Chapter 5
Structured Query Language

Content

- Introduction
- Data definition
- Data manipulation
  - Query
  - Update
- View definition
- Index

Introduction

- Relational algebra language
  - How to execute the query operations (in what order)
  - Difficult for users
- SQL (Structured Query Language)
  - High level declarative language interface
  - The user only specifies what the result is to be
  - Developed at IBM Research (1970s)
  - Also pronounced SEQUEL
  - Expanded to a standard by ANSI
    - SQL1: SQL-86
    - SQL2: SQL-92
    - SQL3: SQL-99, SQL-2003, SQL-2006, SQL-2008

Introduction

- SQL includes
  - Data definition
  - Data manipulation
  - View definition
  - Integrity constraint

  - Security and authorization
  - Transaction control

  - Rules for embedding SQL into programming languages

Content

- Introduction
- **Data definition**
  - Data type
  - Data definition commands
- Data manipulation
- View defintion
- Index

Data definition

- Describes the structure of information in the DB
  - Schema for the relation
  - Domain of each attribute
  - Integrity constraint
  - Index on each relation
- Consists of
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE
  - CREATE DOMAIN
  - CREATE DATABASE
  - …

Data type

- Numeric
  - INTEGER
  - SMALLINT
  - NUMERIC, NUMERIC(p), NUMERIC(p,s)
  - DECIMAL, DECIMAL(p), DECIMAL(p,s)
  - REAL
  - DOUBLE PRECISION
  - FLOAT, FLOAT(p)

Data type

- Character string
  - CHARACTER, CHARACTER(n)
  - CHARACTER VARYING(x)
- Bit string
  - BIT, BIT(x)
  - BIT VARYING(x)
- Datetime
  - DATE
  - TIME
  - TIMESTAMP

Create table command

- Define a new relation by giving
  - A name
  - Attributes
    - Names
    - Data types
    - Integrity constraints on attributes
- Syntax

```
CREATE TABLE <Table_name> (
        <Column_name> <Data_type> [<Constraint>],
        <Column_name> <Data_type> [<Constraint>],
        ...
        [<Constraint>]
)
```

Example

```
CREATE TABLE EMPLOYEE (
        SSN CHAR(9),
        LNAME VARCHAR(10),
        MNAME VARCHAR(20),
        FNAME VARCHAR(10),
        BDATE DATETIME,
        ADDRESS VARCHAR(50),
        SEX CHAR(3),
        SALARY INT,
        SUPERSSN CHAR(9),
        DNO INT
)
```

Create table command

- Basic <Constraint>
  - NOT NULL
  - NULL
  - UNIQUE
  - DEFAULT
  - PRIMARY KEY
  - FOREIGN KEY / REFERENCES
  - CHECK

- Give a name to constraints

CONSTRAINT <Constraint_name> <Constraint>

Example – Constraint

```
CREATE TABLE EMPLOYEE (
        LNAME VARCHAR(10) NOT NULL,
        MNAME VARCHAR(20) NOT NULL,
        FNAME VARCHAR(10) NOT NULL,
        SSN CHAR(9) PRIMARY KEY,
        BDATE DATETIME,
        ADDRESS VARCHAR(50),
        SEX CHAR(3) CHECK (SEX IN ( 'Nam' , 'Nu' )),
        SALARY INT DEFAULT (10000),
        SUPERSSN CHAR(9),
        DNO INT
)
```

Example – Constraint

```
CREATE TABLE DEPARTMENT (
        DNAME VARCHAR(20) UNIQUE,
        DNUMBER INT NOT NULL,
        MGRSSN CHAR(9),
        MGRSTARTDATE DATETIME DEFAULT (GETDATE())
)

CREATE TABLE WORKS_ON (
        SSN CHAR(9) FOREIGN KEY (SSN)
                REFERENCES EMPLOYEE(SSN),
        PNO INT REFERENCES PROJECT(PNumber),
        HOURS DECIMAL(3,1)
)
```

Example – Constraint name

```
CREATE TABLE EMPLOYEE (
        LNAME VARCHAR(10) CONSTRAINT EM_LNAME_NN NOT NULL,
        MNAME VARCHAR(20) NOT NULL,
        FNAME VARCHAR(10) NOT NULL,
        SSN CHAR(9) CONSTRAINT EM_SSN_PK PRIMARY KEY,
        BDATE DATETIME,
        ADDRESS VARCHAR(50),
        SEX CHAR(3) CONSTRAINT EM_Sex_CHK
                CHECK (SEX IN ( 'Nam' , 'Nu' )),
        SALARY INT CONSTRAINT EM_Salary_DF DEFAULT (10000),
        SUPERSSN CHAR(9),
        DNO INT
)
```

Example – Constraint name

```
CREATE TABLE WORKS_ON (
        SSN CHAR(9),
        PNO INT,
        HOURS DECIMAL(3,1),
        CONSTRAINT WO_SSN_PNo_PK PRIMARY KEY (SSN, PNO),
        CONSTRAINT WO_SSN_FK FOREIGN KEY (SSN)
                REFERENCES EMPLOYEE(SSN),
        CONSTRAINT WO_PNo_FK FOREIGN KEY (PNO)
                REFERENCES PROJECT(PNUMBER)
)
```

Alter table command

- Is used for modification
  - The structure of tables
  - Integrity constraints

- Columns

```
ALTER TABLE <Table_name> ADD COLUMN
        <Column_name> <Data_type> [<Constraint>]
```

```
ALTER TABLE <Table_name> DROP COLUMN <Column_name>
```

```
ALTER TABLE <Table_name> ALTER COLUMN
        <Column_name> <New_data_type>
```

Alter table command

- Constraints

```
ALTER TABLE <Table_name> ADD
        CONSTRAINT <Constraint_name> <Constraint>,
        CONSTRAINT <Constraint_name> <Constraint>,
        ...
```

```
ALTER TABLE <Table_name> DROP <Constraint_name>
```

Example

```
ALTER TABLE EMPLOYEE ADD
        JOBTITLE CHAR(20)

ALTER TABLE EMPLOYEE DROP COLUMN JOBTITLE

ALTER TABLE EMPLOYEE ALTER COLUMN
        JOBTITLE CHAR(50)
```

Example

```
CREATE TABLE DEPARTMENT (
        DNAME VARCHAR(20),
        DNUMBER INT NOT NULL,
        MGRSSN CHAR(9),
        MGRSTARTDATE DATETIME
)
ALTER TABLE DEPARTMENT ADD
        CONSTRAINT DE_DNumber_PK PRIMARY KEY (DNUMBER),
        CONSTRAINT DE_MgrSSN_FK FOREIGN KEY (MGRSSN)
                REFERENCES EMPLOYEE(SSN),
        CONSTRAINT DE_MgrStartDate_DF DEFAULT (GETDATE())
                FOR (MGRSTARTDATE),
        CONSTRAINT DE_DName_UN UNIQUE (DNAME)
```

Drop table command

- Is used for deleting the structure of tables
  - All the data in a table are also deleted

- Syntax

```
DROP TABLE <Table_name>
```

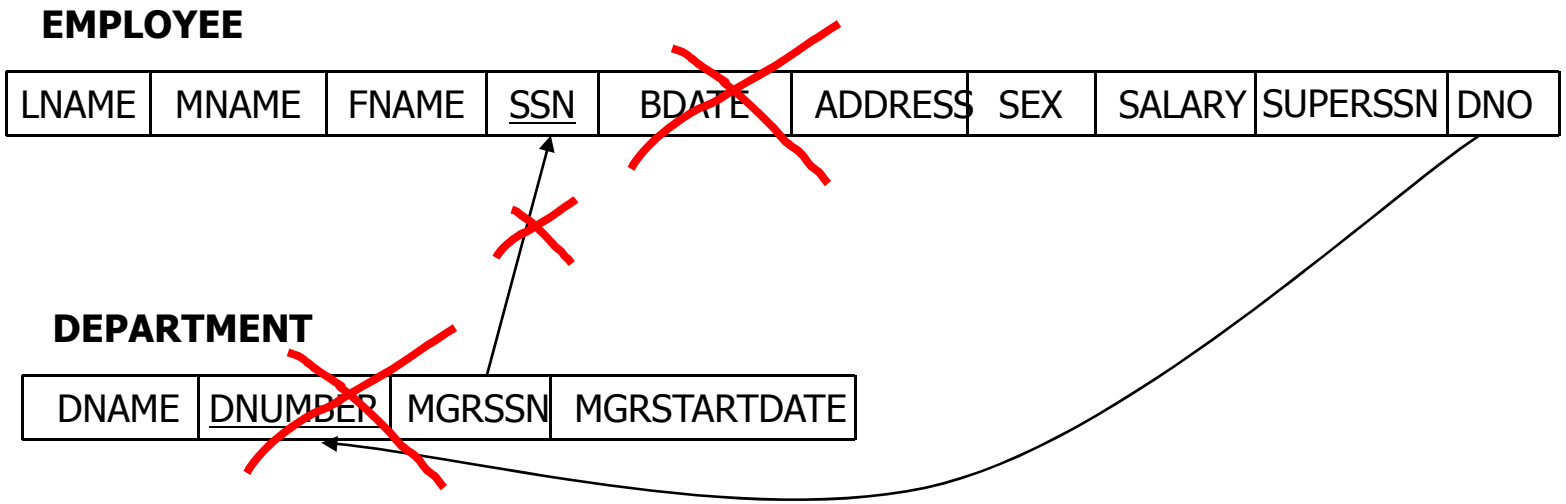
- Example

```
DROP TABLE EMPLOYEE

DROP TABLE DEPARTMENT

DROP TABLE WORKS_ON
```

Example


EMPLOYEE
LNAME
MNAME
FNAME
SSN
BDATE
ADDRESS
SEX
SALARY
SUPERSSN
DNO
DEPARTMENT
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE

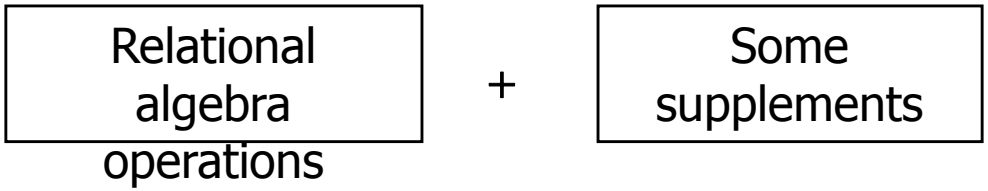
Content

- Introduction
- Data definition
- **Data manipulation**
  - Basic queries
  - Set, set/multiset comparison and nested queries
  - Aggregate functions and grouping
  - Others
- Data update
- View definition
- Index

Query

- Data manipulation language is used for retrieving information from a database
  - These tuples often satisfy a certain condition

- Based on



  - Allow a table to have two or more tuples that are identical in all their attribute values
  - Not a set of tuples, but a multiset or bag

Basic query

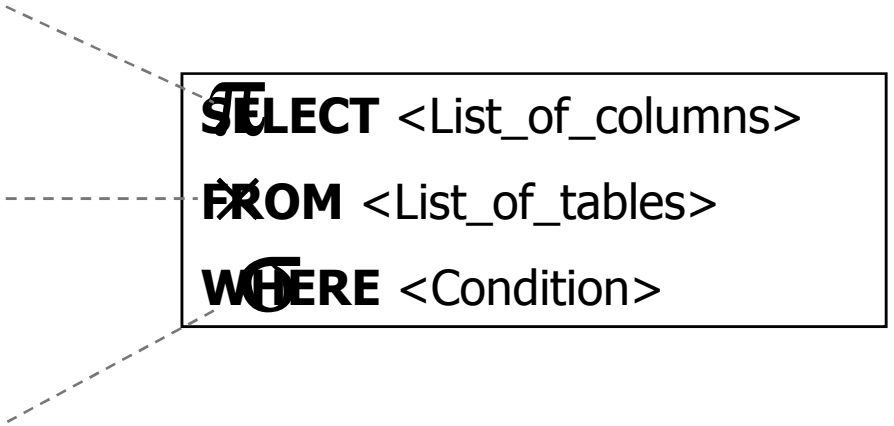
- Is formed of the three clauses

```
SELECT <List_of_columns>
FROM   <List_of_tables>
[WHERE] <Condition>
```

  - <List_of_columns>
    - Column names showed in the result of the query
  - <List_of_tables>
    - Table names required to process the query
  - <Condition>
    - Boolean expression that identifies the rows to be retrieved
    - Expression's connection : AND, OR, and NOT
    - Operations: < , > , <=, >=, <>, =, LIKE and BETWEEN

Basic query

- SQL and Relational Algebra



SELECT L

FROM R

WHERE C

$\Pi_L(\sigma_C(R))$

Example

The entire tuple is produced

```
SELECT  *
FROM    EMPLOYEE
WHERE   DNO=5
```

SSN	LName	MName	FName	BirthDate	Address	Sex	Salary	SuperSSN	DNo
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	5
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	5

$$\sigma_{DNO=5}(EMPLOYEE)$$

SELECT clause

```
SELECT  SSN, LNAME, MNAME, FNAME
FROM    EMPLOYEE
WHERE   DNO=5 AND SEX= 'Nam'
```

SSN	LNAME	MNAME	FNAME
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

$$\pi_{\text{SSN, LNAME, MNAME, FNAME}}(\sigma_{\text{DNO=5} \wedge \text{SEX= 'Nam'}}(\text{EMPLOYEE}))$$

SELECT clause

Alias name

```
SELECT  SSN, LNAME AS 'Last Name', MNAME AS 'Middle Name',
        FNAME AS 'First Name'
FROM    EMPLOYEE
WHERE   DNO=5 AND SEX= 'Nam'
```

SSN	LAST NAME	MIDDLE NAME	FIRST NAME
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

$$R1 \leftarrow (\pi_{SSN,\ LNAME,\ MNAME,\ FNAME}(\sigma_{DNO=5 \wedge SEX=\ 'Nam'}(EMPLOYEE)))$$

$$\rho_{RESULT(SSN,\ LAST\ NAME,\ MIDDLE\ NAME,\ FIRST\ NAME}(R1)$$

SELECT clause

Extension

```
SELECT  SSN, LNAME + ' ' + MNAME+ ' ' + FNAME AS 'Full Name'
FROM    EMPLOYEE
WHERE   DNO=5 AND SEX= 'Nam'
```

SSN	Full Name
333445555	Nguyen Thanh Tung
987987987	Nguyen Manh Hung

$$\rho_{SSN,\ Full\ Name}(\pi_{SSN,LNAME||MNAME||FNAME}(\sigma_{DNO=5 \wedge SEX='Nam'}(EMPLOYEE)))$$

SELECT clause

Extension

```
SELECT  SSN, Salary*1.1 AS '10%SalaryIncrease'
FROM    EMPLOYEE
WHERE   DNo=5 AND Sex= 'Nam'
```

SSN	10%SalaryIncrease
333445555	33000
987987987	27500

$$\rho_{SSN,\ 10\%SalaryIncrease}(\pi_{SSN,\ Salary*1.1}(\sigma_{DNo=5 \wedge Sex= \text{'Nam'}}(EMPLOYEE)))$$

SELECT clause

Duplicate tuples are eliminated

```
SELECT DISTINCT Salary
FROM   EMPLOYEE
WHERE  DNo=5 AND Sex= 'Nam'
```

Salary
30000
25000
38000
38000

- Cost
- Users want to see all tuples

Example

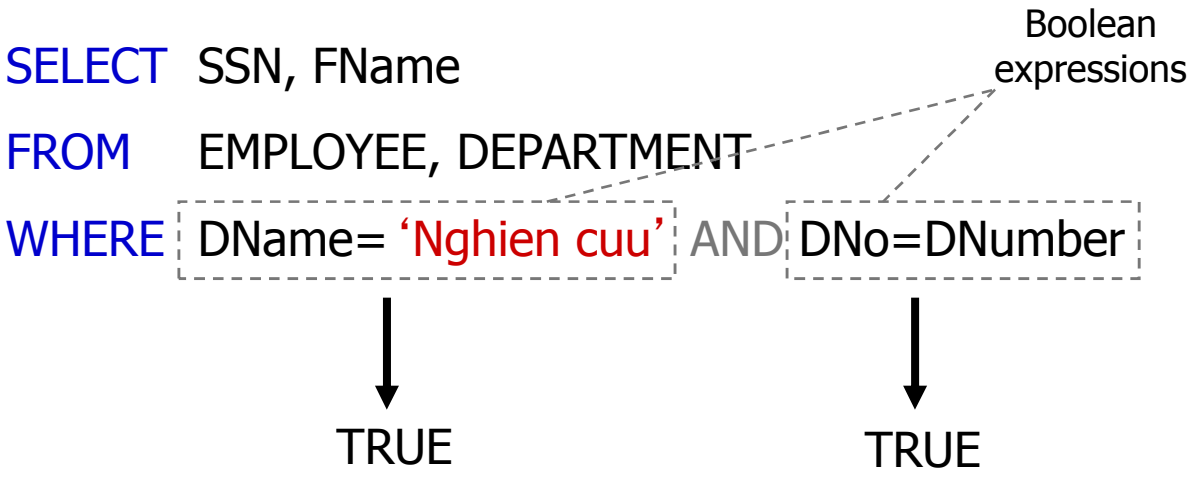
- Find the SSN and first name of employees who work for the department ‘Nghien cuu’

$$R1 \leftarrow EMPLOYEE \bowtie_{DNo=DNumber} DEPARTMENT$$

$$RESULT \leftarrow \pi_{SSN,\ FName}(\sigma_{DName=\text{‘Nghien cuu’}}(R1))$$

```
SELECT    SSN, FName
FROM      EMPLOYEE, DEPARTMENT
WHERE     DName=‘Nghien cuu’  AND  DNo=DNumber
```

WHERE clause



WHERE clause

Priority

```
SELECT  SSN, FName
FROM    EMPLOYEE, DEPARTMENT
WHERE   (DName= 'Nghien cuu'  OR DName= 'Quan ly' ) AND DNo=DNumber
```

WHERE clause

BETWEEN

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   Salary>=20000 AND Salary=<30000
```

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   Salary BETWEEN 20000 AND 30000
```

WHERE clause

NOT BETWEEN

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   Salary NOT BETWEEN 20000 AND 30000
```

WHERE clause

LIKE

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   Address LIKE  'Nguyen _ _ _ _'
```

Arbitrary characters

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   Address LIKE  'Nguyen %'
```

Arbitrary strings

WHERE clause

NOT LIKE

```
SELECT  SSN, FName
FROM    EMPOYEE
WHERE   LName LIKE  'Nguyen'
```

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   LName NOT LIKE  'Nguyen'
```

WHERE clause

ESCAPE

```
SELECT  SSN, FName
FROM    EMPOYEE
WHERE   Address  LIKE  '123x/%Nguyen'  ESCAPE  'x'
```

WHERE clause

Datetime

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   BDate BETWEEN '1955-12-08' AND '1966-07-19'
```

'1955-12-08' YYYY-MM-DD

'12/08/1955' MM/DD/YYYY

'December 8, 1955'

'17:30:00' HH:MI:SS

'05:30 PM'

'1955-12-08 17:30:00'

WHERE clause

NULL

- SQL allows attributes to have value NULL
  - Value unknown
  - Value inapplicable
  - Value withheld
- Operation on a NULL and any value, the result is NULL
  - x has a value NULL
  - x + 3 is also NULL
  - x + 3 is not a legal SQL expression
- Comparison on a NULL value and any value, the result is UNKNOWN
  - The value of x = 3 is UNKNOWN
  - The comparison x = 3 is not correct SQL

WHERE clause

NULL

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   SuperSSN IS NULL
```

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   SuperSSN IS NOT NULL
```

WHERE clause

UNKNOWN

- The comparison involving NULL will result the three-value logic
  - True (1)
  - False (0)
  - Unknown (1/2)

- Logical operator
  - x and y (Minimum value)
  - x or y (Maximum value)
  - not x (1-x)

- The condition in WHERE clauses will result ***False*** if tuples with ***Unknown*** value

FROM clause

Unspecified WHERE clause

```
SELECT  SSN, DNumber
FROM    EMPLOYEE, DEPARTMENT
WHERE   TRUE
```

SSN	DNumber
333445555	1
333445555	4
333445555	5
987987987	1
987987987	4
987987987	5
…	…

FROM clause

Alias name

```
SELECT DNname, DLocation
FROM DEPARTMENTS AS DEPT, DEPT_LOCATIONS AS DL
WHERE DEPT.DNumber = DL.DNumber
```

```
SELECT  FName, BDate, Dependent_Name, BDate
FROM    EMPLOYEE, DEPENDENT DE
WHERE   SSN=ESSN
```

Example 1

- For each project locating in “Ha Noi”, find its number, the department number that controls it, the last and first name of the manager, as well as his/her birth date and address

Example 2

- Find the last and first name of employees who work for the department 5 and work on the project “San pham X” with working hours are larger than 10

Example 3

- Find the first and last name of employees and their supervisor

Example 4

- Find the last and first name of employees who are supervised directly by “Nguyen Thanh Tung”

ORDER BY clause

- Is used for presenting a query in sorted order
- Syntax

```
SELECT <List_of_columns>
FROM <List_of_tables>
WHERE <Conditions>
ORDER BY<List_of columns>
```

  - ASC (default)
  - DESC

Example

```
SELECT  ESSN, PNo
FROM    WORKS_ON
ORDER BY ESSN DESC, PNo
```

ESSN	PNo
999887777	10
999887777	30
987987987	10
987987987	30
987654321	10
987654321	20
987654321	30

Content

- Introduction
- Data definition
- Data manipulation
  - Basic queries
  - **Set, set/multiset comparison and nested queries**
  - Aggregate functions and grouping
  - Others
- Data update
- View definition
- Index

Set operations in SQL

- SQL has implemented set operators
  - UNION
  - INTERSECT
  - EXCEPT

- The result is a set
  - Eliminate identical tuples
  - To keep identical tuples
    - UNION ALL
    - INTERSECT ALL
    - EXCEPT ALL

Set operations in SQL

- Syntax

```
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
UNION [ALL]
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
```

```
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
INTERSECT [ALL]
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
```

```
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
EXCEPT [ALL]
SELECT <Column_list> FROM <Table_list> WHERE <Condition>
```

Example 5

- Find the project numbers that have
  - Either employees with the last name ‘Nguyen’,
  - Or been controlled by the department whose manager has the last name ‘Nguyen’

```
SELECT  PNo
FROM    EMPLOYEE, WORKS_ON
WHERE   SSN=ESSN AND LName= ‘Nguyen’
UNION
SELECT  PNumber
FROM    EMPLOYEE, DEPARTMENT, PROJECT
WHERE   SSN=MgrSSN AND DNumber=DNum AND LName= ‘Nguyen’
```

Example 6

- Find employees who have dependents with the same name and sex

```
SELECT  FName, Sex, SSN FROM EMPLOYEE
INTERSECT
SELECT  Dependent_Name, Sex, ESSN FROM DEPENDENT
```

```
SELECT  EM.*
FROM    EMPLOYEE EM, DEPENDENT DE
WHERE   EM.SSN=DE.ESSN
AND EM.FName=DE.Dependent_name
AND EM.Sex=DE.Sex
```

Example 6'

- Find employees who have the same name and sex to dependents

```
SELECT FName, Sex FROM EMPLOYEE
INTERSECT
SELECT Dependent_Name, Sex FROM DEPENDENT


SELECT EM.*
FROM EMPLOYEE EM, DEPENDENT DE
WHERE EM.FName=DE.Dependent_Name
AND EM.Sex=DE.Sex
```

Example 7

- Find employees who have no dependents

```
SELECT SSN FROM EMPLOYEE
EXCEPT
SELECT ESSN AS SSN FROM DEPENDENT
```

Nested query

```
SELECT  SSN, FName
FROM    EMPLOYEE, DEPARTMENT
WHERE   DName= 'Nghien cuu'  AND DNo=DNumber
```

Outer query

```
SELECT <List_of_columns>
FROM <List_of_tables>
WHERE <Set_comparison> (
        SELECT <List_of_columns>
        FROM <List_of_tables>
        WHERE <Condition>)
```

Subquery

Nested query

- Queries can have several nested levels
  - Usually three
- Subqueries of a WHERE clause are connected by logical connective
  - OR, AND
- Subqueries will return
  - A single attribute and a single tuple (a single value)
  - A table (a set or multiset of tuples)

Nested query

- WHERE clause of the outer query
  - <Expression> <set operation> <subquery>
  - Set comparison includes many operators
    - IN, NOT IN
    - ALL
    - ANY or SOME
  - Check whether the result of subqueries is empty or not
    - EXISTS
    - NOT EXISTS

Nested query

- Categories
  - Subqueries that produce scalar values
    - WHERE clause of subqueries do not refer to attributes of relations in FROM clause of the outer query
    - Subqueries will be performed before the outer query, and be executed just one time
  - Correlated subqueries
    - WHERE clause of subqueries refer to at least one attribute of relations in FROM clause of the outer query
    - Subqueries will be executed many times, each time will correlate to one tuple of the outer query

Example – Scalar value subquery

```
SELECT  SSN, FName
FROM    EMPLOYEE, DEPT_LOCATIONS
WHERE   DLocation= 'TP HCM'  AND DNo=DNumber
```

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   DNo IN (1, 5)
```

```
SELECT  DNumber
FROM    DEPT_LOCATIONS
WHERE   DLocation= 'TP HCM'  )
```

Example 5

```
SELECT  WO.PNo
FROM    EMLOYEE EM, WORKS_ON WO
WHERE   EM.SSN=WO.ESSN AND EM.LName= 'Nguyen'
UNION
SELECT  PR.PNumber
FROM    EMPLOYEE EM, DEPARTMENT DE, PROJECT PR
WHERE   EM.SSN=DE.MgrSSN AND DE.DNumber=PR.DNum
AND EM.LName= 'Nguyen'
```

Example 5

```
SELECT DISTINCT PName
FROM    PROJECT
WHERE  PNumbber IN (
                SELECT  PNo
                FROM    EMPLOYEE, WORKS_ON
                WHERE  SSN=ESSN AND LName= 'Nguyen'  )
OR PNumber IN (
                SELECT  PNumber
                FROM    EMPLOYEE, DEPARTMENT, PROJECT
                WHERE  SSN=ESSN AND PNumber=DNum
                AND LName= 'Nguyen'  )
```

Example 7

- Find employees who have no dependents

```
SELECT  *
FROM    EMPLOYEE
WHERE   SSN NOT IN (
                    SELECT  ESSN
                    FROM    DEPENDENT)

SELECT  *
FROM    EMPLOYEE
WHERE   SSN <> ALL (
                    SELECT  ESSN
                    FROM    DEPENDENT )
```

Example 8

- Find employees whose salary is greater than at least one salary of employees in department 4

```
SELECT  *
FROM    EMPLOYEE
WHERE   Salary > ANY (
                    SELECT  Salary
                    FROM    EMPLOYEE
                    WHERE   DNo=4 )

SELECT  EM1.*
FROM    EMPLOYEE EM1, EMPLOYEE EM2
WHERE   EM1.Salary > EM2.Salary AND EM2.DNo=4
```

Example 9

- Find employees whose salary is greater than all salaries of employees in the department 4

```
SELECT  *
FROM    EMPLOYEE
WHERE   Salary > ALL (
                    SELECT  Salary
                    FROM    EMPLOYEE
                    WHERE   DNo=4 )
```

Example 10

- Find managers who have at least one dependent

```
SELECT  *
FROM    EMPLOYEE
WHERE   SSN IN (SELECT ESSN FROM DEPENDENT)
AND SSN IN (SELECT MgrSSN FROM DEPARTMENT)
```

Example - Correlated subquery

```
SELECT  SSN, FName
FROM    EMPLOYEE, DEPARTMENT
WHERE   DName= 'Nghien cuu'  AND DNo=DNumber
```

```
SELECT  SSN, FName
FROM    EMPLOYEE
WHERE   EXISTS (
        SELECT  *
        FROM    DEPARTMENT
        WHERE   DName= 'Nghien cuu'  AND DNo=DNumber )
```

Example 6

- Retrieve employees who have dependents with the same first name and same sex as the employees

```
SELECT  *
FROM    EMPLOYEE EM
WHERE   EXISTS (
                SELECT  *
                FROM    DEPENDENT DE
                WHERE   EM.SSN=DE.SSN
                AND EM.FName=DE.Name
                AND EM.Sex=DE.Sex )
```

Example 7

- Find employees who have no dependents

```
SELECT  *
FROM    EMPLOYEE
WHERE   NOT EXISTS (
                SELECT  *
                FROM    DEPENDENT
                WHERE   SSN=ESSN)
```

Example 8

- Find employees whose salary is greater than at least one salary of employees in department 4

```
SELECT  *
FROM    EMPLOYEE EM1
WHERE   EXISTS (
                SELECT  *
                FROM    EMPLOYEE EM2
                WHERE   EM2.PNo=4
                AND EM1.Salary>EM2.Salary)
```

Example 10

- Find managers who have at least one dependent

```
SELECT  *
FROM    EMPLOYEE
WHERE   EXISTS (
                SELECT  *
                FROM    DEPENDENT
                WHERE   SSN=ESSN )
AND EXISTS (
                SELECT  *
                FROM    DEPARTMENT
                WHERE   SSN=MgrSSN )
```

Discussion IN and EXISTS

- IN
  - <Column_name> IN <Subquery>
  - Attributes in the subquery's SELECT clause have the same data types as attributes in the outer query's WHERE clause
- EXISTS
  - Do not need attributes, constants or any expressions before it
  - Do not need to specify column names in the subquery's SELECT clause
  - Queries containing "= ANY" or IN can be converted queries containing EXISTS

Discussion

- Comparison of one value and members in a set
  - any/some or exists of nested queries $\Leftrightarrow$ equijoin of basic queries

Divide operation in SQL

R	A	B	C	D	E
	α	a	α	a	1
	α	a	γ	a	1
	α	a	γ	b	1
	β	a	γ	a	1
	β	a	γ	b	3
	γ	a	γ	a	1
	γ	a	γ	b	1
	γ	a	β	b	1

S	D	E
b_i	a	1
	b	1

R÷S	A	B	C
a_i	α	a	γ
	γ	a	γ

- R÷S is a set of values a_i in R such that there is no values b_i in S that makes the tuple (a_i, b_i) does not exist in R

Example 11

- Retrieve the first name of employees who work on all projects

  - Retrieve the first name of employees such that there is no projects that they do not work on

  - R: WORKS_ON(ESSN, PNo)
  - S: PROJECT(PNumber)
  - R÷S: RESULT(ESSN)
  - Joining RESULT to EMPLOYEE to retrieve FName

Example 11

- Using NOT EXISTS two times

```
SELECT  R1.A, R1.B, R1.C
FROM    R R1
WHERE   NOT EXISTS (
        SELECT  *
        FROM    S
        WHERE   NOT EXISTS (
                SELECT  *
                FROM    R R2
                WHERE   R2.D=S.D AND R2.E=S.E
                AND R1.A=R2.A AND R1.B=R2.B AND R1.C=R2.C ))
```

Example 11

```
SELECT  EM.FName
FROM    EMPLOYEE EM, WORKS_ON WO1
WHERE   EM.SSN=WO1.ESSN
AND NOT EXISTS (
        SELECT  *
        FROM    PROJECT PR
        WHERE NOT EXISTS (
                SELECT  *
                FROM    WORKS_ON WO2
                WHERE   WO2.PNo=PR.PNumber
                AND WO1.ESSN=WO2.ESSN ))
```

Example 11

- Using NOT EXISTS and EXCEPT

```
SELECT  R1.A, R1.B, R1.C
FROM    R R1
WHERE   NOT EXISTS (
        ( SELECT D, E FROM S )
        EXCEPT
        ( SELECT D, E FROM R R2 WHERE R1.A=R2.A
                                AND R1.B=R2.B
                                AND R1.C=R2.C ))
```

Example 11

```
SELECT  FName
FROM    EMPLOYEE
WHERE   NOT EXISTS (
        ( SELECT PNumber FROM PROJECT )
        EXCEPT
        ( SELECT PNo FROM WORKS_ON WHERE SSN=ESSN ))
```

Content

- Introduction
- Data definition
- Data manipulation
  - Basic queries
  - Set, set/multiset comparison and nested queries
  - **Aggregate functions and grouping**
  - Others
- Data update
- View definition
- Index

Aggregate functions

- COUNT
  - COUNT(*) : the number of rows
  - COUNT(<Column_name>): the number of non-zero values of the column
  - COUNT(DISTINCT <Column_name>): the number of different and non-zero values of the column
- MIN
- MAX
- SUM
- AVG

- These function is in SELECT clause

Example 12

- Find the sum of salary, highest salary, lowest salary and average salary of employees

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

Example 13

- Find the number of employees in the department ‘Nghien cuu’

```
SELECT  COUNT(*) AS Num_Emp
FROM    EMPLOYEE, DEPARTMENT
WHERE   PNo=PNumber AND PName= ‘Nghien cuu’
```

Example 14

- Find the number of employees for each department

DNo	Num_Emp
5	3
4	3
1	1

SSN	LName	MName	FName	BirthDate	Address	Sex	Salary	SuperSSN	DNo
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	5
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	5
453453453	Tran	Thanh	Tam	07/31/1972	543 MTL Q1	Nu	25000	333445555	5
999887777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	987654321	4
987654321	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	888665555	4
987987987	Tran	Hong	Quang	04/08/1969	980 LHP Q5	Nam	25000	987654321	4
888665555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

Grouping

- Syntax

```
SELECT <List_of_columns>
FROM <List_of_ tables>
WHERE <Conditions>
GROUP BY <List_of_grouping_columns>
```

- After grouping
  - Each group will have identical values at grouping attributes

Example 14

- Find the number of employees for each department

```
SELECT  DNo, COUNT(*) AS Num_Emp
FROM    EMPLOYEE
GROUP BY DNo
```

```
SELECT  DName, COUNT(*) AS Num_Emp
FROM    EMPLOYEE, DEPARTMENT
WHERE   DNo=DNumber
GROUP BY DName
```

Example 15

- For each employee, retrieve the SSN, first name, last name, number of projects as well as total of hours that the employee works on

Example 15

ESSN	PNo	Hours
123456789	1	32.5
123456789	2	7.5
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
888665555	20	20.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
453453453	1	20.0
453453453	2	20.0

Example 15

```
SELECT  ESSN, COUNT(*) AS Num_Pro, SUM(Hours) AS Total_Hours
FROM    WORKS_ON
GROUP BY ESSN
```

```
SELECT  LName, FName, COUNT(*) AS Num_Pro,
        SUM(THOIGIAN) AS Total_Hours
FROM    WORKS_ON, EMPLOYEE
WHERE   ESSN=SSN
GROUP BY ESSN, LName, FName
```

Example 16

- Find employees who work on two or more projects

ESSN	DNo	Hours
123456789	1	32.5
123456789	2	7.5
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
888665555	20	20.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
453453453	1	20.0
453453453	2	20.0

Eliminated

Conditions on groups

- Syntax

```
SELECT <List_of_columns>
FROM   <List_of_tables>
WHERE <Conditions>
GROUP BY <List_of_grouping_columns>
HAVING <Conditions>
```

Example 16

- Find employees who work on two or more projects

```
SELECT  ESSN
FROM    WORKS_ON
GROUP BY ESSN
HAVING COUNT(*) >= 2
```

Example 17

- Retrieve the department name whose the average salary of employees that is greater than 20000

```
SELECT  DNo, AVG(Salary) AS Avg_Salary
FROM    EMPLOYEE
GROUP BY DNo
HAVING  AVG(Salary) > 20000
```

```
SELECT  DName, AVG(Salary) AS Avg_Salary
FROM    EMPLOYEE, DEPARTMENT
WHERE   DNo=DNumber
GROUP BY DNo, DName
HAVING  AVG(Salary) > 20000
```

Discussion

- GROUP BY
  - Attributes in SELECT clause (excepting attributes of aggregate functions) must appear in GROUP BY clause

- HAVING
  - Use aggregate functions in SELECT clause to check a certain condition
  - Just validate the conditions for groups, not a condition for filtering rows
  - After grouping, conditions on groups will be performed

Discussion

- The order of the query execution
  - (1) Pick out rows that satify conditions in WHERE clause
  - (2) Group these rows into many groups in GROUP BY clause
  - (3) Apply aggregate functions for each group
  - (4) Eliminate groups that do not satisfy conditions in HAVING clause
  - (5) Retrieve values from columns and aggregate functions in SELECT clause

Example 18

- Find departments that have the highest average salary

```
SELECT  DNo, AVG(Salary) AS Avg_Salary
FROM    EMPLOYEE
GROUP BY DNo
HAVING AVG(Salary) >= ALL (
                    SELECT  AVG(Salary)
                    FROM    EMPLOYEE
                    GROUP BY DNo)
```

Example 19

- Find three employees who have the highest salary

```
SELECT  FName
FROM    EMPLOYEE EM1
WHERE   2 >= (
                SELECT  COUNT(*)
                FROM    EMPLOYEE EM2
                WHERE   EM2.Salary>EM1.Salary )
```

Discussion

- Find three employees who have the highest salary
  - If salaries are redundant, then ???

Example 12

- Find the first name of employees who work on <u>all</u> projects

```
SELECT  SSN, FName
FROM    EMPLOYEE, WORKS_ON
WHERE   SSN=ESSN
GROUP BY SSN, FName
HAVING COUNT(*) = (
                    SELECT  COUNT(*)
                    FROM    PROJECT)
```

Content

- Introduction
- Data definition
- Data manipulation
  - Basic queries
  - Set, set/multiset comparison and nested queries
  - Aggregate functions and grouping
  - **Other queries**
- Data update
- View definition
- Index

Other queries

- Subquery in FROM clause
- Joining conditions in FROM clause
  - Natural join
  - Outer join
- CASE structure

Subquery in FROM clause

- The result of a subquery is a table
  - Intermediate table in the process of query execution
  - Do not store this result into the database

- Syntax

```
SELECT <List_of_columns>
FROM   R1, R2, (<Subquery>) AS Table_name
WHERE <Conditions>
```

Example 17

- Retrieve the department name whose the average salary of employees that is greater than 20000

```
SELECT  DNo, AVG(Salary) AS Avg_Salary
FROM    EMPLOYEE
GROUP BY DNo
HAVING AVG(Salary) > 20000
```

```
SELECT  DName, AVG(Salary) AS Avg_Salary
FROM    EMPLOYEE, DEPARTMENT
WHERE   DNo=DNumber
GROUP BY DNo, DName
HAVING AVG(Salary) > 20000
```

Example 17

- Retrieve the department name whose the average salary of employees that is greater than 20000

```
SELECT  DName, TEMP.Avg_Salary
FROM    DEPARTMENT, (SELECT DNo, AVG(Salary) AS Avg_Salary
                     FROM  EMPLOYEE
                     GROUP BY DNo
                     HAVING AVG(Salary)> 20000 ) AS TEMP
WHERE DNumber=TEMP.DNo
```

Join conditions in FROM clause

- Equijoin

```
SELECT <List_of_columns>
FROM R1 [INNER] JOIN R2 ON  <Expression>
WHERE <Conditions>
```

- Outer join

```
SELECT <List_of_columns>
FROM R1 LEFT|RIGHT [OUTER] JOIN R2 ON  <Expression>
WHERE <Conditions>
```

Example 20

- Retrieve the SSN and first name of employees who work for the department ‘Nghien cuu’

```
SELECT  SSN, FName
FROM    EMPLOYEE, DEPARTMENT
WHERE   DName= ‘Nghien cuu’  AND DNo=DNumber
```

```
SELECT  SSN, FName
FROM    EMPLOYEE INNER JOIN DEPARTMENT ON DNo=DNumber
WHERE   DName= ‘Nghien cuu’
```

Example 21

- Retrieve the fist and last name of employees, as well as the department name that they are managers

FName	LName	DName
Tung	Nguyen	Nghien cuu
Hang	Bui	null
Nhu	Le	null
Vinh	Pham	Quan ly

```
SELECT  FName, LName, DName
FROM    EMPLOYEE, DEPARTMENT
WHERE   SSN=MgrSSN
```

Example 21

FName	LName	DName
Tung	Nguyen	Nghien cuu
Hang	Bui	null
Nhu	Le	null
Vinh	Pham	Quan ly

Extending the information for EMPLOYEE

EMPLOYEE join DEPARTMENT

SSN=MgrSSN

```
SELECT  FName, LName, DName
FROM    EMPLOYEE LEFT JOIN DEPARTMENT ON SSN=MgrSSN
```

Example 21

FName	LName	DName
Tung	Nguyen	Nghien cuu
Hang	Bui	null
Nhu	Le	null
Vinh	Pham	Quan ly

Extending the information for EMPLOYEE

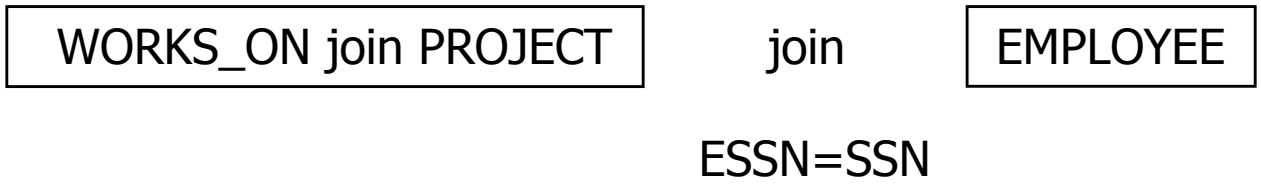
DEPARTMENT join EMPLOYEE

MgrSSN=SSN

```
SELECT  FName, LName, DName
FROM    DEPARTMENT RIGHT JOIN EMPLOYEE ON SSN=MgrSSN
```

Example 22

- Retrieve the first and last name of employees, the name of projects that they work on (if any)




```
SELECT  EM.FName,  PR.PName
FROM    (WORKS_ON WO JOIN PROJECT PR ON PNO=PNumber)
        RIGHT JOIN EMPLOYEE EM ON WO.ESSN=EM.SSN
```

CASE structure

- Allow us to check conditions or output the information in each case

- Syntax

```
CASE <Column_name>
        WHEN <Value> THEN <Expression>
        WHEN <Value> THEN <Expression>
        ...
        [ELSE <Expression>]
END
```

Example 23

- Retrieve the last and first name of employees who are gonna reach the retired age (male: 60 years old, female: 55 years old)

```
SELECT  LName, FName
FROM    EMPLOYEE
WHERE   YEAR(GETDATE()) – YEAR(BirthDate) >= ( CASE Sex
                                                WHEN 'Nam' THEN 60
                                                WHEN 'Nu' THEN 55
                                                END )
```

Example 24

- Retrieve the last and first name of employees, as well as their retired year

```
SELECT LName, FName,
        (CASE Sex
                WHEN 'Nam' THEN  YEAR(BirthDate) + 60
                WHEN 'Nu' THEN YEAR(BirthDate) + 55
        END ) AS RetiredYear
FROM EMPLOYEE
```

Summary

SELECT <List_of_columns>

FROM <List_of_tables>

[**WHERE** <Conditions>]

[**GROUP BY** <List_grouping_columns>]

[**HAVING** <Conditions>]

[**ORDER BY** <List_of_ordering_columns>]

Content

- Introduction
- Data definition
- Data manipulation
- **Data update**
  - Insert
  - Delete
  - Update
- View definition
- Index

INSERT command

- Is used to add 1 or more tuple(s) to a relation

- In order to add a tuple
  - Relation name
  - List of column names
  - List of values for the tuple

INSERT command

- Syntax (one tuple)

INSERT INTO <Table_name>(<List_of_columns>)

VALUES (<List_of_values>)

Example

```
INSERT INTO EMPLOYEE(LName, MName, FName, SSN)
VALUES ( 'Le' , 'Van' , 'Tuyen' , '635635635' )

INSERT INTO EMPLOYEE(LName, MName, FName, SSN, Address)
VALUES ( 'Le' , 'Van' , 'Tuyen' , '635635635' , NULL)

INSERT INTO EMPLOYEE
VALUES ( 'Le' , 'Van' , 'Tuyen' , '635635635' , ' 12/30/1952' , ' 98 HV' , 'Nam' ,
'37000' , 4)
```

INSERT command

- Discussion
  - The order of values is the same to the order of columns
  - The NULL value can be used for non-primary-key attributes
  - INSERT command will raise errors if the integrity constraint is violated
    - Primary key
    - Reference
    - NOT NULL constraint

INSERT command

- Syntax (many tuples)

```
INSERT INTO <Table_name>(<List_of_columns>)
        <Query>
```

Example

```
CREATE TABLE ANALYSE_DEPT (
        DName VARCHAR(20),
        Number_Emp INT,
        Total_Salary INT
)

INSERT INTO ANALYSE_DEPT(DName, Number_Emp, Total_Salary)
    SELECT DName, COUNT(SSN), SUM(Salary)
    FROM EMPLOYEE, DEPARTMENT
    WHERE DNo=DNumber
    GROUP BY DName
```

DELETE command

- Is used to remove tuples from a relation

- Syntax

```
DELETE FROM <Table_name>
[WHERE <Conditions>]
```

Example

```
DELETE FROM EMPLOYEE
WHERE LName= 'Tran'
```

```
DELETE FROM EMPLOYEE
WHERE SSN= '345345345'
```

```
DELETE FROM EMPLOYEE
```

Example 25

- Remove employees who work for the department ‘Nghien cuu’

```
DELETE FROM EMPLOYEE
WHERE DNo IN (
        SELECT DNumber
        FROM DEPARTMENT
        WHERE DName= ‘Nghien cuu’ )
```

DELETE command

- Discussion
  - The number of removed tuples depends on the condition in WHERE clause
  - A missing WHERE clause specifies that all tuples can be deleted
  - DELETE command can cause the violation of reference constraints
    - Do not permit to remove
    - Remove tuples whose value is being referred
      - CASCADE
    - Set the NULL value to cho những giá trị tham chiếu

DELETE command

SSN	LName	MName	FName	BirthDate	Address	Sex	Salary	SuperSSN	DNo
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	5
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	5
453453453	Tran	Thanh	Tam	07/31/1972	543 MTL Q1	Nu	25000	333445555	5
999887777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	987654321	4
987654321	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	888665555	4
987987987	Tran	Hong	Quang	04/08/1969	980 LHP Q5	Nam	25000	987654321	4
888665555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

ESSN	DNo	Hours
333445555	10	10.0
888665555	20	20.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
453453453	1	20.0

DELETE command

DName	DNumber	MgrSSN	MgrStartDate
Nghien cuu	5	333445555	05/22/1988
Dieu hanh	4	987987987	01/01/1995
Quan ly	1	888665555	06/19/1981

SSN	LName	MName	FName	BirthDate	Address	Sex	Salary	SuperSSN	DNo
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	NULL
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	NULL
453453453	Tran	Thanh	Tam	07/31/1972	543 MTL Q1	Nu	25000	333445555	NULL
999887777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	987654321	4
987654321	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	888665555	4
987987987	Tran	Hong	Quang	04/08/1969	980 LHP Q5	Nam	25000	987654321	4
888665555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

UPDATE command

- Is used to change the value of attributes

- Syntax

```
UPDATE <Table_name>
SET <Attribute_name>=<The_new_value>,
      <Attribute_name>=<The_new_value>,
       ...
[WHERE <Condition>]
```

Example

```
UPDATE  EMPLOYEE
SET     BDate=’ 08/12/1965’
WHERE   SSN= ‘333445555’
```

```
UPDATE  EMPLOYEE
SET     Salary=Salary*1.1
```

Example 26

- Change the location and controlling department number of the project 10 to 'Vung Tau' and 5, respectively

```
UPDATE  PROJECT
SET     PLocation='Vung Tau', DNum=5
WHERE   PNumber=10
```

UPDATE command

- Discussion
  - Tuples that satisfy conditions in WHERE clause will be modified to the new value
  - A missing WHERE clause specifies that all tuples can be modified
  - UPDATE command can cause violations of the reference constraint
    - Do not allow to modify
    - Modify the values of tuples that are being referred
      - CASCADE

Content

- Introduction
- Data definition
- Data manipulation
- Data update
- **View definition**
  - Definition
  - Query
  - Modification
- Index

View

- Table is a relation that exist actually the database
  - Stored in some physical organization
  - Persistent

- View is also a relation
  - Do not exist physically (virtual table)
  - Do not contain the data
  - Is derived from other tables
  - Can query or even modify the data through views

View

- Why do we use views?
  - Hide the complexity of data
  - Simplify queries
  - Present data with the purpose “easy to use”
  - Mechanism of data safety

Definition

- Syntax

```
CREATE VIEW <View_name>  AS
        <Query>
```

```
DROP VIEW <View_name>
```

- View contains
  - A list of attributes that are the same as attributes in SELECT clause
  - The number of tuples depending on the conditions in WHERE clause
  - Data derived from tables in FROM clause

Example

```
CREATE VIEW EMP_DEP5 AS
        SELECT  SSN, LName, MName, FName
        FROM    EMPLOYEE
        WHERE   DNo=5

CREATE VIEW STATISTIC_DEP AS
        SELECT DNo, DName, COUNT(*) AS Number_Emp,
                SUM(Salary) AS Total_Salary
        FROM EMPLOYEE, DEPARTMENT
        WHERE DNo=DNumber
        GROUP BY DNumber, DName
```

Querying views

- Although views do not contain data, we can do the query on views

```
SELECT  FName
FROM    EMP_DEP5
WHERE   LName LIKE  'Nguyen'
```

$$EMP\_DEP5 \leftarrow \pi_{SSN,\ LName,\ MName,\ FName}\ (\sigma_{DNo=5}\ (EMPLOYEE))$$

$$\pi_{FName}\ (\sigma_{LName=\ 'Nguyen'}\ (EMP\_DEP5))$$

Querying views

- Can query data from both tables and views

```
SELECT LName, FName, PName, Hours
FROM EMP_DEP5, WORKS_ON, PROJECT
WHERE SSN=ESSN AND PNo=PNumber
```

$$EMP\_DEP5 \leftarrow \pi_{SSN,\ LName,\ MName,\ FName}\ (\sigma_{DNo=5}\ (EMPLOYEE))$$

$$TMP \leftarrow EMP\_DEP5 \bowtie_{SSN=ESSN} (WORKS\_ON \bowtie_{PNo=PNumber} PROJECT)$$

$$\pi_{LName,FName,PName,Hours}(TMP)$$

Modifying views

- Can apply INSERT, DELETE, and UPDATE commands to simple views
  - Views built on one table and having the key attribute of that table

- Cannot modify views
  - Views have a key word DISTINCT
  - Views use aggregate functions
  - Views have extended SELECT clause
  - Views are derived from table containing constrains on columns
  - Views are derived from many tables

Modifying views

- Modify the last name of employee ‘123456789’ in department 5 to ‘Pham’

```
UPDATE  EMP_DEP5
SET     LName= ‘Pham’
WHERE   SSN= ‘123456789’
```

Content

- Introduction
- Data definition
- Data manipulation
- Data update
- View definition
- **Index**

Index

- The index on an attribute A is the data structure that makes it efficient to find tuples having a fixed value for attribute A

```
SELECT  *
FROM    EMPLOYEE
WHERE   DNo=5 AND  Sex= 'Nu'
```

Read 10.000 tuples

Read 200 tuples

Table EMPLOYEE has 10.000 tuples

There are 200 employees who work for the department 5

Read 70 tuples

Index

- Syntax

CREATE INDEX <Index_name> **ON** <Table_name>(<Column_name>)

DROP INDEX <Index_name>

- Example

CREATE INDEX DNo_IND ON EMPLOYEE(DNo)

CREATE INDEX DNo_Sex_IND ON EMPLOYEE(DNo, Sex)

Index

- Discussion
  - Speed up queries in which a value for an attribute is specified, join operations
  - Make insertion, deletion, and update more complex and time-consuming
  - Cost
    - Index storage
    - Disk access (HDD)
- Selection of indexes
  - One of the principal factors that influence a database
  - One of the hardest parts of database design

Example

- Examine the relation WORKS_ON(ESSN, PNo, Hours)
- Assume that

- WORKS_ON is stored in 10 disk blocks
  - The cost of examining the entire relation WORKS_ON is 10
- On the average, an employee works on 3 projects and a project has 3 employees
  - Tuples for a given employee or project are likely to be spread over the 10 disk blocks
  - The cost of finding 3 tuples for an employee or a project will take 3 disk accesses (if we have indexes)
- Using indexes to locate tuples
  - The cost of reading an index's block is 1 disk access
- Insertion costs 2 disk accesses

Example

- Suppose that the following queries are performed frequently
  - Q1
```
SELECT  PNo, Hours
FROM    WORKS_ON
WHERE   ESSN= '123456789'
```
  - Q2
```
SELECT  ESSN
FROM    WORKS_ON
WHERE   PNo=1 AND Hours=20.5
```
  - Q3
```
INSERT INTO WORKS_ON
VALUES ( 123456789' , 1, 20.5)
```

Example

- The costs of each of the 3 queries

Queries	No index	ESSN index	PNo index	Both indexes
Q1	10	4	10	4
Q2	10	10	4	4
Q3	2	4	4	6
Average cost	2 + 8p1 + 8p2	4 + 6p2	4 + 6p1	6 - 2p1 – 2p2

The fraction of the time we do Q1 is p1

The fraction of the time we do Q2 is p2

The fraction of the time we do Q3 là 1 - p1 - p2