HOA SEN
UNIVERSITY

Lecture 4
SQL 3 – Select, Grouping data

Objectives

More Complex SQL Queries:

- Nesting Of Queries
- Joined Relation
- Exists Function
- Aggregate Functions
- Grouping – Having Clause

- Ref.: Chapter 7

ĐẠI HỌC HOA SEN

Set Operations

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in *some versions* of SQL there are set difference (MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to *union compatible relations*; the two relations must have the same attributes and the attributes must appear in the same order

Set Operations - Example

- Find number project in Project or in Works_on relation

```
Select Pnumber From Project
Union
Select Pno From works_on;
```

	PNumber
1	1
2	2
3	3
4	5
5	10
6	20
7	30

Union

	PNo
1	1
2	2
3	3
4	10
5	20
6	30

	PNumber
1	1
2	2
3	3
4	5
5	10
6	20
7	30

Set Operations - Example

- Find number project in Project and in Works_on relation

```
Select Pnumber From Project
Intersect
Select Pno From works_on;
```

	PNumber
1	1
2	2
3	3
4	5
5	10
6	20
7	30

Intersect

	PNo
1	1
2	2
3	3
4	10
5	20
6	30

	PNumber
1	1
2	2
3	3
4	10
5	20
6	30

Set Operations - Example

- Find number project in Project but not in Works_on relation

```
Select Pnumber From Project
 Except
Select Pno From works_on;
```

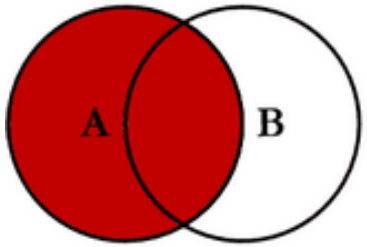
	PNumber
1	1
2	2
3	3
4	5
5	10
6	20
7	30

Except

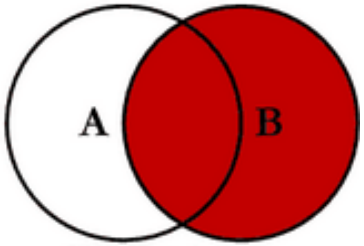
	PNo
1	1
2	2
3	3
4	10
5	20
6	30

	PNumber
1	5

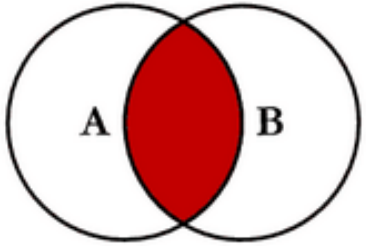
SQL JOINS



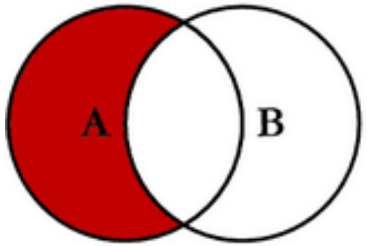

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



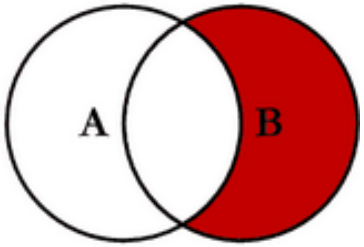

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



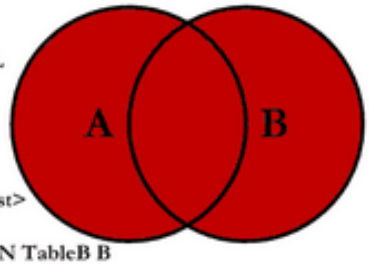

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



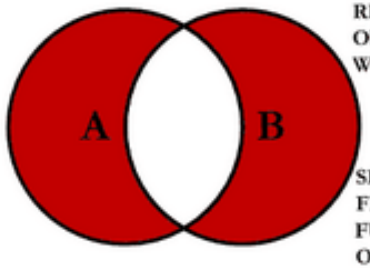

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

ĐẠI HỌC HOA SEN

Set Operations

- SQL has directly incorporated some set operations
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to *union compatible relations*; the two relations must have the same attributes and the attributes must appear in the same order
- To retain all duplicates use the
  - union all,
  - intersect all
  - except all.

Nesting Of Queries - Subqueries

- A complete SELECT query, called a *nested query*, can be specified within the WHERE-clause of another query, called the *outer query*
  - Many of the previous queries can be specified in an alternative form using nesting

- Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT  Fname, Lname, Address
   FROM EMPLOYEE
   WHERE Dno IN
              (SELECT Dnumber
                   FROM DEPARTMENT
                   WHERE Dname='Research');
```

```
Select Fname, Lname, Address
       From   Employee E, Department D
       Where  E.Dno = D.Dnumber And D.Dname='Research';
```

ĐẠI HỌC HOA SEN

Nesting Of Queries (2)

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V
- In general, we can have several levels of nested queries
- A reference to an *unqualified attribute* refers to the relation declared in the *innermost nested query*
- In this example, the nested query is *not correlated* with the outer query

Correlated Nested Queries

- If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query*, the two queries are said to be *correlated*
  - The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query
- Retrieve the name of each employee who has a dependent with the same first name as the employee.

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

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

Correlated Nested Queries (3)

- The original SQL as specified for SYSTEM R also had a **CONTAINS** comparison operator, which is used in conjunction with nested correlated queries
  - This operator was *dropped from the language*, possibly because of the difficulty in implementing it efficiently
  - Most implementations of SQL do not have this operator
  - The CONTAINS operator compares *two sets of values*, and returns TRUE if one set contains all values in the other set
    - Reminiscent of the division operation of algebra

Correlated Nested Queries (4)

- Retrieve the name of each employee who works on all the projects controlled by department number 5.

```
SELECT Fname, Lname
   FROM EMPLOYEE
   WHERE ((SELECT Pno
                  FROM WORKS_ON
                  WHERE Ssn=Essn)
                  CONTAINS
                       (SELECT Pnumber
                       FROM PROJECT
                       WHERE Dnum=5));
```

Set Comparison – "some/any" Clause

- Find names of employees with salary greater than that of some (at least one) employee in the department 5.

```
Select Distinct E.Fname, E.Lname, E.Salary
    From Employee as E, Employee as E5
    Where E.salary > E5.salary and E5.DNo = 5;
```

- Same query using > **some** clause

```
Select Fname, LName, Salary
    From Employee
    Where salary > Some (Select Salary From Employee
                         Where DNo = 5);
```

	FName	LName	Salary	DNo
1	John	Smith	30000.00	5
2	Franklin	Wong	40000.00	5
3	Joyce	English	25000.00	5
4	Ramesh	Narayan	38000.00	5
5	James	Borg	55000.00	1
6	Jennifer	Wallace	43000.00	4
7	Ahmad	Jabbar	25000.00	4
8	Alicia	Zelaya	25000.00	4

	Fname	LName	Salary
1	John	Smith	30000.00
2	Franklin	Wong	40000.00
3	Ramesh	Narayan	38000.00
4	James	Borg	55000.00
5	Jennifer	Wallace	43000.00

Set Comparison – "all" Clause

- Find the names of all employees whose salary is greater than the salary of all employees in the department 5.

```
Select Fname, LName, Salary
     From Employee
     Where salary >
          All (Select Salary From Employee
                    Where DNo = 5);
```

	FName	LName	Salary	DNo
1	John	Smith	30000.00	5
2	Franklin	Wong	40000.00	5
3	Joyce	English	25000.00	5
4	Ramesh	Narayan	38000.00	5
5	James	Borg	55000.00	1
6	Jennifer	Wallace	43000.00	4
7	Ahmad	Jabbar	25000.00	4
8	Alicia	Zelaya	25000.00	4

	Fname	LName	Salary
1	James	Borg	55000.00
2	Jennifer	Wallace	43000.00

Test for Empty Relations

- The **exists** construct returns the value **true** if the argument subquery is nonempty.
  - **exists** $r \Leftrightarrow r \neq \emptyset$
  - **not exists** $r \Leftrightarrow r = \emptyset$

•Retrieve the name of all employees who work for the 'Research' department.

```
Select Fname, Lname
  From Employee
  Where Exists (Select * From Department
                Where DNo = Dnumber And Dname='Research' );
```

```
Select   Fname, Lname
  From   Employee
  Where  Dno IN
             (Select Dnumber From Department
             Where   Dname='Research' );
```

ĐẠI HỌC
HOA SEN

- List the names of employees who have a dependent with the same first name and sex as themselves.

```
Select FName, LName
    From Employee E
    Where Exists
        (Select *  From Dependent D
            Where E.SSN = D.ESSN And
            E.FName = D.Dependent_Name And
            E.Sex = D.Sex);
```

```
Select E.FName, E.Lname
    From Employee E, Dependent D
    Where E.SSN = D.ESSN And
        E.FName = D.Dependent_Name And
        E.Sex = D.Sex;
```

Use of "not exists" Clause

- Find all employees who have taken all projects managed in the department 5.

```
Select E.FName, E.Lname
    From employee E
    Where not exists
    ( (select Pnumber from project where DNum = 5)
        except
    (Select W.Pno From works_on W where W.ESSN = E.SSN));
```

  - First nested query lists all projects managed in department 5.
  - Second nested query lists all projects a particular employee works on

- Note
  - $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
  - Cannot write this query using = all and its variants

ĐẠI HỌC HOA SEN

- Query: Give name of employees participating in projects managed by department 5.

	SSN	FName	LName
1	123456789	John	Smith
2	333445555	Franklin	Wong
3	453453453	Joyce	English
4	666884444	Ramesh	Narayan
5	888665555	James	Borg
6	987654321	Jennifer	Wallace
7	987987987	Ahmad	Jabbar
8	999887777	Alicia	Zelaya

	PName	PNumber	PLocation	DNum
1	ProductX	1	Bellaire	5
2	ProductY	2	Sugarland	5
3	ProductZ	3	Houston	5
4	Computerization	10	Stafford	4
5	Reorganization	20	Houston	1
6	Newbenefits	30	Stafford	4

	ESSN	PNo	Hours
1	123456789	1	32.5
2	123456789	2	7.5
3	123456789	3	10.0
4	333445555	2	10.0
5	333445555	3	10.0
6	333445555	10	10.0
7	333445555	20	10.0
8	453453453	1	20.0
9	453453453	2	20.0
10	453453453	3	20.0
11	666884444	3	40.0
12	888665555	20	NULL
13	987654321	20	15.0
14	987654321	30	20.0
15	987987987	10	35.0
16	987987987	30	5.0
17	999887777	10	10.0
18	999887777	30	30.0

	FName	LName
1	John	Smith
2	Joyce	English

ĐẠI HỌC
HOA SEN

- Or:

```
Select A.FName, A.Lname
From employee A
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 A.SSN = C.ESSN And C.PNo =
                        B.PNo));
```

ĐẠI HỌC HOA SEN

Joined Relations Feature in SQL2

- Can specify a "joined relation" in the FROM-clause
  - Looks like any other relation but is the result of a join
  - Allows the user to specify different types of joins (regular "theta" JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, etc)

ĐẠI HỌC HOA SEN

Joined Relations Feature in SQL2 (2)

- Examples:

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
 FROM EMPLOYEE E S
 WHERE E.super_ssn=S.ssn;
```

- can be written as:

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
 FROM EMPLOYEE E LEFT OUTER JOIN  EMPLOYEE S
      ON E.Super_ssn=S.Ssn);
```

Joined Relations Feature in SQL2 (3)

- Examples:

```
SELECT Fname, Lname, Address
  FROM EMPLOYEE, DEPARTMENT
  WHERE Dname='Research' AND Dnumber=Dno;
```

- could be written as:

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

- or as:

```
SELECT Fname, Lname, Address
  FROM (EMPLOYEE NATURAL JOIN EPARTMENT
        AS DEPT(Dname, Dno, Mssn,Msdate)
  WHERE Dname='Research';
```

Joined Relations Feature in SQL2 (4)

- Another Example:

  For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
SELECT Pnumber, Dnum, Lname, Bdate, Address
    FROM PROJECT, DEPARTMENT, EMPLOYEE
    WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
              AND Plocation='Stafford';
```

Could be written as follows; this illustrates multiple joins in the joined tables

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

ĐẠI HỌC HOA SEN

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values.
- Three forms of outer join:
  - left outer join
  - right outer join
  - full outer join

Outer Join Examples

- Relation Course

	CourseID	CourseName	DNo	Credit
1	CS110	Basic Programming	SC	3
2	CS210	Data Structured	SC	3
3	BA250	Business Management	BA	3

- Relation GradeReport

	CourseID	Grade
1	CS110	B
2	CS210	A
3	BA140	B

- Observe that

  Course information is missing **BA140**

  GradeReport information is missing **BA250**

ĐẠI HỌC HOA SEN

- Course **left outer join** GradeReport
  - In relational algebra: Course ⟕ GradeReport
  - In SQL:

```
Select *
From Course C Left Outer Join GradeReport G On
C.CourseID = G.CourseID;
```

	CourseID	CourseName	DNo	Credit	CourseID	Grade
1	CS110	Basic Programming	SC	3	CS110	B
2	CS210	Data Structured	SC	3	CS210	A
3	BA250	Business Management	BA	3	NULL	NULL

ĐẠI HỌC HOA SEN

- Course **Right outer join** GradeReport
  - In relational algebra: Course ⟖ GradeReport
  - In SQL:

```
Select *
From Course C Right Outer Join GradeReport G On
C.CourseID = G.CourseID;
```

	CourseID	CourseName	DNo	Credit	CourseID	Grade
1	CS110	Basic Programming	SC	3	CS110	B
2	CS210	Data Structured	SC	3	CS210	A
3	NULL	NULL	NULL	NULL	BA140	B

ĐẠI HỌC HOA SEN

- Course **full outer join** GradeReport
  - In relational algebra: Course ⟗ GradeReport
  - In SQL:

```
Select *
     From Course C Full Outer Join GradeReport G
On C.CourseID = G.CourseID;
```

	CourseID	CourseName	DNo	Credit	CourseID	Grade
1	CS110	Basic Programming	SC	3	CS110	B
2	CS210	Data Structured	SC	3	CS210	A
3	BA250	Business Management	BA	3	NULL	NULL
4	NULL	NULL	NULL	NULL	BA140	B

Aggregate Functions

- Include **COUNT, SUM, MAX, MIN, and AVG**

- Find the maximum salary, the minimum salary, the average salary and numbers of employees.

```
SELECT MAX(Salary), MIN(Salary), AVG(Salary), Count(*)
  FROM   EMPLOYEE;
```

	SSN	Salary	DNo
1	123456789	30000.00	5
2	333445555	40000.00	5
3	453453453	25000.00	5
4	666884444	38000.00	5
5	888665555	55000.00	1
6	987654321	43000.00	4
7	987987987	25000.00	4
8	999887777	25000.00	4

	MaxSal	MinSAl	AvgSal	CountEmp
1	55000.00	25000.00	35125.000000	8

- Some SQL implementations *may not allow more than one function* in the SELECT-clause

Aggregate Functions (2)

- Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
SELECT MAX(Salary), MIN(Salary), AVG(Salary)
FROM   EMPLOYEE, DEPARTMENT
WHERE  Dno=Dnumber AND Dname='Research';
```

EMPLOYEE

Salary	Super_ssn	Dno
30000	333445555	5
40000	888665555	5
25000	987654321	4
43000	888665555	4
38000	333445555	5
25000	333445555	5
25000	987654321	4
55000	NULL	1

DEPARTMENT

Dname	Dnumber
Research	5
Administration	4
Headquarters	1

Result?

Aggregate Functions (3)

- Retrieve the total number of employees in the company.

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

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Result?

Aggregate Functions (3)

- The number of employees in the 'Research' department.

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

EMPLOYEE

Fname	Minit	Lname	Dno
John	B	Smith	5
Franklin	T	Wong	5
Alicia	J	Zelaya	4
Jennifer	S	Wallace	4
Ramesh	K	Narayan	5
Joyce	A	English	5
Ahmad	V	Jabbar	4
James	E	Borg	1

DEPARTMENT

Dname	Dnumber
Research	5
Administration	4
Headquarters	1

Result?

ĐẠI HỌC HOA SEN

Grouping

- In many cases, we want to apply the aggregate functions to *subgroups of tuples* in a relation
- Each subgroup of tuples consists of the set of tuples that have the *same value* for the *grouping attribute(s)*
- The function is applied to each subgroup independently
- SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

Grouping (2)

- For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT Dno, COUNT(*), Sum(Salary)
   FROM EMPLOYEE
   GROUP BY Dno;
```

Dno	SSN	Salary
5	123456789	30000.00
5	333445555	40000.00
5	453453453	25000.00
5	666884444	38000.00
1	888665555	55000.00
4	987654321	43000.00
4	987987987	25000.00
4	999887777	25000.00

Dno	(No column name)	(No column name)
1	1	55000.00
4	3	93000.00
5	4	133000.00

Grouping (3)

- For each project, retrieve the project number, project name, and the number of employees who work on that project.

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

PName	PNumber	PLocation	DNum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

ESSN	PNo	Hours
123456789	1	32.5
123456789	2	7.5
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
453453453	1	20.0
453453453	2	20.0
666884444	3	40.0
888665555	20	NU...
987654321	20	15.0
987654321	30	20.0
987987987	10	35.0
987987987	30	5.0
999887777	10	10.0
999887777	30	30.0

Pnumber	Pname	(No column name)
1	ProductX	2
2	ProductY	3
3	ProductZ	2
10	Computerization	3
20	Reorganization	3
30	Newbenefits	3

The HAVING-clause

- Sometimes we want to retrieve the values of these functions for only those *groups that satisfy certain conditions*

- The **HAVING**-clause is used for specifying a selection condition on groups (rather than on individual tuples)

The HAVING-clause (2)

- For each project *on which more than two employees work*, retrieve the project number, project name, and the number of employees who work on that project.

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

Pnumber	Pname	(No column name)
1	ProductX	2
2	ProductY	3
3	ProductZ	2
10	Computerization	3
20	Reorganization	3
30	Newbenefits	3

Pnumber	Pname	(No column name)
2	ProductY	3
10	Computerization	3
20	Reorganization	3
30	Newbenefits	3

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

```
SELECT      <attribute list>
   FROM     <table list>
   [WHERE   <condition>]
       [GROUP BY <grouping attribute(s)>
       [HAVING <group condition>]]
   [ORDER BY  <attribute list>];
```

ĐẠI HỌC HOA SEN

Summary of SQL Queries (2)

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries

- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups

- ORDER BY specifies an order for displaying the result of a query
  - A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

ĐẠI HỌC HOA SEN

Views in SQL

- A view is a “virtual” table that is derived from other tables
- Allows for limited update operations
  - Since the table may not physically be stored
- Allows full query operations
- A convenience for expressing certain operations

Specification of Views

- SQL command: **`CREATE VIEW`**
  - a table (view) name
  - a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
  - a query to specify the table contents

SQL Views: An Example

- Specify a different WORKS_ON table

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

- Using a Virtual Table
  - We can specify SQL queries on a newly create table (view):

```
SELECT Fname, Lname
        FROM v_WORKSON
        WHERE Pname='Seena';
```

Drop view

- When no longer needed, a view can be dropped:

```
DROP v_WORKSON;
```

ĐẠI HỌC HOA SEN

Efficient View Implementation

- Query modification:
  - Present the view query in terms of a query on the underlying base tables
- Disadvantage:
  - Inefficient for views defined via complex queries
    - Especially if additional queries are to be applied to the view within a short time period

ĐẠI HỌC HOA SEN

Efficient View Implementation

- View materialization:
  - Involves physically creating and keeping a temporary table
- Assumption:
  - Other queries on the view will follow
- Concerns:
  - Maintaining correspondence between the base table and the view when the base table is updated
- Strategy:
  - Incremental update

ĐẠI HỌC HOA SEN

Update Views

- Update on a single view without aggregate operations:
  - Update may map to an update on the underlying base table
- Views involving joins:
  - An update *may* map to an update on the underlying base relations
    - Not always possible

Un-updatable Views

- Views defined using groups and aggregate functions are not updateable
- Views defined on multiple tables using joins are generally not updateable
- **`WITH CHECK OPTION`**: must be added to the definition of a view if the view is to be updated
  - To allow check for updatability and to plan for an execution strategy

ĐẠI HỌC
HOA SEN

Q & A