# 데이터베이스 연동

빠른 탐색과 검색을 위해 조직된 데이터의 집합체, 여러 명의 사용자나 응용프로그램이 공유하는 데이터들.

## 데이터베이스(Database)

데이터베이스의 목적 - 데이터를 저장했다가 필요할 때에 사용하는 것

데이터베이스의 기능

1. 데이터의 추가/조회/변경/삭제
2. 데이터의 무결성(integrity) 유지
3. 트랜잭션(Transaction) 관리
4. 데이터의 백업 및 복원
5. 데이터 보안

## DBMS(Database Management System)

데이터베이스를 관리하는 시스템

## DB/DBMS의 특징

### 데이터의 무결성 (Integrity)

데이터베이스 안의 데이터는 오류가 없어야 함. 제약 조건(Constrain)이라는 특성을 가짐

### 데이터의 독립성

데이터베이스 크기 변경하거나 데이터 파일의 저장소 변경할 때 기존에 작성된 응용프로그램은 전혀 영향을 받지 않아야 함.

### 보안

데이터베이스 안의 데이터에 데이터를 소유한 사람이나 데이터에 접근이 허가된 사람만 접근할 수 있어야 하며, 접근할 때도 사용자의 계정에 따라서 다른 권한 가짐.

## DBMS의 종류

계층형 DBMS : 처음으로 나온 DBMS 개념 - 1960년대에 시작. 트리형.

망형 DBMS : 계층형 개선. 그물 구조.

관계형 DBMS : 여러 테이블과 테이블 간의 관계(Relation)으로 구성.

## 테이블 & 레코드

### 테이블

데이터가 저장되는 가상의 장소. 테이블은 1개 이상의 칼럼으로 구성.

각 칼럼은 타입을 가지며, 제약(값의 길이, 가질 수 있는 값 등)을 갖는다.

이런 테이블의 구성을 스키마(schema)라고 함

### 레코드

칼럼의 모음을 레코드(record)라고 표현. 하나의 테이블은 여러 개의 레코드로 구성

## 기본키와 인덱스

### 기본키(Primary Key)

레코드를 구별하기 위해 사용되는 것. 레코드가 서로 다른 값을 갖는 칼럼. 기본키를 이용해서 빠른 검색 가능

### 인덱스

지정한 칼럼에 맞춰 데이터의 정렬 순서를 미리 계산. 주요키도 인덱스의 종류. 중복된 값을 가질 수도 있음

## 데이터베이스 프로그래밍의 일반적인 순서

(1) 데이터베이스를 사용하기 위해 먼저 데이터베이스에 연결

(2) 데이터베이스에 CRUD(Create, Read, Update, Delete)의 작업을 수행

(3) 결과를 전송

(4) 데이터베이스 연결을 종료



## SQL(Structured Query Language)

데이터 조회, 삭제 등의 데이터베이스 작업을 수행할 때 사용되는 언어

### 자료형

|  |  |
| --- | --- |
| **SQL 타입** | **설명** |
| CHAR | 확정 길이의 문자열을 저장. 표준의 경우 255 글자까지만 저장 |
| VARCHAR | 가변 길이의 문자열을 저장. 표준의 경우 255 글자까지만 저장 |
| LONG VARCHAR | 긴 가변 길이의 문자열을 저장 |
| NUMERIC | 숫자를 저장 |
| DECIMAL | 십진수를 저장 |
| INTEGER | 정수를 저장 |
| TIMESTAMP | 날짜 및 시간을 저장 |
| TIME | 시간을 저장 |
| DATE | 날짜를 저장 |
| CLOB | 대량의 문자열 데이터를 저장 |
| BLOB | 대량의 바이너리 데이터를 저장 |

DBMS에 따라 조금씩 다를 수 있음.

### 종류

DDL (Data Description Language) : 테이블 생성과 같이 데이터를 정의할 대 사용되는 SQL

DML (Data Manipulation Language) : 데이터 삽입, 조회, 삭제와 같이 데이터를 다루기 위해 사용되는 SQL

DCL (Data Control Language) : 데이터베이스 관리용. 권한 설정, 서버 on/off 등

## 한글 설정

현재 문자셋 확인 : show variables like ‘c%’;

my.ini 파일 변경 (변경 후 MySQL 재시작)

[client]

default-character-set = utf8

=====================================================

[mysqld]

character-set-client-handshake = FALSE

init\_connect="SET collation\_connection = utf8\_general\_ci"

init\_connect="SET NAMES utf8"

character-set-server = utf8

=====================================================

[mysql]

default-character-set = utf8

=====================================================

[mysqldump] (이거는… ?)

default-character-set = utf8

========================================

## SQL 실습

#주석의 종류

# -> 한줄 주석

-- -> 한줄 주석

/\*

범위 주석

\*/

#스키마 삭제 명령

drop database if exists sqlDB;

#drop은 스키마, 테이블, 뷰 등 삭제 시 사용

#스키마 생성 명령

create database if not exists sqlDB

default charset=utf8 collate=utf8\_bin;

#if exists : 존재하면

#if not exists : 존재하지 않으면

#스키마 사용

use sqlDB;

#테이블 삭제

drop table if exists userTbl;

#테이블 생성

create table if not exists userTbl (

userID char(8) not null primary key,

name char(10) not null,

birthYear int not null,

addr char(3) not null,

mobile1 char(3),

mobile2 char(8),

height int,

mDate date

);

drop table if exists buyTbl;

create table if not exists buyTbl (

num int auto\_increment not null primary key,

userID char(8) not null,

prodName char(4) not null,

groupName char(4),

price int not null,

amount int not null,

foreign key (userID) references userTbl(userID)

);

#데이터 삽입 구문 insert into

INSERT INTO userTbl VALUES ('LSG', '이승기', 1987, '서울', null, null, 182, '2008-8-8');

INSERT INTO userTbl VALUES ('KBS', '김범수', 1979, '경남', '011', '22222222', 173, '2012-4-4');

INSERT INTO userTbl VALUES ('KKH', '김경호', 1971, '전남', '019', '33333333', 177, '2007-7-7');

INSERT INTO userTbl VALUES ('JYP', '조용필', 1950, '경기', '011', '44444444', 166, '2009-4-4');

INSERT INTO userTbl VALUES ('SSK', '성시경', 1979, '서울', NULL, NULL, 186, '2013-12-12');

INSERT INTO userTbl VALUES ('LJB', '임재범', 1963, '서울', '016', '66666666', 182, '2009-9-9');

INSERT INTO userTbl VALUES ('YJS', '윤종신', 1960, '경남', NULL, NULL, 170, '2005-5-5');

INSERT INTO userTbl VALUES ('EJW', '은지원', 1972, '경북', '011', '88888888', 174, '2014-3-3');

INSERT INTO userTbl VALUES ('JKW', '조관우', 1965, '경기', '016', '99999999', 172, '2010-10-10');

INSERT INTO userTbl VALUES ('BBK', '바비킴', 1973, '서울', '010', '00000000', 176, '2013-5-5');

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('KBS', '운동화', '의류', 30, 2);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('KBS', '노트북', '전자', 1000, 1);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('JYP', '모니터', '전자', 200, 1);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('BBK', '모니터', '전자', 200, 5);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('KBS', '청바지', '의류', 50, 3);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('BBK', '메모리', '전자', 80, 10);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('SSK', '책', '서적', 15, 5);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('EJW', '책', '서적', 15, 2);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('EJW', '청바지', '의류', 50, 1);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('BBK', '운동화', '의류', 30, 2);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('EJW', '책', '서적', 15, 1);

INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('BBK', '운동화', '의류', 30, 2);

# select ... from

# 원하는 데이터를 가져와 달라고 요청하는 구문

select \* from userTbl;

/\* select 구문

select ....

from 테이블

where 조건

\*/

select \* from userTbl where name='김경호';

select userID, name from userTbl where birthYear >= 1970;

select \* from userTbl where birthYear >= 1970 and height >= 180;

select \* from userTbl where birthYear >= 1970 or height >= 180;

#between 구문은 and와 결합하여 사용

select \* from userTbl where height >= 172 and height <= 177;

select \* from userTbl where height between 172 and 177;

#in 구문

select \* from userTbl where addr='경남' or addr='전남' or addr='경북';

select \* from userTbl where addr in ('경남', '전남', '경북');

#특정 문자열이 들어간 컬럼 검색 : like, (\_, %)

# \_ : 자리수 고정

# % : 자리수 상관 없이

select \* from userTbl where name like '김%';

select \* from userTbl where name like '신';

select \* from userTbl where name like '\_종\_';

# 서브쿼리 : 쿼리문 내부에 쿼리문이 들어감

# 서브쿼리가 들어가는 구문 : from, where

select \* from userTbl where height >

(select height from userTbl where name='김경호');

select \* from userTbl where height >

(select height from userTbl where addr='경남');

#any, in, all

select \* from userTbl where height =

any (select height from userTbl where addr='경남');

#any : 서브쿼리의 여러 개의 결과 중 한가지만 만족해도 결과 검색

#any = in

select \* from userTbl where height in

(select height from userTbl where addr='경남');

#all : 서브쿼리의 결과들과 모두 일치

#SQL은 DBMS에 따라 쓸 수 있는 구문과 쓸 수 없는 구문이 있다.

#검색 결과의 순서 지정 : order by

select name, mDate from userTbl order by name DESC;

#오름차순 : ASC, 내림차순 : DESC

select name, mdate, height from userTbl order by height DESC, name ASC;

select addr from userTbl;

#중복 제거 : distinct

select distinct addr from userTbl;

#출력 갯수 제한 : limit

#예) 게시판 출력물이 한 페이지당 5

#limit 시작위치, 갯수

select \* from userTbl limit 3;

#기존 테이블에서 새 테이블 만들기

create table buyTbl2 (select \* from buyTbl);

select \* from buyTbl2;

create table buyTbl3 (select userID, prodName from buyTbl);

select \* from buyTbl3;

#테이블 삭제 : drop

drop table buyTbl2;

drop table buyTbl3;

#출력 결과의 묶음 : group by

#집계 함수 : avg(), min(), max(), count()

# count(distinct),

# sedev() - 표준편차

# var\_samp() - 분산

# sum()

#누가 얼마나 구매했는지 구매 수량을 구하기 위한 sql 쿼리문

select userId, sum(amount) from buyTbl group by userId;

#별칭 지정 : as

select userId as '사용자 아이디',

sum(amount) as '총 구매 갯수' from buyTbl group by userId;

#회원별 구매액 구하기

select userId as '아이디', sum(amount \* price) as '총 구매액' from buyTbl group by userId;

#행의 갯수(레코드 갯수) 구하기 count()

select count(\*) from userTbl;

#평균 구매 수량

select avg(amount) as '평균 구매 갯수' from buyTbl;

#min(), max()

select min(height) from userTbl;

select max(height) from userTbl;

select name, height from userTbl where height = (select min(height) from userTbl) or

height = (select max(height) from userTbl);

select count(mobile1) as '휴대폰 있는 사람' from userTbl;

select \* from userTbl;

#1000 이상 구매한 사람의 id 구하기

select userId as '사용자', sum(amount \* price) as '총구매액' from buyTbl

where sum(amount \* price) > 1000 group by userId;

#group by에 조건을 주는 구문 having

select userId as '사용자', sum(amount \* price) as '총구매액' from buyTbl group by userId

having sum(amount \* price) > 1000;

#with rollup : 총합 또는 중간 합계를 구하는 구문

select userId as '사용자', sum(amount \* price) as '총구매액' from buyTbl group by userId with rollup;

select num, groupName, sum(amount \* price) as '비용' from buyTbl group by groupName, num with rollup;

#키, 도메인 설정 없이 생성

create table testTbl (id int, userName char(3), age int);

#데이터 삽입 구문 : insert into

insert into testTbl values (1, '홍길동', 25);

insert into testTbl (id, userName) values (2, '전우치');

insert into testTbl (userName, id, age) values ('지민후', 3, 28);

select \* from testTbl;

#AUTO\_INCREMENT 속성 : 자동 증가(1씩)

#반드시 int 타입, 대체적으로 기본키에 적용

#unique 속성에 적용 가능

create table testTbl2

(id int auto\_increment primary key,

userName char(3),

age int

);

insert into testTbl2 values (1, '홍길동', 25);

insert into testTbl2 values (null, '지민후', 28);

insert into testTbl2 values (null, '이우진', 26);

insert into testTbl2 values (null, '이주미', 25);

select \* from testTbl2;

#자동 증가 숫자의 변경

# -> alter table로 변경

alter table testTbl2 auto\_increment=100;

insert into testTbl2 values (null, '유효림', 21);

#201822001 : 전산과

#201823001 : 전자과

#데이터의 갱신(수정) : update

#제품 가격 상승, 50% 상승

update buyTbl set price = price \* 1.5;

select \* from buyTbl;

#update, delete 시 주의할 점

#where 절로 조건을 주지 않으면 모든 레코드 변경

update testTbl2 set age=age+1;

update testTbl2 set id=5, age=21 where userName='유효림';

#레코드 삭제 구문 : delete

delete from testTbl2 -- 모든 데이터 삭제

where id=1;

#변수는 접속이 유지되는 동안 사용 가능

#선언 및 초기화 : set @변수명 = 값;

set @cnt=0; -- 변수 사용

select @cnt;

update testTbl2 set id=@cnt:=@cnt+1;

select \* from testTbl2;

/\*

auto\_increment 초기화 및 모든 컬럼 적용

alter table 테이블명 auto\_increment=1;

set @변수명=0;

update 테이블명

set 컬럼명=@변수명:=@변수명+1;

\*/

#적체 삭제

delete from testTbl; -- 테이블 내용 삭제

drop table testTbl; -- 테이블 삭제

#테이블의 데이터 전체 삭제 : truncate

#truncate와 delete의 차이

-- truncate는 DDL이기 때문에 트랜젝션 없이 처리되어 빠르다.

#참고1. DDL은 트랜젝션 없음(되돌리기 불가)

#참고2. DML은 트랜젝션 있음(되돌리기 가능)

#참고3. 트랜젝션은 로그처리를 해야하기 때문에 처리시간이 느림

truncate table testTbl2;

#join(조인)

-- 두 개 이상의 테이블 결합

#관계형 데이터베이스는 여러개의 테이블로 분리하여 데이터를 저장

#테이블간의 관계를 활용하여 원하는 정보 검색

#외래키를 사용하여 다른 테이블을 참조

select userId as '사용자', sum(amount \* price) as '총구매액'

from buyTbl group by userId having sum(amount \* price) > 1000;

select name from userTbl where userId='KBS';

#10개를 구매한 고객의 이름을 구하는 쿼리

#서브쿼리를 활용한 형태 -> inner join으로 변경

select name from userTbl

where userId = ( select userId from buyTbl where amount = 10 );

-- 실습용 테이블 생성 및 데이터 입력

#학생 정보 테이블

create table stdTbl

( stdname varchar(10) not null primary key,

addr char(4) not null

);

#클럽 정보 테이블

create table clubTbl

( clubname varchar(10) not null primary key,

roomno char(4) not null

);

#학생 정보, 클럽 정보

create table stdclubTbl

( num int auto\_increment primary key,

stdname varchar(10) not null,

clubname varchar(10) not null,

foreign key(stdname) references stdtbl(stdname),

foreign key(clubname) references clubtbl(clubname)

);

#여러 데이터 묶음 insert

insert into stdtbl values ('김범수', '경남'), ('성시경', '서울'), ('조용필', '경기'), ('은지원', '경북'), ('바비킴', '서울');

insert into clubtbl values ('수영', '101호'), ('바둑', '102호'), ('축구', '103호'), ('봉사', '104호');

insert into stdclubtbl values

(null, '김범수', '바둑'), (null, '김범수', '축구'), (null, '조용필', '축구'),

(null, '은지원', '축구'), (null, '은지원', '봉사'), (null, '바비킴', '봉사');

select \* from stdtbl;

select \* from clubtbl;

select \* from stdclubtbl;

#조인 종류

-- 1. inner join

# 기본 조인 형태

# select 컬럼들

# from 첫번째테이블

# inner join 두번째테이블

# on 조인조건

# where 검색조건

select s.stdname, s.addr, sc.clubname from stdtbl s

inner join stdclubtbl sc

on s.stdname = sc.stdname

where sc.clubname='축구';

select \* from usertbl u

join buytbl b

on u.userId = b.userId;

#outer join (외부 조인)

#left, right, full(mysql에서는 안씀)

select \* from usertbl u

left outer join buytbl b

on u.userId = b.userId; #usertbl의 정보는 다 나와야 한다!

#제품을 구매하지 않은 회원 검색

select u.userId, u.name, b.prodName, u.addr

from usertbl u

left outer join buytbl b

on u.userId = b.userId

where b.prodName is not null;

#동아리에 가입하지 않는 학생

select \*

from stdtbl s

left outer join stdclubtbl sc

on s.stdname = sc.stdname

left outer join clubtbl c

on sc.clubname = c.clubname;

#동아리를 기준으로 가입 안한 학생(회원이 없는 동아리 출력)

select c.clubname, c.roomno, s.stdname, s.addr

from stdtbl s

left join stdclubtbl sc

on s.stdname = sc.stdname

right join clubtbl c

on sc.clubname = c.clubname

order by c.clubname;

#cross join

select \* from buytbl

cross join usertbl;

#두테이블의 곱(cartesian product)

#회원 수 10, 구매수 12

# -> 레코드 수는 10 \* 12

#self join : 자체 조인

create table emptbl

(emp char(3), manager char(3),

emptel varchar(8)

);

insert into emptbl values

('나사장', null, '0000'),

('김재무', '나사장', '2222'),

('김부장', '김재무', '2222-1'),

('이부장', '김재무', '2222-2'),

('우대리', '이부장', '2222-2-1'),

('지사원', '이부장', '2222-2-2'),

('이영업', '나사장', '1111'),

('한과장', '이영업', '1111-1'),

('최정보', '나사장', '3333'),

('윤차장', '최정보', '3333-1'),

('이주임', '윤차장', '3333-1-1');

select \* from emptbl;

select a.emp as '부하직원', b.emp as '직속상관', b.emptel as '직속상관연락처'

from emptbl a

inner join emptbl b

on a.manager = b.emp;

## JDBC 프로그래밍 구조



## JDBC (드라이버) 설치

MySQL 설치 시 다운받은 Connector/j jar 파일을 톰캣 lib에 저장

## JDBC 프로그램의 전형적인 실행 순서

① JDBC 드라이버로딩

② 데이터베이스 커넥션 구함

③ 쿼리 실행을 위한 Statement 객체 생성

④ 쿼리 실행

⑤ 쿼리 실행 결과 사용

⑥ Statement 종료

⑦ 데이터베이스 커넥션 종료

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  **<%@ page import=*"java.sql.\*"*%>** //DB 연동을 위한 패키지 포함  <html><head><title>회원 목록</title></head>  <body>  <h3>회원 정보</h3>  <table>  <tr>  <th width=*"100"*>이름</th>  <th width=*"100"*>아이디</th>  <th width=*"100"*>주소</th>  </tr>  <%  String dburl = "jdbc:mysql://127.0.0.1:3306/sqldb";  String dbuser = "root";  String dbpass = "1234";    String query = "select \* from usertbl order by userId";    Connection conn = **null**;  Statement stmt = **null**;  ResultSet rs = **null**;    **try**{  Class.forName("com.mysql.jdbc.Driver");    conn = DriverManager.getConnection(dburl, dbuser, dbpass);  stmt = conn.createStatement();  rs = stmt.executeQuery(query);    **while**(rs.next()){  %>  <tr>  <td><%=rs.getString("name") %></td>  <td><%=rs.getString("userId") %></td>  <td><%=rs.getString("addr") %></td>  </tr>  <%  }  }**catch**(SQLException e){  e.printStackTrace();  }**finally**{  **try**{  **if**(rs != **null**) rs.close();  **if**(stmt != **null**) stmt.close();  **if**(conn != **null**) conn.close();  }**catch**(SQLException se){  se.printStackTrace();  }  }  %>  </table>  </body>  </html> |