Database Management, 2019 Midterm 1

Q1. (6%) Explain the **entity integrity constraint, foreign key** and the **referential integrity constraint**.

Q2.
(a) (4%) Give an example to explain the differences between a key and a superkey.
(b) (4%) When a NULL is involved in a comparison operation, the result is considered to be UNKNOWN. Show the results of the following logical expression respectively: (i) FALSE OR（TRUE OR UNKNOWN）; (ii) UNKNOWN AND UNKNOWN; (iii) (TRUE AND UNKNOWN) OR UNKNOWN; (iv) (FALSE AND UNKNOWN) OR (TRUE OR UNKNOWN).

Q3. (5%) Describe three possible options to handle the **Delete** operation when a constraint is violated. Please use diagrams (examples) to aid your explanations.

Q4. Aggregate functions can be applied to a particular column (attribute), that is, a collection of values.
(a) (2%) Explain how the NULL values are handled when the AVG function is applied to the attribute Salary in the Employee table.
(b) (2%) If the collection becomes empty because all values are NULL, what will the COUNT function return?
(c) (4%) Explain how will the COUNT(*) handle the tuple (of the query result) that contain NULL values of some attributes in the tuple? Explain the differences between COUNT(Salary) and COUNT(Distinct Salary).

Q5. Below are three tables for “2019 Best Singer Battle!!” in NCTU:

STUDENT

Student_number	Name	Sex	Major
5	Tony	male	MIS
16	Kelly	female	FL
49	Jay	male	EE

SONG

Song_id	Language	Type	Producer
2	Taiwanese	Electropop	SeedMusic
11	Chinese	Pop	BinMusic
17	English	Country	SouthernMusic
34	English	Blues	SonyMusic
82	Chinese	R&B	EnjoyMusic

COMPETITION

Student_number	Song_id	Score
49	82	B
16	2	A
49	34	C
5	17	B
5	11	B

Write SQL update statement to do the following on the database schema shown in above Figure.
a. (4%) Update all the Competition Scores of the songs, in which Type is ’Pop’ and is sang by ‘Tony’ to ‘A’.
b. (3%) Insert a new student, <13,’Alice’,’female’,’IMF’>
c. (3%) Delete records from the Song table, in which Producer is ‘BinMusic’ and Type is ‘R&B’.
d. (5%) Write a SQL query to list the names of all male students who have at least two scores of ‘B’ for singing songs produced by ‘SonyMusic’ in the competitions.

Q6. Below is a subset of relations from COMPANY schema. The keys have been underlined.
EMPLOYEE (FNAME, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO)
DEPARTMENT (DNAME, DNUMBER, MGRSSN, MGRSTARTDATE)
PROJECT (PNAME, PNUMBER, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)
DEPENDENT (ESSN, DEPENDENT_NAME, SEX, BDATE, RELATIONSHIP)

Express the following Queries in SQL statements.

(1) (6%) Query 1: For each employee who works on the project controlled by the Research department, list the name of the employee and the names of the projects that he/she works on.

(2) (6%) Query 2: For each employee who has more than two dependents, retrieve the name, the salary, and the department name of the employee.

(3) (6%) Query 3: For each department with more than five employees, list the department name, the sum of salaries and the total number of employees of the department.

(4) (6%) Query 4: For each employee who works on the "Mountain Travel" project and whose salary is greater than the salary of his/her supervisor, list the name of employee and the name of his/her supervisor.

(5) (6%) Query 5: For each project located in "Hsinchu" and with more than ten employees working on, retrieve the project number, the project name, and the number of Male employees who work on the project.

(6) (6%) Query 6: For each employee who has no dependents and does not work on any project, list the name of the employee and the name of his/her manager. (hint: NOT EXISTS)

(7) (6%) Query 7: For each employee who has no dependents and whose number of working projects is greater than the number of working projects of every employee in department number 5, list the name of the employee and the name of his/her department.

(8) (6%) Query 8: Retrieve the name and salary of each employee who is the manager of the "R&D" department and is the direct supervisor of at least five employees.

(9)

(a) (6%) Query 9: Retrieve the name of each employee who works on all the projects on which the employee John Smith works.
(b) (4%) Assume that JSmithPNOs denotes the set of projects on which the employee John Smith works; and EmpPNOs denotes the set of projects on which an employee works. Draw the set diagrams to show four possible set relations between JSmithPNOs and EmpPNOS, and explain why NOT_EXISTS and EXCEPT can be used to correctly implement the set relation that EmpPNOS contains JSmithPNOS.

(10) (8%) For each employee who works for the AI project and has more than two supervisors (direct and indirect supervisors), list the name of the employee and the names of all his/her supervisees at all levels (direct and indirect supervisees).

Q1.

Entity Integrity(一致性) constraint：

The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).

This is because primary key values are used to identify the individual tuples.

t[PK] ≠ null for any tuple t in r(R)

If PK has several attributes, null is not allowed in any of these attributes

Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

foreign key：

reference the primary key attributes PK of the another referenced relation R2.

If a relation schema includes the primary key of another relation schema, that attribute is called the foreign key

EX：

EMPLOYEE

SSN	SUPERSSN	DNO
e1	e6	2
e3	e4	2
e4	e5	3

WORK_ON

ESSN	PNO	HOURS
e1	p1	5
e3	p1	8
e4	p3	7
e5	p4	6

DEPT

DNumber	Dname	MGRSSN
1	Develop	e21
2	Design	e21
3	AI	e39

WORK_ON's ESSN is a foreign key since it's reference to **EMPLOYEE**'s primary key (SSN)

EMPLOYEE's DNO is a foreign key because it is reference to **DEPT**'s primary key (Dnumber)

DEPT's MGRSSN is a foreign key because it is reference to **EMPLOYEE**'s primary key (SSN)

referential integrity constraint：

A constraint involving two relations

The previous constraints involve a single relation.

Used to specify a relationship among tuples in two relations:

The referencing relation and the referenced relation.

Tuples in the referencing relation R1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R2.

A tuple t1 in R1 is said to reference a tuple t2 in R2 if t1[FK] = t2[PK].

A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Statement of the constraint

The value in the foreign key column (or columns) FK of the the referencing relation R1 can be either:

(1) a value of an existing primary key value of a corresponding primary key PK in the referenced relation R2, or

(2) a null.

In case (2), the FK in R1 should not be a part of its own primary key

Q2

(a)

Superkey of R:

set of attributes of table for which every row has distinct set of values

Is a set of attributes SK of R with the following condition:

No two tuples in any valid relation state r(R) will have the same value for SK

That is, for any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK]

This condition must hold in any valid state r(R)

Key of R:

A "minimal" superkey

That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

A Key is a Superkey but not vice versa

In general:

Any key is a superkey (but not vice versa)

Any set of attributes that includes a key is a superkey

A minimal superkey is also a key

EX：

EMP

SSN	Salary
101	45k
103	45k
104	50k

Key: {SSN}、{SSN, Salary}

Superkey: {SSN}

PEOPLE

Sex	Name	Birthday
Male	Jason	1031
Female	Winni	1031
Female	Jason	1031

Key: {Sex, Name }、{Sex, Name, Birthday}

Superkey: {Sex, Name}

(b)

(i) FALSE OR（TRUE OR UNKNOWN）→ FALSE OR TRUE → TRUE

(ii) UNKNOWN AND UNKNOWN → UNKNOWN

(iii) (TRUE AND UNKNOWN) OR UNKNOWN → UNKNOWN OR UNKNOWN → UNKNOWN

(iv) (FALSE AND UNKNOWN) OR (TRUE OR UNKNOWN) → FALSE OR TRUE → TRUE

Q3

RESTRICT option: reject the deletion

CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples

SET NULL option: set the foreign keys of the referencing tuples to NULL

	EMP			DEPT	
	SSN	DNUM		DNUM	DLO
	1	3		3	TW
t1	2	4	t1	4	AM
	3	5		5	CN

SET NULL: t2[Dnum] would be set to NULL

CASCADE： t2 would be delete

Q4

(a)

NULL values are discarded when aggregate functions are applied to a particular column

The average of all values in the “Salary” column in the “Employee” table

AVG() ignore “NULL”, instead of counting as 0.

It would be eliminated before calculation

(b)

COUNT(col) ： 0

COUNT(*) ：the number of rows

(c)

COUNT(*) ：

will return the total of all records returned in the result set regardless of NULL values.

counts the number of rows.

Count the number of tuple not of attribute ➔ the tuple would be counted in

COUNT(Salary) ：The number of values in the Salary column

Count as many as the number of these tuples

COUNT(Distinct Salary) ：The number of unique values in the Salary column

Only count once

Q5

a

```
UPDATE COMPETITION
SET Score = ‘A’
WHERE Song_id IN (SELECT Song_id
                  FROM SONG
```

```
                    WHERE Type = 'Pop')
            AND Student_number IN (SELECT Student_number
                                   FROM STUDENT
                                   WHERE Name = 'Tony');
```

b

```
INSERT INTO STUDENT
VALUES (13,'Alice','female','IMF');
```

c

```
DELETE FROM SONG
WHERE Producer = 'BinMusic' and Type = 'R&B';
```

d

```
SELECT Distinct Name
FROM STUDENT
WHERE STUDENT. Student_number IN (SELECT Student_number
                                  FROM SONG, COMPETITION
                                  WHERE SONG.Producer = 'SonyMusic'
                                        AND
                                  COMPETITION.Song_id = SONG.Song_id
                                  AND
                                   COMPETITION.Score = 'B'
                                 GROUP BY SONG.Student_number
                                 HAVING COUNT (*) >= 2);
```

Q6

(1)

```
SELECT EMPLOYEE.FNAME, EMPLOYEE.LNAME, PROJECT.PNAME
FROM EMPLOYEE, WORKS_ON
WHERE EMPLOYEE.SSN = WORKS_ON.ESSN AND WORKS_ON.PNO IN
(
SELECT PROJECT.PNUMBER
FROM DEPARTMENT, PROJECT
WHERE PROJECT.DNUM = DEPARTMENT.DNUMBER AND DNAME = 'Research'
);
```

(2)

```
SELECT EMPLOYEE.FNAME, EMPLOYEE.LNAME, EMPLOYEE.SALARY,
DEPARTMENT.DNAME
FROM EMPLOYEE, PROJECT, DEPARTMENT
WHERE EMPLOYEE.SSN = WORKS_ON.ESSN
   AND WORKS_ON.PNO = PROJECT.PNUMBER
   AND PROJECT.DNUM = DEPARTMENT.DNUMBER
   AND EMPLOYEE.SSN IN (SELECT DEPENDENT.ESSN
                        FROM DEPENDENT
                        GROUP BY DEPENDENT.ESSN
```

```
                                    HAVING COUNT (*) > 2);
```

(3)

```
SELECT DEPARTMENT.DNAME, SUM(SALARY), COUNT (*)
FROM WORKS_ON, PROJECT, DEPARTMENT
WHERE WORKS_ON.PNO = PROJECT.PNUMBER
    AND PROJECT.DNUM = DEPARTMENT.DNUMBER
GROUP BY PROJECT.DNUM
HAVING COUNT (*) > 5;
```

(4)

```
SELECT EMPLOYEE.FNAME, EMPLOYEE.LNAME, PROJECT.PNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SALARY > S.SALARY AND E.SUPERSSN = S.SSN AND E.SSN IN
(
SELECT EE.SSN
FROM EMPLOYEE EE, WORKS_ON, PROJECT
WHERE WORKS_ON.PNO = PROJECT.PNUMBER
    AND PROJECT. PNAME = 'Mountain Travel'
    AND EE.SSN = WORKS_ON.ESSN
);
```

(5)

```
SELECT PROJECT.PNAME, PROJECT.PNUMBER COUNT (*)
FROM EMPLOYEE, WORKS_ON W1
WHERE EMPLOYEE.SEX = 'MALE' AND EMPLOYEE.SSN = W2.ESSN
  AND W1.PNO IN (
        SELECT W2.PNO
        FROM WORKS_ON W2, PROJECT
        WHERE W2.PNO = PROJECT.PNUMBER AND PROJECT.PLOCATION = "Hsinchu"
        GROUP BY W2.PNO
        HAVING COUNT (*) > 10 );
```

(6)

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN = S.SSN
  AND E.SSN NOT EXISTS ((SELECT DISTINCT DEPENDENT.ESSN
                          FROM DEPENDENT)
                        UNION
                         (SELECT DISTINCT WORKS_ON.ESSN
                          FROM PROJECT, WORKS_ON
                          WHERE WORKS_ON.PNO = PROJECT.PNUMBER));
```

(7)

```
SELECT EMPLOYEE.FNAME, EMPLOYEE.LNAME, DEPARTMENT.DNAME
```

```
FROM EMPLOYEE, WORKS_ON W1, PROJECT P1, DEPARTMENT
WHERE EMPLOYEE.SSN = W1.ESSN
   AND W1.PNO = P1.PNUMBER
   AND P2.DNUM = DEPARTMENT.DNUMBER
   AND EMPLOYEE.SSN NOT EXISTS (SELECT DISTINCT DEPENDENT.ESSN
                                FROM DEPENDENT)
GROUP BY W1.ESSN
HAVING COUNT (*) > (SELECT  MAX(COUNT (*)
                    FROM WORKS_ON W2, PROJECT P2
                    WHERE W2.PNO = P2.PNUMBER
                       AND P2.DNUM = 5
                    GROUP BY W2.ESSN);
```

(8)

```
SELECT E.FNAME, E.LNAME, E.SALARY
FROM EMPLOYEE E, DEPARTMENT
WHERE E.SSN = MGRSSN
   AND DEPARTMENT.DNAME = 'R&D'
   AND DEPARTMENT.MGRSSN IN (SELECT S.SUPERSSN
                             FROM EMPLOYEE S
                             GROUP BY S.SUPERSSN
                             HAVING COUNT (*) > 5 );
```

(9)

(a)

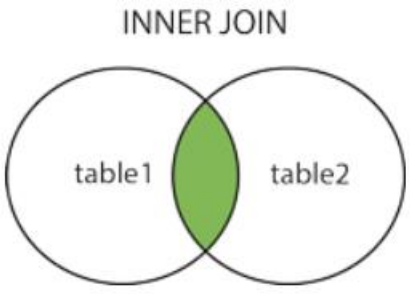
```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE E, WORKS_ON W1
WHERE E.SSN = W1.ESSN AND W1.PNO IN (SELECT WORKS_ON.PNO
                                     FROM EMPLOYEE JS, WORKS_ON W2
                                     WHERE JS.FNAME =  'John'
                                     AND JS.LNAME = 'Smith'
                                     AND JS.SSN = W2.ESSN);
```

(b)

INNER JOIN

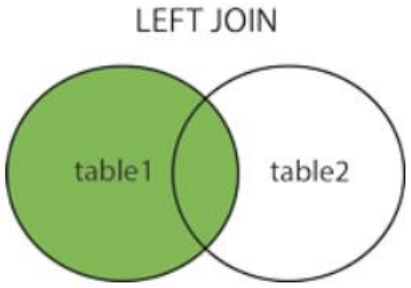
Default type of join in a joined table

Tuple is included in the result only if a matching tuple exists in the other relation

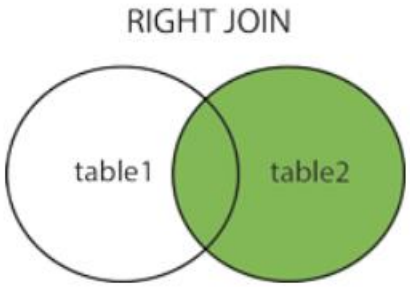


LEFT OUTER JOIN

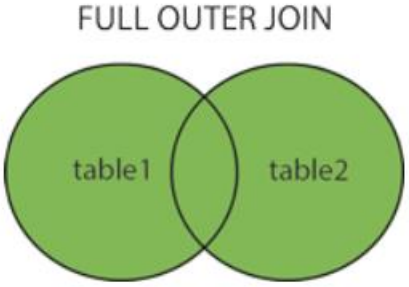
Every tuple in left table must appear in result
If no matching tuple
Padded with NULL values for attributes of right table



RIGHT OUTER JOIN
Every tuple in right table must appear in result
If no matching tuple
Padded with NULL values for attributes of left table



FULL OUTER JOIN
combines result if LEFT and RIGHT OUTER JOIN



To achieve the “for all” effect, we use double negation this way in SQL:

NOT EXIST(J EXCEPT E) ➔ true -> implies J 屬於 E
➔ false -> implies E 屬於 J

(10)

```
WITH RECURSIVE SUP_EMP (SUPERSSN, SSN) AS
(SELECT SUPERSSN, SSN
FROM EMPLOYEE
        UNION
SELECT E.SSN, S.SUPSSN
FROM EMPLOYEE AS E, SUP_EMP AS S
```

```
        WHERE E.SUPERSSN = S.SSN)
SELECT SUP_EMP.FNAME, SUP_EMP.LNAME, SUPORVISOR.FNAME, SUPORVISOR.LNAME
FROM SUP_EMP, EMPLOYEE SUPORVISOR, WORKS_ON, PROJECT
WHERE SUP_EMP.SSN = WORKS_ON.ESSN
   AND WORKS_ON.PNUM = PROJECT.PNUMBER
   AND PROJECT.PNAME = 'AI'
   AND SUP_EMP.SUPERSSN = SUPORVISOR.SSN
GROUP BY SUP_EMP.SSN
HAVING COUNT(*) > 2;
```