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] ≠ 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

DEPT

|  |  |  |
| --- | --- | --- |
| Dnumber |  |  |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

EMP

|  |  |  |
| --- | --- | --- |
| SSN |  | DNO |
|  |  | 2 |
|  |  | 4 |
|  |  | 4 |
|  |  | 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: 3Sum: 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

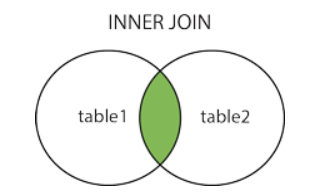
WHERE JS.SSN = W2.ESSN));

(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

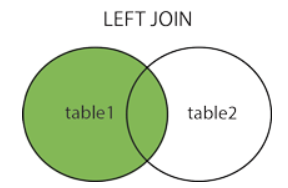


LEFT OUTER JOIN

Every tuple in left table must appear in result

If no matching tuple

Padded with NULL values for attributes of right table

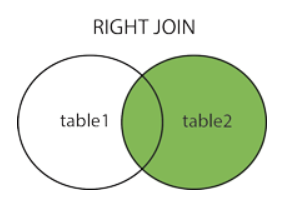


RIGHT OUTER JOIN

Every tuple in right table must appear in result

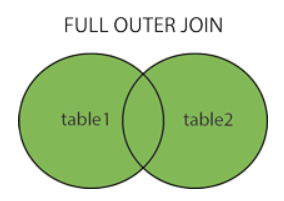
If no matching tuple

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;