Q1.

Entity Integrity(一致性) constraint：

no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples. For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations.

Primary key 代表 entity，所以 primary key 不能為 NULL

Primary key 是代表資料庫每一筆 tuple 的 idntity，所以 primary key 不能有 NULL 因為 domain constraints，key 是 unique 的

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

Reference 其他 table 的 primary key 的 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：

foreign key either reference 存在的值 or 是 NULL

specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation

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:

屬性的值具有唯一性

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

保有唯一性最少需要的屬性

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:

Super key 的集合比 key 大

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

Difference:

Super key 的集合比 key 大

Superkey 是屬性的值具有唯一性就好，而 Key 是保有唯一性最少需要的屬性

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

通常是違反 referential integrity constraint：

RESTRICT option: reject the deletion

CASCADE option: 有擴散性，跟著刪除，propagate the new primary key value into the foreign keys of the referencing tuples

SET NULL/ SET Default option: set the foreign keys of the referencing tuples to NULL or default value

EX:

EMP’s DNO reference to DEPT’s Dnumber

EMP

SSN		DNO
		2
		4
		4
		4

DEPT

Dnumber		
1		
2		
3		
4		

如果要刪除 DEPT 中 Dnumber 為 4 的

RESTRICT option：禁止你刪因為 EMP 中的 DNO(foreign key)有為 4 的

CASCADE option：有擴散性，跟著刪除，刪除所有 EMP 中的 DNO(foreign key)為 4 的

SET NULL/ SET Default option：將所有 EMP 中的 DNO(foreign key)為 4 的設為 NULL
或是預設值

Q4
(a)

通常不算 NULL 值

EMP

Salary
50000
60000
NULL
40000

Count: 3 Sum: 150000 AVG: 50000

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)

全部都是 empty ➔ Count return 0

如果是用 Agggregate function 且全部都是 NULL ➔ NULL

(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) ：

有重複值會照算、NULL 值不算
The number of values in the Salary column
Count as many as the number of these tuples

COUNT(Distinct Salary) ：

有重複值不會照算、NULL 值不算
The number of unique values in the Salary column
Only count once

EX:

EMP

Salary
40000
40000
50000
50000
60000
NULL

COUNT (*)：6 COUNT(Salary) ： 5 COUNT(Distinct Salary) ： 3

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 STUDENT.Name
FROM STUDENT
WHERE STUDENT.Sex = ‘male’ AND (SELECT COUNT (*)
                                FROM SONG, COMPETITION
                                WHERE SONG.Producer = ‘SonyMusic’
                                AND COMPETITION.Song_id = SONG.Song_id
                                 AND COMPETITION.Score = ‘B’
                                 AND
                                SONG.Student_number = COMPETITION.Student_number
                                  ) >= 2;
```

Q6

(1)

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

(2)

```
SELECT EMPLOYEE.FNAME, EMPLOYEE.LNAME, EMPLOYEE.SALARY,
DEPARTMENT.DNAME
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNUMBER
   AND (SELECT COUNT (*)
          FROM DEPENDENT
          WHERE EMPLOYEE.SSN = DEPENDENT.ESSN) > 2;
```

(3)

```
SELECT DEPARTMENT.DNAME, SUM(SALARY), COUNT (*)
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNUMBER
GROUP BY DEPARTMENT.DNAME
HAVING COUNT (*) > 5;
```

(4)

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S, WORKS_ON, PROJECT
WHERE E.SALARY > S.SALARY
   AND E.SUPERSSN = S.SSN
   AND WORKS_ON.PNO = PROJECT.PNUMBER
   AND PROJECT. PNAME = ‘Mountain Travel’
   AND E.SSN = WORKS_ON.ESSN;
```

(5)

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

(6)

```
SELECT E.FNAME, E.LNAME, M.FNAME, M.LNAME
FROM EMPLOYEE E, DEPARTMENT, EMPLOYEE M
WHERE E.DNO = DEPARTMENT.DNUMBER
   AND DEPARTMENT.MGRSSN = M.SSN
   AND NOT EXISTS (SELECT *
                    FROM DEPENDENT
                    WHERE E.SSN = DEPENDENT.ESSN)
   AND NOT EXISTS (SELECT *
                    FROM WORKS_ON
                    WHERE E.SSN = WORKS_ON.ESSN);
```

(7)

```
WITH DNOfiveP (ESSN, PCOUNT) AS
     (SELECT WORKS_ON.ESSN, COUNT (*)
     FPOM EMPLOYEE , WORKS_ON
     WHERE EMPLOYEE.DNO = 5 AND WORKS_ON.ESSN = EMPLOYEE.SSN
     GROUP BY EMPLOYEE.SSN)
SELELT E.FNAME, E.LNAME, DEPARTMENT.DNAME
FROM DEPARTMENT, EMPLOYEE E
```

```
WHERE E.DNO = DEPARTMENT.DNUMBER
  AND NOT EXTSTS (SELECT *
                        FROM DEPEDENT
                        WHERE DEPEDENT.ESSN= E.SSN)
  AND (SELECT COUNT (*)
         FROM WORKS_ON W
         WHERE W.ESSN = E.SSN)
          > ALL (SELECT PCOUNT
                  FROM DNOfiveP);
```

(8)

```
SELECT E.FNAME, E.LNAME, E.SALARY
FROM EMPLOYEE E
WHERE EXISTS (SELECT *
                    FROM DEPARTMENT
                    WHERE E.SSN = DEPARTMENT.MGRSSN
                       AND  DNAME = ‘R&D’)
   AND (SELECT COUNT (*)
          FROM EMPLOYEE S
          WHERE E.SSN = S.SUPERSSN) >= 5;
```

(9)

(a)

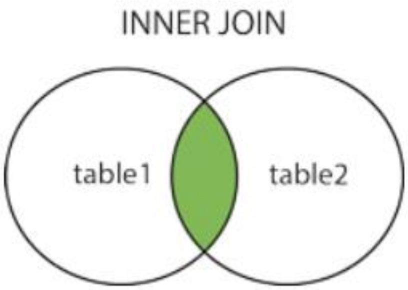
```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE E
WHERE NOT EXISTS ((SELECT W1.PNO
                         FROM EMPLOYEE JS, WORKS_ON W1
                         WHERE JS.FNAME =  ‘John’
                            AND JS.LNAME = ‘Smith’
                            AND JS.SSN = W1.ESSN)
                         EXCEPT (SELECT W2.PNO
                                    FROM WORKS_ON W2
                                    WHERE 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

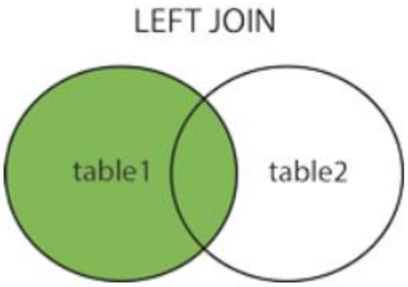


LEFT OUTER JOIN

Every tuple in left table must appear in result

If no matching tuple

Padded with NULL values for attributes of right table

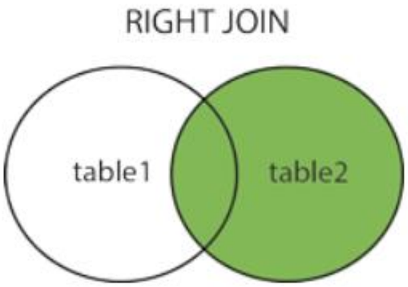


RIGHT OUTER JOIN

Every tuple in right table must appear in result

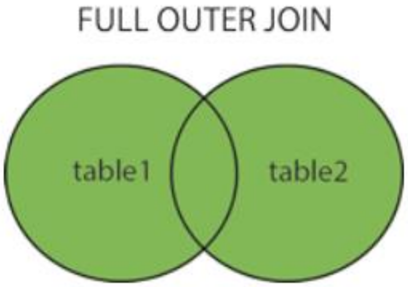
If no matching tuple

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 (SupSSN, EmpSSN) AS
        (SELECT SUPERSSN, SSN
        FROM EMPLOYEE
                UNION
        SELECT S.SupSSN, E.SSN
        FROM EMPLOYEE AS E, SUP_EMP AS S
        WHERE E.SUPERSSN = S.EmpSSN)
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, WORKS_ON, PROJECT, SUP_EMP SE, EMPLOYEE S
WHERE E.SSN = SE.SupSSN
   AND SE.EmpSSN = S.SSN
   AND E.SSN = WORKS_ON.ESSN
   AND WORKS_ON.PNO = PROJECT.PNUMBER
   AND PROJECT.PNAME = ‘AI’
   AND (SELECT (*)
          FROM SUP_EMP
          WHERE E.SSN = EmpSSN) > 2;
```