

2023 년 데이터베이스 ISQL 과제

숙명여자대학교 공과대학 인공지능공학부

Issue: 2023.05.08.

교수 박 영 호

yhpark@sm.ac.kr

1. Subject

관계형 데이터베이스 관리 시스템에서 **Interactive SQL** (10 점 만점)

2. Due Date

2023 년 5 월 15 일(월요일) 22 시 정시 마감, **snowboard**

3. Spec

- 1) 사용 시스템 : DBMS 선택에 제한 없이 어떠한 제품이든 자신의 PC 에서 설치
(MS SQL, MySQL, Oracle 등 상용 데이터베이스 자유 선택, 단 SQLite 는 안됨.)
- 2) 실습 중에 다룰 내용 : isql (interactive SQL)의 사용

4. 과제 제출 방법

질의를 작성하여 “그 질의 번호(그룹-번호)와 수행 결과 화면을 캡춰”하여 **Word** 나 **HWP** 에 순차적으로 담아 제출 문서를 작성한다. 그리고, **SnowBoard** 에 제출한다.

5. 제출시 작성 내용

문서 작성 시, 과제를 시작하는 첫 페이지 상단에 다음을 기술한다.

만약 본인 스스로 20 문제를 풀었다면, “본인이 풀고 제출하는 답안 수: **20/24**”

라고 기술한다.

6. 채점 기준

- (1) 본인이 스스로 푼 문제라고 인정되는 경우 점수를 부여한다.
- (2) 본인의 어떤 가정하에서 풀었다면, 그 가정을 제시한 경우, 그 가정에 따라 문제를 채점한다. 이때 그 문제를 풀기 전에 본인의 가정 사항을 붉은 색 글자로 기술해 두어야 한다.
- (3) 제출 점수 0.4 점 + (각 질의 결과 당 0.4 * 24 문항 = 9.6) = 10 점.

<과제 내용 설명>

본 실습 과정 동안 “suppliers-parts-projects”라는 데이터베이스를 사용한다. 이 데이터베이스는 예제와 숙제에 사용되는 sample database 이므로 실습 전에 데이터베이스의 스키마를 숙지하여야 한다. 이후, 문제로 주어진 6 개의 질의 그룹과 그에 속한 24 문항을 반드시 순서대로 처리해야 한다. 순서대로 지문의 지시에 따라 수행해야 한다.

다음은 sample database 에 대한 설명이다.

㉞ 어떤 상용데이터베이스는 특수 문자인 ‘#’을 속성명에서 사용할 수 없는 경우가 있다. 이때는 S# 대신 S_shap 과 같은 방법으로 속성명을 변경하여 스키마를 정의하여야 한다.

Sample Database 이름: **suppliers-parts-projects (공급자-부품-프로젝트)**

* **Table “Supplier”** : Supplier 에 대한 테이블이다. Supplier 에 대한 튜플은 supplier number (S#), supplier name (SNAME), rating or status value (STATUS), location (CITY)들로 이루어져 있다. 여기서 supplier number 는 supplier 마다 서로 다른 값을 갖는다. Supplier name 은 서로 다른 supplier 가 같은 name 을 가질 수도 있다. 또한 예제를 간단히 하기 위하여, 각 supplier 는 단지 한 도시에만 위치할 수 있다고 가정한다. STATUS 는 NULL 값을 가질 수 있다.

* **Table “Part”** : Part(정확히 말하면 part 의 종류)에 대한 테이블이다. Part 에 대한 튜플은 part number (P#), part name (PNAME), color (COLOR), weight (WEIGHT), 해당 part 가 저장되어 있는 location (CITY)들로 이루어져 있다. 여기서 각 part 는 part number 를 가지고 유일하게 식별이 가능하다. 또한 예제를 간단히 하기 위하여, 각 part 는 오직 한가지 color 만 가지며, 단 한군데의 city 에만 저장되어 있다고 가정한다.

* **Table “Project”** : Project 에 대한 테이블이다. Project 에 대한 튜플은 project number (J#), project name (JNAME), location (CITY)들로 이루어져 있다. 여기서 각 project 는 project number 를 가지고 유일하게 식별이 가능하다.

* **Table “SPJ”** : Supplier, part, project 사이의 관계를 나타내는 테이블이다. 즉, 각 튜플은 특정 supplier 가 어떤 project 에 무슨 part 를 얼마만큼 공급하고 있는지를 나타내기 위한, supplier number (S#), part number (P#), project number (J#), supplied quantity (Q#)로 구성되어 있다. 여기서 supplier number, part number, project number 의 쌍은 각 튜플을 유일하게 식별할 수 있다. 또한 QTY 는 NULL 값을 가질 수 있다.

Supplier	(S#, SNAME, STATUS, CITY)
Part	(P#, PNAME, COLOR, WEIGHT, CITY)
Project	(J#, JNAME, CITY)
SPJ	(S#, P#, J#, QTY)

Sample data values for **suppliers-parts-projects** database

Supplier		S#	SNAME	STATUS	CITY
		S1	Smith	20	London
		S2	Jones	10	Paris
		S3	Blake	30	Paris
		S4	Clark	20	London
		S5	Adams	30	Athens
Part	P#	PNAME	COLOR	WEIGHT	CITY
	P1	Nut	Red	12	London
	P2	Bolt	Green	17	Paris
	P3	Screw	Blue	17	Rome
	P4	Screw	Red	14	London
	P5	Cam	Blue	12	Paris
	P6	Cog	Red	19	London
Project	J#	JNAME	CITY		
	J1	Sorter	Paris		
	J2	Punch	Rome		
	J3	Reader	Athens		
	J4	Console	Athens		
	J5	Collator	London		
	J6	Terminal	Oslo		
	J7	Tape	London		
SPJ	S#	P#	J#	QTY	
	S1	P1	J1	200	
	S1	P1	J4	700	
	S2	P3	J1	400	
	S2	P3	J2	200	
	S2	P3	J3	200	
	S2	P3	J4	500	
	S2	P3	J5	600	
	S2	P3	J6	400	
	S2	P3	J7	800	
	S2	P5	J2	100	
	S3	P3	J1	200	
	S3	P4	J2	500	
	S4	P6	J3	300	
	S4	P6	J7	300	
	S5	P2	J2	200	
	S5	P2	J4	100	
	S5	P5	J5	500	
	S5	P5	J7	100	
	S5	P6	J2	200	
	S5	P1	J4	100	
	S5	P3	J4	200	
	S5	P4	J4	800	
	S5	P5	J4	400	
	S5	P6	J4	500	

질의 그룹 # 1: 4 questions

<단순 질의 문제>

1. Get supplier numbers for suppliers who supply project J1, in supplier number order.
2. Get all shipments where the quantity is in the range 300 to 750 inclusive.

<Joins 을 사용하는 질의 문제>

3. Get part numbers for parts supplied by a supplier in London to a project in London.
4. Get all pairs of city names such that a supplier in the first city supplies a project in the second city.

질의 그룹 # 2: 5 questions

<Aggregate Functions>

1. For each part being supplied to a project, get the part number, the project number, and the corresponding total quantity.
2. Get part numbers of parts supplied to some project in an average quantity of more than 320.

<Miscellaneous>

3. Get project numbers and cities where the city has an "o" as the second letter of its name.

<Subqueries>

4. Get project numbers for projects whose city is first in the alphabetic list of such cities.
5. Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.

질의 그룹 # 3: 4 questions

<EXISTS>

1. Get part numbers for parts supplied to any projects in London. (Use EXISTS in your solution.)
2. Get project numbers for projects not supplied with any red part by any London suppliers.
3. Get project numbers for projects supplied with at least all parts available from supplier S1.

<Union>

4. Construct an ordered list of all cities in which at least one supplier, part, or project is located.

질의 그룹 #4: 4 questions

<Update operations>

1. Delete all projects for which there are no shipments.
2. Insert a new supplier (S10) into table "Supplier". The name and city are Smith and New York, respectively; the status is not yet known.
3. Construct a table containing a list of part numbers for parts that are supplied either by a London supplier or to a London project.
4. Construct a table containing a list of project numbers for projects that are either located in London or are supplied by a London supplier.

질의 그룹 #5: 3 questions

<Data Definition Language>

1. Write a suitable set of CREATE TABLE statements for the sample database suppliers-parts-projects.
2. Write a set of CREATE INDEX statements for the sample database suppliers-parts-projects to enforce the required primary key constraints.
3. Create the sample database with sample data values.

질의 그룹 #6: 4 questions

<Views>

1. Create a view consisting of supplier numbers and part numbers for suppliers and parts that are not "colocated."
2. Create a view consisting of supplier records for suppliers that are located in London.
3. Create a view from the suppliers-parts-projects database consisting of all projects (project number and city fields only) that are supplied by supplier S1 and use part P1.

4. Given the view definition:

```
CREATE VIEW HEAVYWEIGHTS (P#, WT, COL)
AS SELECT P#, WEIGHT, COLOR
FROM Part
WHERE WEIGHT > 14;
```

show the result for each of the following SQL statement:

- a) SELECT *
 FROM HEAVYWEIGHTS
 WHERE COL = 'Green';

- b) UPDATE HEAVYWEIGHTS
 SET COL = 'White'
 WHERE WT = 18;

- c) DELETE
 FROM HEAVYWEIGHTS
 WHERE WT < 10;

- d) INSERT
 INTO HEAVYWEIGHTS(P#, WT, COL)
 VALUES ('P99', 12, 'Purle');

(각 그룹 문제에 답하기 위한)

과제를 위한 참고 질의 예제 제공

공과대학 인공지능공학부, 교수 박 영 호

질의 그룹 # 1

- 1) Simple retrieval. Get part numbers for all parts supplied.

```
SELECT P#  
FROM SPJ;
```

cf.

```
SELECT DISTINCT P#  
FROM SPJ;
```

- 2) Retrieval of computed values. For all parts, get the part number and the weight of that part in grams (part weights are given in table "Part" in pounds).

```
SELECT Part.P#, 'Weight in grams = ', Part.WEIGHT*454  
FROM Part;
```

- 3) Simple retrieval ("SELECT *"). Get full details of all suppliers.

```
SELECT *  
FROM Supplier;
```

- 4) Qualified retrieval. Get supplier numbers for suppliers in Paris with status > 20.

```
SELECT S#  
FROM Supplier  
WHERE CITY = 'Paris'  
AND STATUS > 20;
```

- 5) Retrieval with ordering. Get supplier numbers and status for suppliers in Paris, in descending order of status.

```
SELECT S#, STATUS  
FROM Supplier  
WHERE CITY = 'Paris'  
ORDER BY STATUS DESC;
```

- 6) Simple equijoin. Get all combinations of supplier and part information such that the supplier and part in question are located in the same city (i.e., are "colocated," to coin an ugly but convenient term).

```
SELECT Supplier.*, Part.*  
FROM Supplier, Part  
WHERE Supplier.CITY = Part.CITY;
```

- 7) Greater-than join. Get all combinations of supplier and part information such that the supplier city follows the part city in alphabetical order.

```
SELECT Supplier.*, Part.*  
FROM Supplier, Part  
WHERE Supplier.CITY > Part.CITY;
```

- 8) Join query with an additional condition. Get all combinations of supplier information and

part information where the supplier and part concerned are colocated, but omitting suppliers with status 20.

```
SELECT Supplier.*, Part.*
FROM      Supplier, Part
WHERE Supplier.CITY = Part.CITY
AND       Supplier.STATUS <> 20;
```

9) Retrieving specified fields from a join. Get all supplier-number/part-number combinations such that the supplier and part in question are colocated.

```
SELECT Supplier.S#, Part.P#
FROM      Supplier, Part
WHERE Supplier.CITY = Part.CITY;
```

10) Join of three tables. Get all supplier-number/part-number/project-number triples such that the indicated supplier, part, and project are colocated.

```
SELECT S#, P#, J#
FROM      Supplier, Part, Project
WHERE Supplier.CITY = Part.CITY
AND       Part.CITY = Project.CITY;
```

11) Joining a table with itself. Get all pairs of supplier numbers such that the two suppliers concerned are colocated.

```
SELECT FIRST.S#, ECOND.S#
FROM      Supplier FIRST, Supplier SECOND
WHERE FIRST.CITY = SECOND.CITY
AND       FIRST.S# < SECOND.S#;
```


질의 그룹 # 2

- 1) Aggregate function in the SELECT clause. Get the total number of suppliers.

```
SELECT COUNT(*)
FROM Supplier;
```

- 2) Aggregate function in the SELECT clause, with DISTINCT. Get the total number of suppliers currently supplying parts.

```
SELECT COUNT(DISTINCT S#)
FROM SPJ;
```

- 3) Aggregate function in the SELECT clause, with a condition. Get the number of shipments for part P2.

```
SELECT COUNT(*)
FROM SPJ
WHERE P# = 'P2';
```

- 4) Aggregate function in the SELECT clause, with a condition. Get the total quantity of part P2 supplied.

```
SELECT SUM(QTY)
FROM SPJ
WHERE P# = 'P2';
```

- 5) Use of GROUP BY. For each part supplied, get the part number and the total shipment quantity for that part.

```
SELECT P#, SUM(QTY)
FROM SPJ
GROUP BY P#;
```

- 6) Use of HAVING. Get part numbers for all parts supplied by more than one supplier.

```
SELECT P#
FROM SPJ
GROUP BY P#
HAVING COUNT(*) > 1;
```

- 7) Retrieval using LIKE. Get all parts whose names begin with the letter C.

```
SELECT Part.*
FROM Part
WHERE Part.PNAME LIKE 'C%';
```

- 8) Retrieval involving NULL. Get supplier numbers for suppliers with status greater than 25.

```
SELECT S#
FROM Supplier
WHERE STATUS > 25;
```

- 9) Retrieval involving a subquery. Get supplier names for suppliers who supply part P2.

```
SELECT SNAME
FROM Supplier
WHERE S# IN
( SELECT S#
```

```

FROM      SPJ
WHERE     P# = 'P2');

```

10) Subquery with multiple levels of nesting. Get supplier names for suppliers who supply at least one red part.

```

SELECT SNAME
FROM      Supplier
WHERE S#  IN
        ( SELECT S#
          FROM      SPJ
          WHERE     P# IN ( SELECT P#
                           FROM      Part
                           WHERE     COLOR = 'Red'));

```

11) Subquery with comparison operator other than IN. Get supplier numbers for suppliers who are located in the same city as supplier S1.

```

SELECT S#
FROM      Supplier
WHERE CITY = ( SELECT CITY
              FROM      Supplier
              WHERE S# = 'S1');

```

12) Aggregate function in a subquery. Get supplier numbers for suppliers with status value less than the current maximum status value in the Supplier table.

```

SELECT S#
FROM      Supplier
WHERE STATUS < ( SELECT MAX(STATUS)
                FROM      Supplier);

```

질의 그룹 # 3

- 1) Query using EXISTS. Get supplier names for suppliers who supply part P2.

```
SELECT SNAME
FROM Supplier
WHERE EXISTS (
    SELECT *
    FROM SPJ
    WHERE S# = Supplier.S#
    AND P# = 'P2'
);
```

- 2) Query using NOT EXISTS. Get supplier names for suppliers who do not supply part P2.

```
SELECT SNAME
FROM Supplier
WHERE NOT EXISTS (
    SELECT *
    FROM SPJ
    WHERE S# = Supplier.S#
    AND P# = 'P2');
```

- 3) Query using NOT EXISTS. Get supplier names for suppliers who supply all parts.

```
SELECT SNAME
FROM Supplier
WHERE NOT EXISTS (
    SELECT *
    FROM Part
    WHERE NOT EXISTS (
        SELECT *
        FROM SPJ
        WHERE S# = Supplier.S#
        AND P# = Part.P#));
```

- 4) Query using NOT EXISTS. Get supplier numbers for suppliers who supply at least all those parts supplied by supplier S2.

```
SELECT DISTINCT S#
FROM SPJ SPJ_X
WHERE NOT EXISTS (
    SELECT *
    FROM SPJ SPJ_Y
    WHERE S# = 'S2'
    AND NOT EXISTS (
        SELECT *
        FROM SPJ SPJ_Z
        WHERE SPJ_Z.S# = SPJ_X.S#
        AND SPJ_Z.P# = SPJ_Y.P#));
```

- 6) Query involving UNION. Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2 (or both).

```
SELECT P#
FROM Part
WHERE WEIGHT > 16
UNION
SELECT P#
FROM SPJ
WHERE S# = 'S2';
```

질의 그룹 # 4

1) Single-record UPDATE. Change the color of part P2 to yellow, increase its weight by 5, and set its city to "unknown" (NULL).

```
UPDATE      Part
SET         COLOR='Yellow', WEIGHT=WEIGHT+5, CITY=NULL
WHERE P# = 'P2';
```

2) Multiple-record UPDATE. Double the status of all suppliers in London.

```
UPDATE      Supplier
SET         STATUS = 2*STATUS
WHERE CITY = 'London';
```

3) UPDATE with a subquery. Set the shipment quantity to zero for all suppliers in London.

```
UPDATE      SPJ
SET         QTY = 0
WHERE 'London' = ( SELECT CITY
                  FROM      Supplier
                  WHERE      Supplier.S# = SPJ.S#);
```

4) Multiple-table UPDATE. Change the supplier number for supplier S2 to S9.

```
UPDATE      S
SET         S# = 'S9'
WHERE S# = 'S2';
```

```
UPDATE      SPJ
SET         S# = 'S9'
WHERE S# = 'S2';
```

6) Single-record DELETE. Delete supplier S5.

```
DELETE
FROM      Supplier
WHERE S# = 'S5';
```

6) Multiple-record DELETE. Delete all shipments with quantity greater than 300.

```
DELETE
FROM      SPJ
WHERE QTY > 300;
```

7) DELETE with a subquery. Delete all shipments.

```
DELETE
FROM      SPJ;
```

8) DELETE with a subquery. Delete all shipments for suppliers in London.

```
DELETE
FROM      SPJ
WHERE 'London' = ( SELECT CITY
                  FROM      Supplier
                  WHERE      Supplier.S# = SP.S#);
```

9) Single-record INSERT. Add part P7 (city Athens, weight 24, name and color at present unknown) to table P.

```
INSERT
INTO P          (P#, CITY, WEIGHT)
VALUES         ('P7', 'Athens', 24);
```

10) Single-record INSERT, with field names omitted. Add part P8 (name Sprocket, color Pink, weight 14, city Nice) to table Part.

```
INSERT
INTO      Part
VALUES    ('P8', 'Sprocket', 'Pink', 14, 'Nice');
```

11) Multiple-record INSERT. For each part supplied, get the part number and the total quantity supplied of that part, and save the result in the database.

```
CREATE TABLE TEMP
(P#          CHAR(6) NOT NULL,
TOTQTY      INTEGER    NOT NULL,
PRIMARY KEY (P#));
```

```
CREATE UNIQUE INDEX XT ON TEMP (P#);
```

```
INSERT
INTO TEMP (P#, TOTQTY)
SELECT P#, SUM(QTY)
FROM SPJ
GROUP BY P#;
```

질의 그룹 5.

1) CREATE TABLE statement for table S:

```
CREATE TABLE Supplier
  ( S#          CHAR(5)          NOT NULL,
    SNAME       CHAR(20)        NOT NULL,
    STATUS      SMALLINT        NULL,
    CITY        CHAR(15)        NOT NULL,
    PRIMARY KEY (S#));
```

SYBASE 에서는 NULL 또는 NOT NULL 이라고 명시하지 않는 경우, default 는 NOT NULL 이 된다(ANSI 에서는 default 가 NULL 이다).

2) DROP TABLE statement for table Supplier:

```
DROP TABLE Supplier;
```

3) CREATE INDEX statement for index XS on S# of tables Supplier:

```
CREATE UNIQUE INDEX XS ON Supplier (S#) CLUSTER;
```

SYBASE 라는 DBMS 에서는 다음과 같이 사용한다. (다른 DBMS 에서도 유사하다.)

```
CREATE UNIQUE CLUSTERED INDEX XS ON Supplier (S#);
```

4) DROP INDEX statement for index XS:

```
DROP INDEX XS;
```

질의 그룹 6.

1) Create a view called REDPARTS, consisting of part numbers, part names, weights, and citys for parts that are red.

```
CREATE VIEW REDPARTS (P#, PNAME, WT, CITY)
AS SELECT P#, PNAME, WEIGHT, CITY
FROM Part
WHERE COLOR = 'Red';
```

2) Create a view consisting of part numbers and their total supplied quantity.

```
CREATE VIEW PQ (P#, TOTQTY)
AS SELECT P#, SUM(QTY)
FROM SPJ
GROUP BY P#;
```

3) Create a view consisting of pairs of city names (x,y), where a supplier located in city x supplies a part stored in city y.

```
CREATE VIEW CITY_PAIRS (SCITY, PCITY)
AS SELECT DISTINCT Supplier.CITY, Part.CITY
FROM Supplier, SPJ, Part
WHERE Supplier.S# = SPJ.S#
AND SPJ.P# = Part.P#;
```

4) Create a view consisting of part numbers and weights for parts that are red and located in London.

```
CREATE VIEW LONDON_REDPARTS
AS SELECT P#, WT
FROM REDPARTS
WHERE CITY = 'London';
```

5) Create a view consisting of supplier numbers, status, and citys for suppliers that have a status value greater than 15.

```
CREATE VIEW GOOD_SUPPLIERS
AS SELECT S#, STATUS, CITY
FROM Supplier
WHERE STATUS > 15
WITH CHECK OPTION;
```