DB 중간고사 시험대비 summary (시험 들어가기전에 꼭 볼것)

**JAVA코드**

**jdbc.java**

public static void JDBCexample(String userid, String passwd)

{

try(

Connection conn = DriverManager.getConnection(

"jdbc:oracle:thin:@db.yale.edu:1521:unidb",

userid, passwd);

Statement stmt = conn.createStatement();

){

try{

stmt.executeUpdate(

"insert into instructor values('77987','Kim','Physics',98000

}

catch (SQLExceptionUpdate sqle) {

System.out.println("Could not insert tuple. " + sqle);

}

ResultSet rset = stmt.executeQuery(

"select dept\_name, avg(salary)" +

" from instructor " +

" group by dept\_name");

while (rset.next()) {

System.out.println(rset.getString("dept\_name") + " " + rset.getFloat(2));

}

}

catch (Exception sqle)

{

System.out.println("Exception : " + sqle);

}

}

**stmt.executeQuery()를 사용하여 질의 수행 >> resultset객체 rset에 저장된 결과 튜플 집합으로부터 검색을 해서 한 번에 한 튜플씩 받아 올 수 있다.**

PreparedStatement pStmt = conn.prepareStatement(

"insert into instructor values(?,?,?,?)");

pStmt.setString(1, "88877");

pStmt.setString(2, "Perry");

pStmt.setString(3, "Finance");

pStmt.setInt(4, 125000);

pStmt.executeUpdate();

pStmt.setString(1, "88878");

pStmt.executeUpdate();

**결과 집합의 모든 열의 타입과 이름을 출력하기 위해 JDBC사용**

ResultSetMetaData rsmd = rs.getMetaData();

for(int i=1; i<=rsmd.getColumnCount(); i++) {

System.out.println(rsmd.getColumnName(i));

System.out.println(rsmd.getColumnTypeName(i));

}

**DatabaseMetaData를 이용하여 JDBC에서 열에 대한 정보 찾기**

DatabaseMetaData dbmd = conn.getMetaData();

ResultSet rs = dbmd.getColumns(null, "univdb", "department", "%");

while( rs.next()) {

System.out.println(rs.getString("COLUMN\_NAME"),

rs.getString("TYPE\_NAME");

}

**Python을 이용한 DB접근**

import psycopg2

def PythonDatabaseExample(userid, passwd)

try:

conn = psycopg2.connect( host="db.yale.edu", port=5432,

dbname="univdb", user=userid, password=passwd)

cur = conn.cursor()

try:

cur.execute("insert into instructor values(%**s**, %s, %s, %s)", ("77987", "Kim", "Physics", 98000))

conn.commit();

except Exception as sqle:

print("Could not insert tuple. ", sqle)

conn.rollback()

cur.execute( ("select dept\_name, avg (salary) "

"from instructor group by dept\_name"))

for dept in cur:

print dept[0], dept[1]

except Exception as sqle:

print("Exception : ", sqle)

**ODBC코드의 예**

void ODBCexample()

{

RETCODE error;

HENV env;

HDBC conn;

SQLAllocEnv(&env);

SQLAllocConnect(env, &conn);

SQLConnect(conn, "db.yale.edu", SQL\_NTS, "avi", SQL\_NTS, "avipasswd", SQL\_NTS);

{

char deptname[80];

float salary;

int lenOut1, lenOut2;

HSTMt stmt;

char \* sqlquery = "select dept\_name, sum (salary)

from instructor

group by dept\_name";

SQLAllocStmt(conn, &stmt);

error = SQLExecDirect(stmt, sqlquery, SQL\_NTS);

if ( error == SQL\_SUCCESS) {

SQLBindCol(stmt, 1, SQL\_C\_CHAR, deptname, 80, &lenOut1);

SQLBindCol(stmt, 2, SQL\_C\_FLOAT, &salary, 0, &lenOut2);

while (SQLFetch(stmt) == SQL\_SUCCESS) {

printf(" %s %g\n", deptname, salary);

}

}

SQLFreeStmt(stmt, SQL\_DROP);

}

SQLDisconnect(conn);

SQLFreeConnect(conn);

SQLFreeEnv(env);

}

**SQL로 정의된 함수(dept\_count())**

**create function** dept\_count (dept\_name **varchar**(20))

**returns integer  
 begin**

**declare** d\_count **integer;**

**select count** (\* ) **into** d\_count

**from** instructor  
 **where** instructor.dept\_name = dept\_name

**return** d\_count;

**end**

**SQL의 테이블 함수**

create function instructor\_of(dept\_name varchar(20))

returns table(

ID varchar (5),

name varchar (20),

dept\_name varchar (20),

salary numeric (8,2))

return table

(select ID, name, dept\_name, salary

from instructor

where istructor.dept\_name = instructor\_of.dept\_name);

**과목 분반에 학생을 등록하기 위한 프로시저**

create function registerStudent(

in s\_id varchar(5),

in s\_courseid varchar (8),

in s\_secid varchar (8),

in s\_semester varchar (6),

in s\_year numeric (4,0),

out errorMsg varchar(100)

returns integer

begin

declare currEnrol int;

select count(\*) into currEnrol

from takes

where course\_id = s\_courseid and sec\_id = s\_secid

and semester = s\_semester and year = s\_year;

declare limit int;

select capacity into limit

from classroom natural join section

where course\_id = s\_courseid and sec\_id = s\_secid

and semester = s\_semester and year = s\_year;

if(currEnrol < limit)

begin

insert into takes values

(s\_id, s\_courseid, s\_secid, s\_semester, s\_year, null);

returns(0);

end

set errorMsg = 'Enrollment limit reached for course' || s\_courseid || ' section ' || s\_secid;

return(-1)

end;

**참조 무결성을 유지하기 위한 트리거의 사용(timeslot\_check1) p.189**

create trigger timeslot\_check1 after insert on section referencing new row as nrow  
 for each row  
 when (nrow.time\_slot\_id not in (

select time\_slot\_id

from time\_slot)) /\* time\_slot에 나타나진 않은 time\_slot\_id \*/ begin

rollback end;

create trigger timeslot\_check2 after delete on timeslot referencing old row as orow  
 for each row  
 when (orow.time\_slot\_id not in (

select time\_slot\_id  
 from time\_slot)  
 /\* last tuple for time slot id deleted from time slot \*/

and orow.time\_slot\_id in (  
 select time\_slot\_id  
 from section)) /\* and time\_slot\_id still referenced from section\*/

begin rollback

end;

**Credits\_earned값을 유지하기 위한 트리거(격발기)의 사용 p.191**

create trigger credits\_earned after update of takes on (grade)

referencing new row as nrow  
referencing old row as orow  
for each row

when nrow.grade <> ’F’ and nrow.grade is not null

and (orow.grade = ’F’ or orow.grade is null)

begin atomic  
 update student  
 set tot\_cred= tot\_cred +

(select credits  
 from course  
 where course.course\_id= nrow.course\_id)

where student.id = nrow.id;

end;

**삽입된 값을 변화시키기 위해서 set을 사용하는 예제(for each row대신 for each statement사용) p.191하단**

**create trigger setnull before update of takes**

**referencing new row as nrow  
for each row  
when (nrow.grade = ‘ ‘)**

**begin atomic**

**set nrow.grade = null;  
end;**

**품목을 재주문하기 위한 트리거의 예**

create trigger reorder after update of level on inventory

referencing old row as orow, new row as nrow

for each row

when nrow.level <= (select level

from minlevel

where minlevel.item = orow.item)

and orow.level > (select level

from minlevel

where minlevel.item = orow.item)

begin atomic

insert into orders

(select item, amount

from reorder

where reorder.item = orow.item);

end;

**과목의 모든 선행 과목 찾기(재귀이용) p.197**

create function findAllPrereqs(cid varchar(8))

returns table (course\_id varchar(8))

begin

create temporary table c\_prereq (course\_id varchar(8));

create temporary table new\_c\_prereq (course\_id varchar(8));

create temporary table temp (course\_id varchar(8));

insert into new\_c\_prereq

select prereq\_id

from prereq

where course\_id = cid;

repeat

insert into c\_prereq

select course\_id

from new\_c\_prereq;

insert into temp

(select prereq.prereq\_id

from new\_c\_prereq, prereq

where new\_c\_prereq.course\_id = prereq.course\_id

)

except (

select course\_id

from c\_prereq

);

delete from new\_c\_prereq;

insert into new\_c\_prereq

select \*

from temp;

delete from temp;

until not exists (select \* from new\_c\_prereq)

end repeat;

return table c\_prereq;

end

**SQL 재귀 질의**

with recursive rec\_prereq(course\_id, prereq\_id) as (

select course\_id, prereq\_id

from prereq

union

select rec\_prereq.course\_id, prereq.prereq\_id

from rec\_prereq, prereq

where rec\_prereq.prereq\_id = prereq.course\_id

)

select \*

from rec\_prereq;

**이론 기출**

**UTF-8** : Unicode Transformed Form 8bits

**DBMS** : Data base management system

**SQL** : Structured Query Language

**DDL :** Data Definition Language

**DML :** Data Manipulation Language

**DBMS:** DataBase Management System

**SQL의 문자, 정수, 실수 값의 범위 유형 7가지를 쓰시오.**

char(n), varchar(n), int, smallint, numeric(p,d), real, double precision, float(n)

**데이터 추상화 3단계 수준을 쓰시오.**

물리적, 논리적, 뷰

**관계형 자료모형(Data Model)는 몇 년도 누가 만들었나?**

1970년 Edgar Frank codd

**릴레이션과 표(table)의 대응하는 각 구성요소를 모두 쓰시오.**

속성, 투플

**관계대수의 기본 연산자를 모두 쓰시오.**

선택, 추출, 자연조인, 합집합, 곱집합, 차집합.

**추가연산자를 아는 대로 모두 쓰시오.**

조인, 카티션 곱, rename, 문자열

**외부 조인은 어떤 경우에 사용하며 어떤 종류가 있는가?**

관련된 행이 있는 경우, INNER, LEFT OUTER, FULL OUTER

"교수는 개설강좌를 가르친다", "학생은 개설강좌를 수강한다"의 ERD(개체관계도)는?

**SQL 기출**

**2009년 가을과(and) 2010년 봄에 개설한 교과목을 찾아라(자동중복제거됨) (중복제거 안하고싶으면 intersect를 intersect all로 바꿈)**

(Select course\_id

from section

Where sem = ‘Fall’ and year = 2009)

intersect

(Select course\_id

from section

Where sem = ‘Spring’ and year = 2010);

**2017년 가을 학기와 2018년 봄 학기에 둘다 개설된 교과목을 구하라(distinct, in 접속사이용)**

(Select course\_id

From section

Where semester = ‘Fall’ and year= 2017) and

Course\_id in(select course\_id

From section

Where semester =’Spring’ and year =2018);\

**2017년 가을학기와 2018년 봄 학기 둘 다 있는 교과목을 구해라 (exists구문이용)**

Select course\_id

From section as S

Where semester = ‘Fall’ and year = 2017 and

Exists (select \*

From section as T

Where semester = ‘Spring’ and year = 2018 and

S.course\_id = T.course\_id);

**총학점(totCred) 값이 null인 학생의 학번과 이름을찾아라**

Select name,ID

From student

Where tot\_cred is null;

**생물학과에서 제공하는 모든 과목을 수강하는 (모든) 학생을 구하라**

Select S.ID, S.name

From student as S

Where not exists((select course\_id

From course

Where dept\_name =’Biology’)

Except

(select T.course\_id

From takes as T

Where S.ID=T.ID));

**2017년에 많아야 한 번 개설된 모든 과목을 구하라**

Select T.course\_id

From course as T

Where unique (select R.course\_id

From section as R

Where T.course\_id = R.course\_id and

R.year = 2017);

**총학점이 12점 이하인 학생을 삭하는 질의어를 쓰시오.**

Delete from student

Where tot\_cred < 13;

**Amit에게 student릴레이션에 대한 select 특권을 주고 amit이 특권을 다른 사용자에게 양도할 수 있도록 하라**

Grant select on student to amit with grant option;

**학생 자신의 정보를 student에 삽입하는 질의어를 쓰시오.**

(tot\_cred가 디폴트 0으로 설정되어있어서 학생 삽입시 빠뜨리고 삽입가능)

Insert into student(ID, name, dept\_name)

Values(‘12789’, ‘Newman’,’Comp. Sci.’);

**과목 릴레이 션을 생성하는 SQL문을 작성하시오.**

Create table course (

   course\_id     varchar(8)   primary key,

   title          varchar(50),

   dept\_name    varchar(20),

   credits numeric(2,0),

   foreign key(dept\_name) references department));

**학생 관계의 모든 투플을 찾아라**

Select count(\*) (distinct 사용 금지)

From course;

**3학점인 Comp. Sci. Dept에 개설된 과목의 제목을 찾으시오.**

select   title

from    course

where   dept\_name = 'Comp. Sci. 'and credits = 3;

**강사 가운데 연봉이 가장 큰 금액을 구하시오.**

select max (salary)

from instructor;

**강사의 이름에 성이 朴氏(박씨)인 강사의 ID를 출력하시오.**

select   name

from    instructor

where   name like '박%‘;

**강사가 가르친 과목들의 과목 ID를 가지고 담당 강사 이름을 내림차 순으로 나열하시오.**

**1) 카티전곱+선택:**

select   name, course\_id

from    instructor, teaches

where   instructor.ID = teaches.ID

order by name desc;

**2) 자연 조인:**

from  instructor naturel join teaches;

**2010년 봄과 2010년 가을에 동시에 개설된 과목 ID를 찾으시오.(표현 법이 다른 두 개 SQL문으로)**

1) (select  course\_id

from  section

where  semester = 'spring' and year = 2010)

intersect

(select  course\_id

from  section

where semester = 'Fall' and year = 2010)

2) select course\_id

from  section

where  semester = 'string' and year = 2010 and

        course\_id in (select course\_id

                     from section

        where semester = 'Fall' and year = 2010);

**연봉의 합이 $50000가 넘는 강사의 이름을 내림차순으로 출력하시오.**

select  name

from   instructor

where  salary > 50000

order  by name desc;

**과목이름 Database에 등록한, 학생의 수룰 구하시오.**

select  count(distinct ID)

from takes

where course = 'Data base';

**과목 ID 'CS-111', 제목 'Web Services', 학점 3을 과목 릴레이션에 삽입하시오.**

insert  into course

   values ('CS-111' , 'web services' , 3);