스 및 환경구성	
2	관계형 데이터베이스	
3	SQL - 기본 및 고급 질의	
4	JDBC	
5	인덱스와 트랜잭션(원자성)	
6	트랜잭션 고급(격리성/일관성/MVCC)	
7	MySQL Replication(복제) - DB이중화	
8	중간고사	
9	MySQL HA(고가용성)	
10	파티셔닝/백업	
11	샤딩	
12	MyBatis, Hibernate, JPA	
13	반정형데이터처리(JSON, NoSQL)	
14	비정형데이터처리	
15	기말고사	

관계형 데이터베이스

RDBMS의 주요개념

- 테이블
- 모델링
- 외래키
- 정규화
- 조인/서브쿼리
- 인덱스
- 트랜잭션

RDBMS의 특징(테이블 기반, 정규화)

- 데이터베이스는
  - 데이터를 **CRUD(Create, Retrieve, Update, Delete)**하는 시스템이다.
- 관계형 DBMS는 **테이블기반(Table based)**으로 데이터 저장하는 시스템이다.
  - 테이블, 컬럼, 레코드, ...
- 그러면 전체를 큰 테이블 하나로 저장하면 되는 것인가?
  - 데이터의 중복을 **정규화(Normalization)**를 통해 중복성제거
  - 보통 1,2,3정규형까지 적용한 것을 정규화(Normalize)했다고 말함
  - 정규화를 하면 보통 테이블이 큰 테이블에서 여러 개의 작은 테이블로 분리

RDBMS의 특징(조인)

- 테이블이 여러 개로 나눠진 상태에서 원하는 데이터를 찾으려면
  테이블을 합쳐서(조인:JOIN) 필요한 데이터 검색
  - 내부조인(INNER)/외부조인(LEFT/RIGHT/FULL JOIN)
    - 조인은 테이블단위 연산(Operation)임
  - 조인과정에서 테이블 간의 공통컬럼이 있어야 함
    - 외래키(FK:Foreign Key)
  - 서브쿼리(Subquery)로 검색하는 방법도 가능하지만
    기본방식은 조인을 기반으로 처리한다

RDBMS의 특징(인덱스)

- RDBMS는 **쓰기(write)보다는 읽기(read)를 빠르게 할 수 있도록 최적화**된 기술임
  - 이를 위해 보통 **인덱스(index)**라고 불리는 자료구조를 테이블의 컬럼에 추가해서 검색속도를 향상시킴
  - 보통 **B+트리** 형태의 인덱스를 사용함
    - 해시, 전문검색, R-트리 등의 인덱스도 지원함

- DBA는 인덱스를 잘 적용하고 적용하는 인덱스종류도 잘 알아야 함

RDBMS의 특징(인덱스)

- **읽기보다 쓰기가 빈번한 시스템**이라면?
  - 쓰기에 최적화된 **NoSQL(Not Only SQL)**
  - 키(Key)/밸류(Value) 구조저장방식
    - JSON이 대표적인 기술임
  - 빠른 쓰기를 위해 컬럼기반 저장 테이블을 지원함

  - 요즘에는 NoSQL의 주요특징인 JSON처리,
    컬럼기반 저장 테이블 등을 RDBMS에서
    지원하는 경우가 늘어나고 있슴

RDBMS의 특징(트랜잭션)

- All or Nothing
  - 커밋(Commit) or 롤백(Rollback) / savepoint
- 여러 줄의 SQL쿼리를 실행할 때 장애가 발생했을 경우 쿼리 전체를 취소하거나 확정
- 보통 RDBMS에서는 기본으로 지원하는 경우가 많음

- 트랜잭션의 ACID 특성
  - **원자성(Atomicity)** / 일관성(Consistency)
  - 격리성(Isolation) / 내구성(Durability)

정형 / 반정형 / 비정형데이터

- **정형데이터(Structured Data)**
  - 스키마에 따른 데이터저장, 자유도가 낮음
  - RDBMS, XML 형태의 데이터를 말함
- **반정형데이터(Semi-Structured Data)**
  - 명확한 스키마가 없슴, 어느 정도의 자유도 허용
    - 데이터마다 완전히 동일한 구조를 따를 필요는 없슴
  - 주로 JSON형태의 데이터를 말함
- **비정형데이터(Unstructured Data)**
  - 스키마가 없슴
  - 비디오, 오디오 등의 멀티미디어 데이터 / 일반 텍스트나 웹자료 등을 말함

RDBMS의 특징(모델링)

- **개념**모델링 - **논리**모델링 - **물리**모델링
- 논리모델링(E-R Modeling)
  - 테이블을 좀 더 세분해서 1차(기본) 테이블인 **엔티티(entity)**와 유도된 테이블인 **릴레이션십(Relationship)**으로 구분
  - 피터 첸의 표기법(ERD:E-R Diagram)
    - 엔티티는 사각형 / 릴레이션십은 다이아몬드로 표현
  - 까마귀 발(Crow's Foot) 표기법
    - 테이블 종류를 구별하지 않음
    - 대응수(Cardinality) 표현이 쉬움
      - 1:1, 1:N, M:N 학생(1) - 수강(2) - 교수(1)

RDBMS의 특징(모델링)

- **개념**모델링 - **논리**모델링 - **물리**모델링
- 물리모델링
  - 컬럼(속성)명 영어로 변경(ERWin Glossary)
  - 개념모델링을 진행하면 테이블의 수가 많아짐
  - 엔티티 수 + 릴레이션십 수
  - 1:1과 1:N 대응수를 가지는 경우는 중간 릴레이션십을 삭제한 후
    - 1:1 -> 엔티티에 직접 컬럼을 추가
    - 1:N -> 외래키로 엔티티간 직접 연결

- 다대다(M:N) 이상의 관계를 모델링 하려면 GraphDB을 사용

정규화(Normalization)

- 제1정규형
  - 컬럼값이 원자값을 가지지 않으면 레코드 분리

- 학생테이블
  - 학생 / 나이 / 과목
  - Adam/15/생물,수학
  - Alex /14/ 수학
  - Stuart/17/ 수학

- 학생테이블
  - 학생 / 나이 / 과목
    - Adam/15/생물
    - Adam/15/수학
    - Alex /14/ 수학
    - Stuart/17/ 수학

정규화(Normalization)

- 제2정규형
  - 키 컬럼과 나머지 일반 컬럼이 직접적인 연관관계(직접 종속)만 가지도록 테이블 구성

- 학생테이블
  - 학생 / 나이
    - Adam/15
    - Alex /14
    - Stuart/17

- 수강테이블
  - 학생 / 과목
    - Adam/생물
    - Adam/수학
    - Alex / 수학
    - Stuart/ 수학

정규화(Normalization)

- 제3정규형
  - 일반 컬럼끼리 간접 연관관계(간접 종속)를 가지지 않도록 테이블 분리

- 학생테이블
  - 학생 / ... / 우편번호 / 시,도 / 구 / 동

- 학생테이블
  - 학생/.../우편번호

- 주소 테이블
  - 우편번호 / 시 / 구 / 동

정규화(Normalization)

- BCNF 정규형(Boyce-Codd Normal Form)
  - 제 3정규형의 강화버전
  - 모든 결정자가 후보키 집합에 속한 경우

- 학생테이블
  - 학생/교수/과목

- 학생테이블
  - 학생/과목

- 교수테이블
  - 교수/과목

외래키(Foreign Key)

- 어떤 테이블의 **기본키**가 다른 테이블에 속해 있는 컬럼
  - FK가 PK인 경우
    - 식별관계(Identifying Relationship)
  - FK가 일반 컬럼인 경우
    - 비식별관계(Non-identifying Relationship)

- 두 테이블 간의 공통 컬럼
  - 테이블 조인시 기준이 되는 컬럼

외래키(Foreign Key)

- 학생테이블
  - id/ 이름 / 나이
    - 1/Adam/15
    - 2/Alex /14
    - 3/Stuart/17
- 수강테이블
  - id / 과목 / 학생테이블_id
    - 1 / 생물 / 1
    - 2/ 수학 / 1
    - 3/ 수학 / 3
    - 4/ 수학 / 4

외래키(Foreign Key)

- **참조무결성 제약조건(Referential Integrity)**
  - 외래키로 연결되어 있는 테이블의 경우에는 외래키로 연결 테이블을 검색하면 데이터가 존재해야 함
  - 삽입/삭제에 순서가 있어야 함
    - 해당 순서를 어기면 에러발생
      - 원 레코드를 삭제하려면 참조하는 레코드를 먼저 삭제해야 가능
      - 외래키를 가지는 레코드를 추가하려면 연결테이블에 레코드를 먼저 추가해야 함

외래키(Foreign Key)

- 학생테이블
  - id/학생 / 나이
  - 1 /Adam/15
  - 3 /Alex /14
  - 4 /Stuart/17
  - **5 /Kim / 16**
- 수강테이블
  - id/ 과목 / 학생_id
  - 1 / 생물 / 1
  - 2/ 수학 / 1
  - 3/ 수학 / 3
  - 4/ 수학 / 4
  - **5 / 수학/ 5**

- 삽입/삭제시 순서가 있어야 함

조인(Join)

- 2개 이상의 테이블을 합쳐 큰 테이블을 만듬
  - 외래키 이용 / Nested Join 방식 사용(기본)
- 내부조인(Inner Join) / 외부조인(Outer Join)
  - 널 값의 허용 여부에 따라 나뉨
- **Inner Join**
  - 드라이빙(기준) 테이블/드리븐 테이블의 널값을 허용하지 않음
- **Outer Join**
  - Left : 드라이빙테이블의 레코드가 누락되면 안 됨
  - Right : 드리븐테이블의 레코드가 누락되면 안 됨
  - Full Join : 양쪽 모두 누락되면 안 됨
    - MySQL은 지원 안 됨

조인(Join)

- 직원테이블
  - id/직원명/나이/부서_id
- 부서테이블
  - id/부서명

- 직원-부서 테이블(조인결과)
  - 직원_id/직원_직원명/직원_나이/직원_부서_id/부서_id/부서_부서명

조인(Join)

- Inner 조인
  - 직원중 부서가 없는 직원, 직원이 없는 부서는 누락시킴
- Left 조인(driving테이블의 레코드가 누락되면 안됨)
  - 직원 중 부서가 없는 직원은 누락시키지 않음
- Right 조인(driven테이블에 레코드가 누락되면 안됨)
  - 부서중 직원이 없는 부서는 누락시키지 않음
- Full 조인
  - 두 경우 모두 누락시키지 않음

도커기반의 MySQL설치

MySQL 설치

- MySQL 컨테이너 실행

  $ docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=**ssu** --name=mysql1 mysql:5.7

  $ docker ps -a (up인지 확인)

- MySQL 컨테이너 접속

  $ docker exec -it mysql1 bash

  # mysql -uroot -p

MySQL 기본사용법

```
mysql> show databases;

mysql> use mysql;

mysql> show tables;

mysql> desc user;

mysql> desc user\G

mysql> show create table user\G

mysql> select * from user\G

mysql> quit
```

감사합니다