INT1313 – Cơ Sở Dữ Liệu

Mô hình Dữ Liệu Quan hệ và SQL

SQL nâng cao

Giảng viên: Lê Hà Thanh

lehathanh@ptithcm.edu.vn

Hành trình

- Mô hình Dữ liệu Quan hệ
- Các Ràng buộc CSDL Quan hệ
- SQL
- Đại số quan hệ và Phép tính quan hệ
- Thiết kế Lược đồ CSDL quan hệ từ lược đồ khái niệm

Nội dung

- Truy vấn SQL phức tạp
- Đánh giá kết quả và bộ kích hoạt
- Các chế độ xem (views) trong SQL
- Các câu lệnh điều chỉnh lược đồ trong SQL

Truy vấn SQL phức tạp

So sánh liên quan đến NULL

Có ba cách diễn giải ý nghĩa của NULL:

- Giá trị không biết (*unknown*): giá trị có tồn tại nhưng không được biết đến, hoặc cũng không rõ liệu giá trị có tồn tại hay không.

  VD: ngày sinh hay số điện thoại nhà của một người.

- Giá trị không khả dụng (*not available*): giá trị có tồn tại nhưng được cố ý giữ lại không sử dụng.

  VD: một người có dùng điện thoại nhưng không muốn khai báo số này khi nhập liệu

- Giá trị không dùng được (*not applicable*): thuộc tính không áp dụng được cho bộ dữ liệu hoặc không được định nghĩa cho bộ này.

  VD: thông tin bằng cấp sau đại học không áp dụng cho một số người.

Logic ba giá trị

Khi một bản ghi có thuộc tính NULL xuất hiện trong phép so sánh, kết quả được xem là UNKNOWN (có thể TRUE, có thể FALSE).

→SQL sử dụng logic ba giá trị.

Bảng chân trị:

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Kiểm tra NULL trong SQL

Dùng toán tử kiểm tra **IS** hoặc IS **NOT**.

Ví dụ: "Lấy tên tất cả nhân viên không có người quản lý"

```
Q18:  SELECT    Fname, Lname
      FROM      EMPLOYEE
      WHERE     Super_ssn IS NULL;
```

Lệnh Truy vấn lồng nhau

- Các lệnh truy vấn có thể lồng nhau.
- Khối lệnh lồng là lệnh truy vấn select-from-where hoàn chỉnh bên trong một lệnh SQL khác.
- Khối lệnh lồng có thể xuất hiện trong:
  - Mệnh đề WHERE, còn gọi là truy vấn lồng nhau có liên quan, hoặc
  - Mệnh đề FROM, hoặc
  - Mệnh đề SELECT, hoặc
  - Các mệnh đề SQL khác.

Lệnh Truy vấn lồng nhau – Ví dụ 1

Dùng toán tử so sánh **IN** để duyệt các giá trị có trong tập kết quả của khối lệnh lồng.

Query 4: "Tạo một danh sách gồm tất cả các mã số dự án có sự tham gia của nhân viên có họ là 'Smith', dưới vai trò là công nhân (worker) hoặc là người quản lý của phòng ban (manager) đang kiểm soát dự án."

```
Q4A:  SELECT   DISTINCT Pnumber
      FROM     PROJECT
      WHERE    Pnumber IN
               ( SELECT    Pnumber
                 FROM      PROJECT, DEPARTMENT, EMPLOYEE
                 WHERE     Dnum = Dnumber AND Mgr_ssn = Ssn AND Lname = 'Smith' )
               OR
               Pnumber IN
               ( SELECT    Pno
                 FROM      WORKS_ON, EMPLOYEE
                 WHERE     Essn = Ssn AND Lname = 'Smith' );
```

Lệnh Truy vấn lồng nhau – Ví dụ 2

Query: “Chọn ra số Essn của tất cả nhân viên làm việc cùng (project, hours) trong các dự án mà nhân viên ‘John Smith’ (có Ssn=‘123456789’) đang làm.”

```
Q:      SELECT   DISTINCT Essn
        FROM     WORDKS_ON
        WHERE    (Pno, Hours) IN
                 ( SELECT   Pno, Hours
                   FROM     WORKS_ON
                   WHERE    Essn = ‘123456789’ );
```

Lệnh Truy vấn lồng nhau – Ví dụ 3

Để so sánh một giá trị v với tập kết quả V của khối lệnh lồng:

- = ANY (= SOME): TRUE nếu giá trị v bằng một giá trị nào đó trong tập V
- ALL, ví dụ: ($v > ALL\ V$) : TRUE nếu v lớn hơn mọi giá trị có trong tập V.
- Kết hợp ANY hoặc SOME, hoặc ALL với >, >=, < <=, <>

Query: "Chọn ra tên các nhân viên có mức lương cao hơn lương của tất cả các nhân viên làm tại phòng ban số 5."

Q:
```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   Salary > ALL
        ( SELECT  Salary
          FROM    EMPLOYEE
          WHERE   Dno = 5 );
```

Lệnh Truy vấn lồng nhau – Ví dụ 4

Phải xác định rõ phạm vi tham chiếu đến tên thuộc tính trong các truy vấn lồng nhau.

Query 16: "Lấy ra tên từng nhân viên có người phụ thuộc có cùng tên và giới tính với nhân viên đó."

```
Q16:  SELECT  E.Fname, E.Lname
      FROM    EMPLOYEE AS E
      WHERE   E.Ssn IN
              ( SELECT  D.Essn
                FROM    DEPENDENT AS D
                WHERE   E.Fname = D.Dependent_name
                        AND E.Sex = D.Sex );
```

Ví dụ 4, viết lại

```
Q16A:  SELECT   E.Fname, E.Lname
       FROM     EMPLOYEE AS E, DEPENDENT AS D
       WHERE    E.Ssn = D.Essn AND E.Sex = D.Sex
                AND E.Fname = D.Dependent_name;
```

Hàm EXISTS(Q)

- TRUE: nếu tập kết quả truy vấn Q có ít nhất một bộ (tập khác rỗng),
- FALSE: nếu tập kết quả truy vấn Q là tập rỗng.

Query 16: "Lấy ra tên từng nhân viên có người phụ thuộc có cùng tên và giới tính với nhân viên đó."

```
Q16B: SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    EXISTS  ( SELECT   *
                         FROM     DEPENDENT AS D
                         WHERE    E.Ssn = D.Essn AND E.Sex = D.Sex
                                  AND E.Fname = D.Dependent_name);
```

Hàm NOT EXISTS(Q)

- TRUE: nếu tập kết quả truy vấn Q là tập rỗng,
- FALSE: nếu ngược lại.

Query 6: "Lấy ra tên các nhân viên không có người phụ thuộc."

```
Q6:     SELECT   Fname, Lname
        FROM     EMPLOYEE
        WHERE    NOT EXISTS  ( SELECT   *
                               FROM     DEPENDENT
                               WHERE    Ssn = Essn);
```

Ví dụ - Q7

Query 7: “Liệt kê tên các quản lý có ít nhất một người phụ thuộc.”

```
Q7:   SELECT   Fname, Lname
      FROM     EMPLOYEE
      WHERE    EXISTS   ( SELECT   *
                          FROM     DEPENDENT
                          WHERE    Ssn = Essn)
               AND
               EXISTS   ( SELECT   *
                          FROM     DEPARTMENT
                          WHERE    Ssn = Mgr_ssn);
```

Ví dụ – Q3A

Query 3: “Lấy tên các nhân viên làm việc trong tất cả các dự án được quản lý bởi phòng ban số 5.”

```
Q3A:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     NOT EXISTS ( ( SELECT    Pnumber
                                 FROM      PROJECT
                                 WHERE     Dnum = 5)
                                 EXCEPT    ( SELECT    Pno
                                             FROM      WORKS_ON
                                             WHERE     Ssn = Essn) );
```

Ví dụ – Q3B

Query 3: “Lấy tên các nhân viên làm việc trong tất cả các dự án được quản lý bởi phòng ban số 5.”

```
Q3B:  SELECT  Lname, Fname
      FROM    EMPLOYEE
      WHERE   NOT EXISTS  ( SELECT  *
                            FROM    WORKS_ON B
                            WHERE   ( B.Pno IN  ( SELECT  Pnumber
                                                  FROM    PROJECT
                                                  WHERE   Dnum = 5 )
                            AND
                            NOT EXISTS  ( SELECT  *
                                          FROM    WORKS_ON C
                                          WHERE   C.Essn = Ssn
                                          AND     C.Pno = B.Pno )));
```

Hàm UNIQUE(Q)

- TRUE: nếu không có các bộ bị trùng lặp trong tập kết quả của truy vấn Q,
- FALSE: ngược lại.

→ Dùng kiểm tra xem kết quả của truy vấn lồng có phải là một tập hợp (không có bộ trùng) hay là một tập hợp có trùng lặp

DISCTINT – Phân biệt giá trị

Query 17: “Lấy ra các số an sinh xã hội (SSN) của tất cả người lao động đang làm trong các dự án có số hiệu 1, 2 hoặc 3”

```
Q17:  SELECT   DISTINCT  Essn
      FROM     WORKS_ON
      WHERE    Pno IN (1, 2, 3);
```

AS – Đổi tên thuộc tính

Query 8: “Lấy họ của từng người lao động và người quản lý của người đó.”

Q8A:
```
SELECT  E.Lname AS Employee_name, S.Lname AS Supervisor_name
FROM    EMPLOYEE AS E, EMPLOYEE AS S
WHERE   E.Super_ssn = S.Ssn;
```

- Tên mới của thuộc tính sẽ hiển thị ở tiêu đề cột tương ứng trong kết quả truy vấn.

Kết (JOIN) các bảng

- Bảng kết (quan hệ kết) là bảng kết quả của thao tác kết tại mệnh đề FROM trong câu truy vấn.
- Là phép kết mặc định được sử dụng khi kết một bảng, trong đó một bộ giá trị được đưa vào kết quả nếu có một bộ giá trị tương hợp tồn tại trong bảng còn lại
- Bảng kết quả tại mệnh đề FROM có toàn bộ các thuộc tính của cả hai bảng, theo thứ tự khai báo bảng tương ứng.
- Còn gọi là phép kết nội (**INNER JOIN**)

Query 1: *"Lấy tên và địa chỉ của tất cả người lao động đang làm tại phòng Nghiên cứu - 'Research'."*

Q1A:
```
SELECT   Fname, Lname, Address
FROM     (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
WHERE    Dname = 'Research';
```

NATURAL JOIN – Kết tự nhiên

- Kết tự nhiên giữa hai quan hệ R và S không cần khai báo điều kiện kết.
- Áp dụng điều kiện ngầm định EQUIJOIN cho từng cặp thuộc tính cùng tên ở R và S.
- Nếu tên của các thuộc tính kết ở hai quan hệ khác nhau, có thể khai báo đổi tên thành giống nhau và áp dụng phép kết tự nhiên.

Q1B: **SELECT** Fname, Lname, Address
FROM (EMPLOYEE **NATURAL JOIN**
(DEPARTMENT **AS** DEPT (DNAME, Dno, MSSN, MSDATE)))
WHERE Dname = 'Research'**;**

OUTER JOIN – Phép Kết ngoại các bảng

- Kết ngoại trái (**LEFT OUTER JOIN**): mọi bộ trong bảng bên trái phải xuất hiện trong kết quả; nếu không có bộ khớp, bảng kết quả sẽ được đệm bằng các giá trị NULL cho các thuộc tính của bảng bên phải.
  = **LEFT JOIN**

- Kết ngoại phải (**RIGHT OUTER JOIN**): mọi bộ trong bảng bên phải phải xuất hiện trong kết quả; nếu không có bộ khớp, bảng kết quả sẽ được đệm bằng các giá trị NULL cho các thuộc tính của bảng bên trái.
  = **RIGHT JOIN**

- Kết ngoại đầy đủ (**FULL OUTER JOIN**):
  = **FULL JOIN**

OUTER JOIN – Ví dụ

```
Q8B:   SELECT  E.Lname AS Employee_name,
                S.Lname AS Supervisor_name
       FROM    (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S
                ON  E.Super_ssn = S.Ssn);
```

Các biến thể của phép kết

- Nếu các thuộc tính kết có cùng tên, có thể áp dụng kết tự nhiên cho các phép kết ngoại.
  Ví dụ:

  NATURAL LEFT OUTER JOIN

- **CROSS JOIN:** dùng tích Decartes để xác định phép kết.
  Cẩn thận khi sử dụng vì kết quả chứa tất cả các tổ hợp bộ giá trị!!

Phép kết đa chiều

Cho phép thực hiện kết từ ba bảng trở lên.

Ví dụ:

Query 2: *"Với mọi dự án triển khai tại 'Stafford', hãy liệt kê mã số dự án, mã số phòng ban quản lý, và họ, địa chỉ, ngày sinh của người quản lý phòng ban này."*

```
Q2A:  SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     ((PROJECT JOIN DEPARTMENT ON Dnum = Dnumber)
                 JOIN EMPLOYEE ON Mgr_ssn = Ssn)
      WHERE    Plocation = 'Stafford';
```

Các hàm tổng hợp (Aggregate)

- Các hàm tổng hợp được sử dụng để tóm tắt thông tin từ nhiều bộ thành một bản tóm tắt trong một bộ kết quả.

- Gom nhóm (grouping) được sử dụng để tạo các nhóm con của các bộ trước khi thực hiện tóm tắt.

- Các hàm tổng hợp có sẵn: **COUNT**, **SUM**, **MAX**, **MIN**, **AVG**.

Ví dụ: Q19

Query 19: *"Tìm tổng lương của tất cả người lao động, mức lương cao nhất, mức lương thấp nhất, và lương trung bình."*

```
Q19:   SELECT   SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
       FROM     EMPLOYEE;
```

```
Q19A:  SELECT   SUM(Salary) AS Total_Sal, MAX(Salary) AS Highest_Sal,
                MIN(Salary) AS Lowest_Sal, AVG(Salary) AS Average_Sal
       FROM     EMPLOYEE;
```

Ví dụ: Q20, 21, 22

Query 20: "*Tìm tổng lương của tất cả người lao động thuộc phòng Nghiên cứu - 'Research', và mức lương cao nhất, mức lương thấp nhất, và lương trung bình trong phòng này.*"

```
Q20:  SELECT  SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
      FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
      WHERE   Dname = 'Research';
```

Query 21, 22: "Lấy tổng số nhân viên trong công ty (Q21) và số nhân viên thuộc phòng Nghiên cứu - 'Research'."

```
Q21:  SELECT  COUNT(*)
      FROM    EMPLOYEE;
```

```
Q22:  SELECT  COUNT(*)
      FROM    EMPLOYEE, DEPARTMENT;
      WHERE   Dno = Dnumber AND Dname = 'Research';
```

Ví dụ: Q23

Query 23: *"Đếm số lượng các giá trị lương phân biệt trong CSDL."*

Q23: **SELECT** **COUNT** (**DISTINCT** Salary)
FROM EMPLOYEE;

Ví dụ: Q5

Query 5: *"Lấy tên tất cả người lao động có từ hai người phụ thuộc trở lên."*

```
Q5:  SELECT   Lname, Fname
     FROM     EMPLOYEE
     WHERE    ( SELECT     COUNT(*)
                FROM       DEPENDENT
                WHERE      Ssn = Essn) >= 2;
```

Mệnh đề GROUP BY – Gom nhóm theo thuộc tính

Thực hiện gom nhóm các bộ dữ liệu dựa trên giá trị của thuộc tính được chọn để phân nhóm.

= Phân chia quan hệ thành các vùng tách biệt.

- ***Chú ý***: Nếu thuộc tính gom nhóm có chứa giá trị NULL, việc gom nhóm sẽ tạo ra một nhóm riêng gồm các bộ có giá trị NULL tại thuộc tính này.

Mệnh đề **GROUP BY** – Ví dụ: Q24

Query 24: "*Với từng phòng ban, hãy lấy ra mã số, số lượng nhân sự của phòng ban, và mức lương trung bình của họ.*"

```
Q24:  SELECT      Dno, COUNT(*), AVG(Salary)
      FROM        EMPLOYEE
      GROUP BY    Dno;
```

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

GROUP BY – Ví dụ: Q25

Query 25: *"Với mỗi dự án, hãy lấy ra mã số dự án, tên dự án và số lượng nhân sự làm việc trong dự án đó."*

```
Q25:    SELECT      Pnumber, Pname, COUNT(*)
        FROM        PROJECT, WORKS_ON
        WHERE       Pnumber = Pno
        GROUP BY    Pnumber, Pname;
```

HAVING - Thêm điều kiện cho GROUP BY

Áp dụng điều kiện cho các nhóm.

```
…
GROUP BY …
HAVING   <group_conditions>;
```

GROUP BY … HAVING … - Ví dụ Q26

Query 26: *"Với mỗi dự án trong đó có hơn hai nhân sự tham gia, hãy lấy ra mã số, tên dự án, và số lượng nhân sự trong dự án đó."*

```
Q26:    SELECT      Pnumber, Pname, COUNT (*)
        FROM        PROJECT, WORKS_ON
        WHERE       Pnumber = Pno
        GROUP BY    Pnumber, Pname
        HAVING      COUNT (*) > 2;
```

GROUP BY … HAVING … - Ví dụ Q26 – Kết quả

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

GROUP BY … HAVING … - Ví dụ Q26 – Kết quả

Pname	Pnumber	···	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	···	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

GROUP BY - Ví dụ Q27

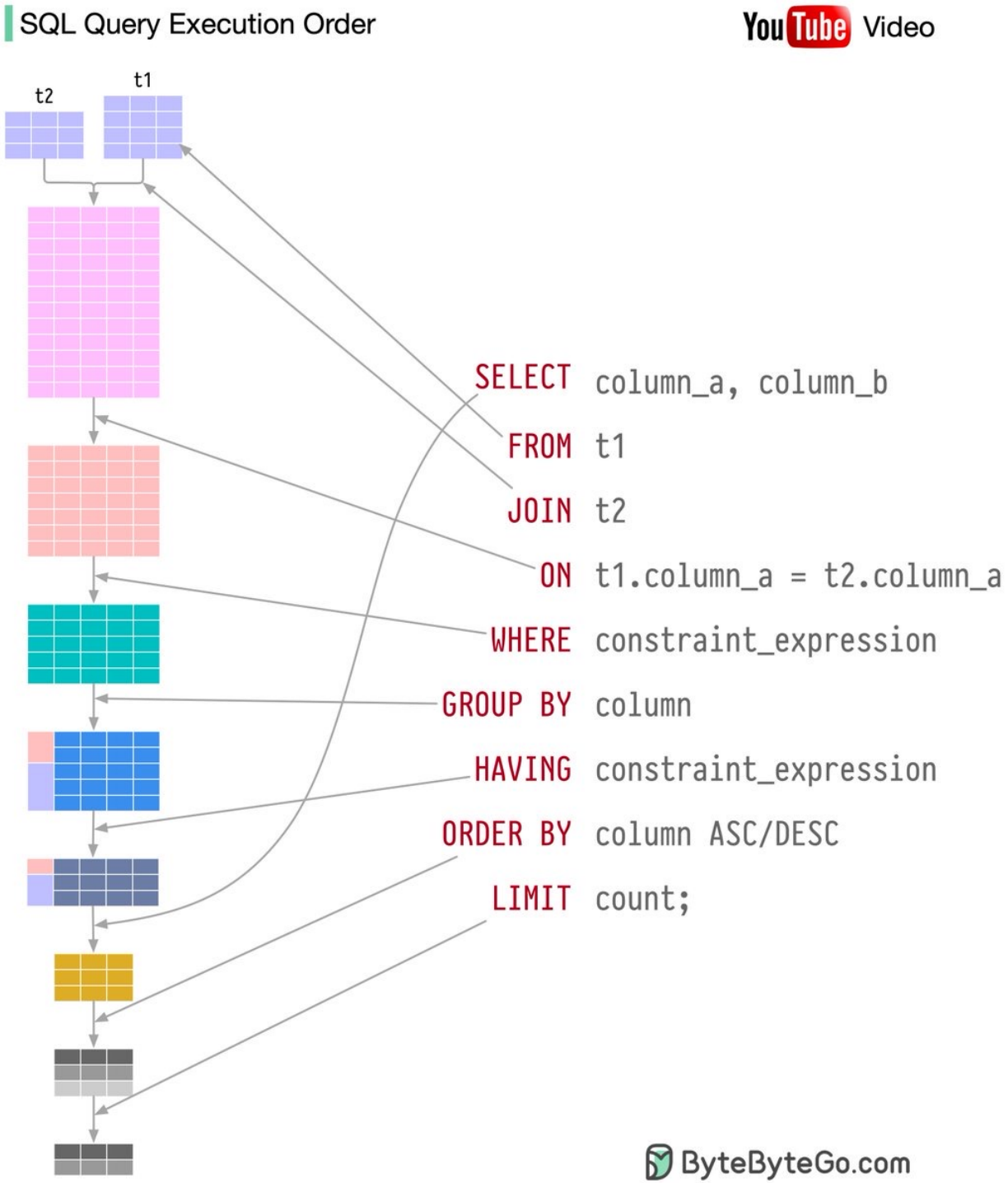
Query 27: *"Với mỗi dự án, hãy lấy ra mã số, tên dự án, và số lượng nhân sự từ phòng ban số 5 đang làm trong dự án đó."*

```
Q27:    SELECT      Pnumber, Pname, COUNT (*)
        FROM        PROJECT, WORKS_ON, EMPLOYEE
        WHERE       Pnumber = Pno AND Ssn = Essn AND Dno = 5
        GROUP BY    Pnumber, Pname;
```

Thứ tự thực hiện kiểm tra điều kiện WHERE và HAVING

1. Mệnh đề WHERE được gọi thực hiện trước, để chọn hoặc kết các bộ dữ liệu,
2. Mệnh đề HAVING thực hiện sau đó, mới thực hiện gom nhóm trên các bộ kết quả của 1.

Lệnh SELECT


SQL Query Execution Order
YouTube Video
t2
t1
SELECT column_a, column_b
FROM t1
JOIN t2
ON t1.column_a = t2.column_a
WHERE constraint_expression
GROUP BY column
HAVING constraint_expression
ORDER BY column ASC/DESC
LIMIT count;
ByteByteGo.com

Ví dụ: Q28

Query 28: *"Với các phòng ban có hơn năm nhân sự, hãy lấy ra mã số của phòng ban và số lượng nhân sự có mức thu nhập hơn $40,000."*

```
SELECT      Dno, COUNT (*)
FROM        EMPLOYEE
WHERE       Salary>40000
GROUP BY    Dno
HAVING      COUNT (*) > 5;
```

```
Q28:  SELECT      Dno, COUNT (*)
      FROM        EMPLOYEE
      WHERE       Salary>40000 AND Dno IN
                  ( SELECT      Dno
                    FROM        EMPLOYEE
                  GROUP BY    Dno
                    HAVING      COUNT (*) > 5)
      GROUP BY    Dno;
```

WITH

Cho phép người dùng định nghĩa một bảng chỉ được sử dụng trong một truy vấn cụ thể.

- Tương tự việc tạo ra một quan sát (view) trên một truy vấn rồi xóa view này đi.

WITH – Ví dụ: Viết lại Q28

```
Q28′:   WITH    BIGDEPTS (Dno) AS
                ( SELECT    Dno
                  FROM      EMPLOYEE
                  GROUP BY  Dno
                  HAVING    COUNT (*) > 5)
        SELECT  Dno, COUNT (*)
        FROM    EMPLOYEE
        WHERE   Salary>40000 AND Dno IN BIGDEPTS
        GROUP BY Dno;
```

CASE

Chia trường hợp theo các điều kiện khác nhau.

<u>Ví dụ</u>: "Lương các nhân sự thuộc phòng ban 5 tăng thêm $2,000, nhân sự thuộc phòng ban 4 tăng thêm $1,500, nhân sự thuộc phòng ban 1 tăng thêm $3,000."

```
U6′:    UPDATE   EMPLOYEE
        SET      Salary =
        CASE     WHEN   Dno = 5    THEN Salary + 2000
                 WHEN   Dno = 4    THEN Salary + 1500
                 WHEN   Dno = 1    THEN Salary + 3000
                 ELSE   Salary + 0 ;
```

Truy vấn lặp trong SQL

```
Q29:    WITH RECURSIVE    SUP_EMP (SupSsn, EmpSsn) AS
            ( SELECT      SupervisorSsn, Ssn
              FROM        EMPLOYEE
              UNION
              SELECT      E.Ssn, S.SupSsn
              FROM        EMPLOYEE AS E, SUP_EMP AS S
              WHERE       E.SupervisorSsn = S.EmpSsn)
        SELECT*
        FROM              SUP_EMP;
```

Đánh giá kết quả (Assertion)
Các cơ chế kích hoạt (Trigger)

Assertion

- Được sử dụng để xác định các kiểu ràng buộc bổ sung, nằm ngoài phạm vi các ràng buộc trong mô hình quan hệ

  (*Nhắc lại: các ràng buộc trong mô hình quan hệ gồm: ràng buộc khóa chính, khóa duy nhất; ràng buộc toàn vẹn thực thể, toàn vẹn tham chiếu*)

- Tạo các ràng buộc cần đánh giá

  Lệnh **CREATE ASSERTION**

Assertion – Ví dụ

Ràng buộc: *"Lương của một nhân sự phải không được nhiều hơn lương của người quản lý của phòng ban mà nhân sự đó đang làm việc."*

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS ( SELECT     *
                     FROM       EMPLOYEE E, EMPLOYEE M,
                                DEPARTMENT D
                     WHERE      E.Salary>M.Salary
                                AND     E.Dno = D.Dnumber
                                AND     D.Mgr_ssn = M.Ssn ) );
```

Trigger

- Được sử dụng để xác định các hành động tự động mà hệ thống CSDL sẽ thực hiện khi các sự kiện và điều kiện nhất định xảy ra.

Lệnh **CREATE TRIGGER**

- Các điều kiện được dùng để theo dõi CSDL.

Trigger – các thành phần

1. Sự kiện (events): các sự kiện được lên kế hoạch kiểm tra,
2. Điều kiện (condition): xác định xem các hành động có được thực hiện hay không,
3. Hành động (action): thường là chuỗi các lệnh SQL, giao tác trên CSDL, hoặc thực thi của chương trình bên ngoài được gọi chạy tự động.

Trigger – Ví dụ

Kiểm tra: *"bất cứ khi nào mức lương của một nhân viên nhiều hơn lương của người quản lý trực tiếp của người ấy trong CSDL COMPANY"*

Thực hiện khi: thêm nhân sự mới, thay đổi mức lương nhân sự, thay đổi người quản lý của một nhân viên.

```
R5: CREATE TRIGGER SALARY_VIOLATION
    BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
       ON EMPLOYEE
    FOR EACH ROW
       WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE
                             WHERE SSN = NEW.SUPERVISOR_SSN ) )
                             INFORM_SUPERVISOR(NEW.Supervisor_ssn,
                             NEW.Ssn );
```

Các chế độ xem trong SQL (Views)

Định nghĩa

- Một chế độ xem (view) là một bảng duy nhất được dẫn ra từ các bảng khác.
  - Các bảng nguồn có thể là bảng cơ sở hoặc là một bảng chế độ xem được xác định trước đó.
- View có thể là một bảng ảo.
  - Hạn chế các thao tác cập nhật trên view
  - Cho phép các thao tác truy vấn
- View được xem là bảng “đệm”, được người dùng tham chiếu thường xuyên trong quá trình tương tác với CSDL.

Tạo View

Lệnh **CREATE VIEW** với các thành phần:

- Tên bảng – tên view,
- Danh sách thuộc tính,
  - Có thể đặt tên mới cho thuộc tính
- Câu truy vấn để lấy nội dung cho view

Tạo View – Ví dụ

V1:
```
CREATE VIEW   WORKS_ON1
AS SELECT     Fname, Lname, Pname, Hours
   FROM       EMPLOYEE, PROJECT, WORKS_ON
   WHERE      Ssn = Essn AND Pno = Pnumber;
```

WORKS_ON1

Fname	Lname	Pname	Hours

V2:
```
CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT     Dname, COUNT(*), SUM(Salary)
   FROM       DEPARTMENT, EMPLOYEE
   WHERE      Dnumber = Dno
   GROUP BY   Dname;
```

DEPT_INFO

Dept_name	No_of_emps	Total_sal

Truy vấn trên View

Query V1: *"Lấy họ tên của tất cả nhân sự tham gia dự án 'ProjectX'."*

QV1: **SELECT** Fname, Lname
FROM WORKS_ON1
WHERE Pname = 'ProductX';

- View đơn giản hóa các truy vấn phức tạp.
- View luôn được tự động cập nhật theo các thay đổi trong CSDL.
- Được sử dụng như cơ chế bảo mật và an toàn cho CSDL.

Hủy View

Khi không sử dụng View nữa, dùng lệnh DROP VIEW để hủy.

Ví dụ:

V1A: **DROP VIEW** WORKS_ON1;

Lựa chọn Cài đặt và Cập nhật View

1. Chuyển view thành truy vấn trên các bảng cơ sở.

QV1 →

```
SELECT   Fname, Lname
FROM     EMPLOYEE, PROJECT, WORKS_ON
WHERE    Ssn = Essn AND Pno = Pnumber
         AND Pname = 'ProductX';
```

- Không hiệu quả do tốn kém khi thực hiện truy vấn, đặc biệt khi có liên tục nhiều truy vấn đến view.

Lựa chọn Cài đặt và Cập nhật View

2. Hiện thực hóa view bằng cách tạo vật lý thành một bảng tạm thời hoặc vĩnh viễn, kèm theo chế độ cập nhật gia tăng (incremental update) phù hợp.

   - Chiến lược cập nhật ngay lập tức cập nhật view ngay khi các bảng cơ sở thay đổi;
   - Chiến lược cập nhật chậm cập nhật view mỗi khi cần bằng một truy vấn view;
   - Chiến lược cập nhật định kỳ cập nhật dạng xem theo định kỳ.
     Truy vấn tới view có thể nhận được kết quả không được cập nhật).

INSERT, DELETE, UPDATE trên view

- Một view từ một bảng cơ sở **có thể được cập nhật** nếu thuộc tính của view có chứa khóa chính của quan hệ cơ sở, và mọi thuộc tính có ràng buộc NOT NULL không dùng thiết lập sử dụng giá trị mặc định.

- View được định nghĩa từ nhiều bảng qua phép kết **không cập nhật được**.

- View được định nghĩa bằng gom nhóm và các hàm tổng **không cập nhật được**.

Ví dụ

```
UV1:    UPDATE WORKS_ON1
        SET     Pname = 'ProductY'
        WHERE   Lname = 'Smith' AND Fname = 'John'
                AND Pname = 'ProductX';
```

```
(a):    UPDATE WORKS_ON
        SET     Pno =       ( SELECT Pnumber
                              FROM   PROJECT
                              WHERE  Pname = 'ProductY' )
        WHERE   Essn IN     ( SELECT Ssn
                              FROM   EMPLOYEE
                              WHERE  Lname = 'Smith' AND Fname = 'John' )
                AND
                Pno =       ( SELECT Pnumber
                              FROM   PROJECT
                              WHERE  Pname = 'ProductX' );
```

```
(b):    UPDATE PROJECT      SET     Pname = 'ProductY'
        WHERE   Pname = 'ProductX';
```

Sử dụng View để đảm bảo an toàn CSDL

View được sử dụng để ẩn đi các thuộc tính/bộ dữ liệu khỏi những người dùng không được phép.

Ví dụ:

```
CREATE VIEW   DEPT5EMP   AS
SELECT        *
FROM          EMPLOYEE
WHERE         Dno = 5;
```

```
CREATE VIEW   BASIC_EMP_DATA   AS
SELECT        Fname, Lname, Address
FROM          EMPLOYEE;
```

Thay đổi lược đồ trong SQL

DROP

Dùng để xóa các phần tử của lược đồ như bảng, miền, kiểu, ràng buộc, hay xóa chính lược đồ.

DROP SCHEMA

DROP TABLE

DROP DOMAIN

DROP TYPE

DROP CONSTRAINT

- Bổ sung tùy chọn CASCADE/RESTRICT để xác lập cách thức thực hiện xóa (hành vi xóa).
- *Chú ý* cú pháp lệnh của các hệ quản trị CSDL sẽ khác nhau một chút.

DROP SCHEMA

Xóa toàn bộ lược đồ.

- **DROP SCHEMA** <schema name> **CASCADE**: thực hiện xóa lần lượt các thành phần trong lược đồ, và xóa lược đồ.

Ví dụ:

DROP SCHEMA COMPANY **CASCADE;**

- **DROP SCHEMA** <schema name> **RESTRICT**: chỉ xóa lược đồ khi các thành phần của nó không còn.

  Người dùng phải thực hiện xóa từng thành phần trong lược đồ, rồi mới thực hiện xóa lược đồ.

DROP TABLE

Xóa tất cả các bản ghi trong bảng, sau đó xóa luôn định nghĩa của bảng khỏi catalog.

- **DROP TABLE** <table name> **CASCADE**: tự động xóa lần lượt các ràng buộc, view, các phần tử có tham chiếu đến <table name> ra khỏi lược đồ; và xóa <table name>.

Ví dụ:

DROP TABLE DEPENDENT **CASCADE**;

- **DROP TABLE** <table name> **RESTRICT**: chỉ xóa được <table name> nếu nó không còn được tham chiếu đến từ trong các ràng buộc (khóa ngoại), view hoặc một thành phần nào khác trong lược đồ.

ALTER TABLE <table name> ADD COLUMN <column name> <type> ;

Lệnh **ALTER** thay đổi định nghĩa của bảng cơ sở hoặc của các thành phần lược đồ khác.

- Thêm cột (thêm thuộc tính)

Vd:
ALTER TABLE COMPANY.EMPLOYEE **ADD COLUMN** Job VARCHAR(12);

- Sau đó còn phải nhập giá trị của thuộc tính mới cho từng bộ dữ liệu.

ALTER TABLE <table name> DROP COLUMN <column name> CASCADE/RESTRICT;

Xóa cột.

- Tùy chọn CASCADE: mọi ràng buộc và views tham chiếu đến cột <column name> sẽ bị xóa tự động khỏi lược đồ, cùng với cột này.
- Tùy chọn RESTRICT: lệnh chỉ thành công nếu không có view hoặc ràng buộc nào (hoặc các phần tử lược đồ khác) tham chiếu đến <column name>

Vd:

ALTER TABLE COMPANY.EMPLOYEE **DROP COLUMN** Addess **CASCADE**;

ALTER <table name> ALTER COLUMN <column name> DROP/SET DEFAULT;

Ví dụ:

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **DROP DEFAULT**;

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **SET DEFAULT** '333445555';

ALTER <table name> DROP/ADD CONSTRAINT ;

Ví dụ:

ALTER TABLE COMPANY.EMPLOYEE **DROP CONSTRAINT** EMPSUPERFK **CASCADE;**

ALTER TABLE Persons **ADD CONSTRAINT** PK_Person **PRIMARY KEY** (ID,LastName)**;**

Bảng tổng hợp cú pháp SQL (1/2)

- **CREATE TABLE** <table name> (<column name> <column type> [<attribute constraint>]
  { , <column name> <column type> [<attribute constraint>] }
  [<table constraint> { , <table constraint> }])
- **DROP TABLE** <table name>
  ALTER TABLE <table name> **ADD** <column name> <column type>
- **SELECT** [**DISTINCT**] <attribute list>
  FROM (<table name> { <alias> } | <joined table>) { , (<table name> { <alias> } | <joined table>) }
  [**WHERE** <condition>]
  [**GROUP BY** <grouping attributes> [**HAVING** <group selection condition>]]
  [**ORDER BY** <column name> [<order>] { , <column name> [<order>] }]

  <attribute list> ::= (* | (<column name> | <function> (([**DISTINCT**] <column name> | *)))
  { , (<column name> | <function> (([**DISTINCT**] <column name> | *)) }))
- <grouping attributes> ::= <column name> { , <column name> }
- <order> ::= (**ASC** | **DESC**)

Bảng tổng hợp cú pháp SQL (2/2)

- **INSERT INTO** <table name> [(<column name> { , <column name> })]
  (**VALUES** (<constant value> , { <constant value> }) { , (<constant value> { , <constant value> }) } | <select statement>)
- **DELETE FROM** <table name>
  [**WHERE** <selection condition>]
- **UPDATE** <table name>
  SET <column name> = <value expression> { , <column name> = <value expression> }
  [**WHERE** <selection condition>]
- **CREATE** [**UNIQUE**] **INDEX** <index name>
  ON <table name> (<column name> [<order>] { , <column name> [<order>] })
  [**CLUSTER**]
- **DROP INDEX** <index name>
- **CREATE VIEW** <view name> [(<column name> { , <column name> })] **AS** <select statement>
- **DROP VIEW** <view name>

Kết thúc phần này

Bài tập chương 7.