

ADVANCED DATABASE

Relational database review & Advanced queries

NGUYEN Hoang Ha

(nguyen-hoang.ha@usth.edu.vn)



Course information

- Moodle page: <https://moodle.usth.edu.vn/course/view.php?id=491>
 - Materials
 - Assignment submissions
- Assessment
 - Final test: 50%
 - Rewards (+2, +1), Penalties (-2, -1)
 - Midterm test: 30%
 - Assignment: 10%
 - Attendance: 10%

Contents & Schedule

Class	Contents	Hours		
		Lect.	Exr.	Prc.
1	Relational databases & advanced queries	3		
2	View, stored procedure, function, and trigger	3		2
3	Transaction and concurrency	3		
4	Relational database performance tuning	3		2
5	Data warehouse	3		
6	Object-oriented databases	3		
7	Semi-structure databases	3		2
8	NoSQL overview	3		
9	High performance NoSQL systems	3		3

Session I Agenda

- Relational Algebra Review
- Normal Forms
- Advanced Queries

RELATION DATABASE REVIEW

Relational Algebra Operations: π , σ , ρ

π :project

Categories

CategoryID	CategoryName	Description	Picture
1	Beverages	Soft drinks, coffees, teas, beers, and ales	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
2	Condiments	Sweet and savory sauces, relishes, spreads, and ...	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
3	Confections	Desserts, candies, and sweet breads	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
4	Dairy Products	Cheeses	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
5	Grains/Cereals	Breads, crackers, pasta, and cereal	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
6	Meat/Poultry	Prepared meats	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
7	Produce	Dried fruit and bean curd	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
8	Seafood	Seaweed and fish	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...

σ :select

$\rho_{\text{Cats}(\text{ID, Name, Description, Picture})}(\text{Categories})$

ρ :rename

Cats

ID	Name	Description	Picture
1	Beverages	Soft drinks, coffees, teas, beers, and ales	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
2	Condiments	Sweet and savory sauces, relishes, spreads, and ...	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
3	Confections	Desserts, candies, and sweet breads	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
4	Dairy Products	Cheeses	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
5	Grains/Cereals	Breads, crackers, pasta, and cereal	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
6	Meat/Poultry	Prepared meats	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
7	Produce	Dried fruit and bean curd	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...
8	Seafood	Seaweed and fish	Qx151C2F00020000000D000E0014002100FFFFFFFF4269746...

Relational Algebra Operations: \bowtie

R1(A	B)
		1	2	
		3	4	
		5	6	

R2(C	D)
		2	7	
		3	5	
		6	9	

Join: $R1 \bowtie_{B=C} R2$



A	B	C	D
1	2	2	7
5	6	6	7

Mathematics to Computer: RA to SQL

Math



RA is the **conceptual basis** for DB systems

Computer



Key of a relation

- A set attributes $\{A_1, A_2, \dots, A_n\}$ is called a **key** of the relation R if:
 - 1. Those attributes determine all other attributes.
→ It is impossible for 2 tuples of R to agree on all of $\{A_1, A_2, \dots, A_n\}$
 - 2. No subset of $\{A_1, A_2, \dots, A_n\}$ determines all other attributes
 - → A Key must be minimal
- Example
 - AccademicResults (StudentID, SubjectID, grade, comment)

Different key types of a relation

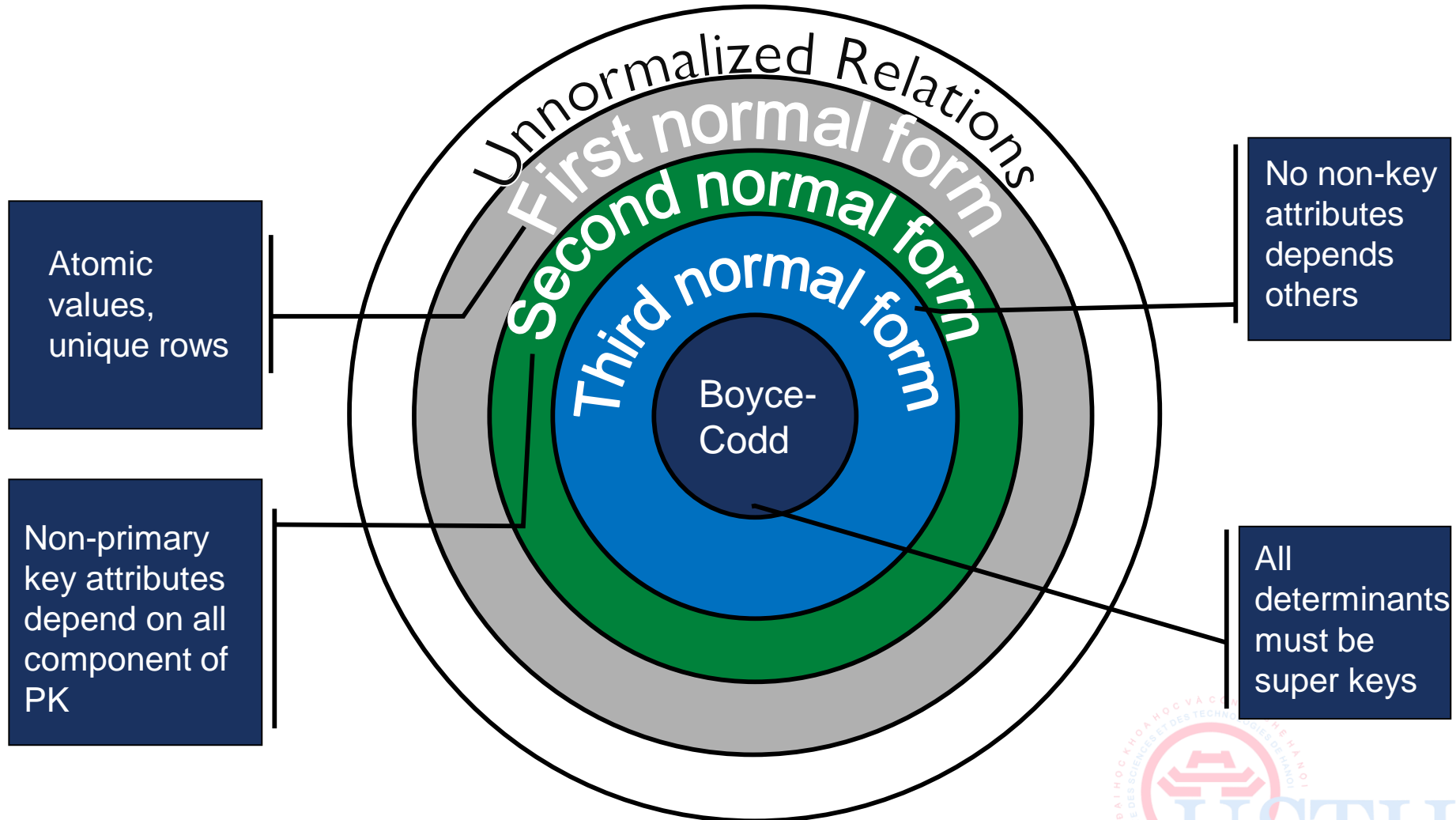
- Super key
 - Set of attributes (columns) to uniquely identify rows
- Key or candidate key
 - Minimal super key
- Primary key
 - One selected from candidate keys
- Alternate key
 - Candidate key other than PK
- Foreign key
 - Attribute refers to a PK of another relation

NORMAL FORMS

Why normalization?

- Data redundancy
- Update abnormally

Normal forms



1NF

- First Normal Form
 - Attributes are **single-valued, atomic**
 - Tuples are **unique** (Identified by PK)
- Case study: Garment management
 - Is Products (item, colors, price, tax) in 1NF?

item	colors	price	tax
T-shirt	red, blue	12.00	0.60
polo	red, yellow	12.00	0.60
T-shirt	red, blue	12.00	0.60
sweatshirt	blue, black	25.00	1.25

- No

1NF normalized

item	colors	price	tax
T-shirt	red, blue	12.00	0.60
polo	red, yellow	12.00	0.60
T-shirt	red, blue	12.00	0.60
sweatshirt	blue, black	25.00	1.25

item	color	price	tax
T-shirt	red	12.00	0.60
T-shirt	blue	12.00	0.60
polo	red	12.00	0.60
polo	yellow	12.00	0.60
sweatshirt	blue	25.00	1.25
sweatshirt	black	25.00	1.25

2NF

- Non-primary key attributes depend on all component of PK
 - PK is a single attribute → guaranteed
- Our case:
 - {item} → {price} , {price} → {tax}

item	color	price	tax
T-shirt	red	12.00	0.60
T-shirt	blue	12.00	0.60
polo	red	12.00	0.60
polo	yellow	12.00	0.60
sweatshirt	blue	25.00	1.25
sweatshirt	black	25.00	1.25

2NF normalized

item	color	price	tax
T-shirt	red	12.00	0.60
T-shirt	blue	12.00	0.60
polo	red	12.00	0.60
polo	yellow	12.00	0.60
sweatshirt	blue	25.00	1.25
sweatshirt	black	25.00	1.25

item	color
T-shirt	red
T-shirt	blue
polo	red
polo	yellow
sweatshirt	blue
sweatshirt	black

item	price	tax
T-shirt	12.00	0.60
polo	12.00	0.60
sweatshirt	25.00	1.25

3NF

- No non-key attributes depends on others
- Example
 - Products (item, price, tax)

item	price	tax
T-shirt	12.00	0.60
polo	12.00	0.60
sweatshirt	25.00	1.25

- $\{item\} \rightarrow \{price\}$, $\{price\} \rightarrow \{tax\}$

item	price
T-shirt	12.00
polo	12.00
sweatshirt	25.00

price	tax
12.00	0.60
25.00	1.25

BCNF (3.5NF or 4NF)

- BCNF Condition: If $X \rightarrow Y$ then X is a **super key**
- Example: Registration (Student, Subject, Lecturer):

<u>Student</u>	<u>Subject</u>	Lecturer
C1	ADB	Daniel
C2	ADB	Daniel
C3	ADB	Green
C4	Image Processing	Kevin
C5	Image Processing	Kevin
C1	Communication System	Brown
C3	Communication System	Brown

- Rules:
 - Each student registers to courses delivered by lecturers
 - Every lecture is in charge maximum one course.
- Functional dependencies:
 - $\{\text{Student, Course}\} \rightarrow \{\text{Lecturer}\}$ **R is in 3NF** as $\{\text{Student, Course}\}$ form the PK
 - $\{\text{Lecturer}\} \rightarrow \{\text{Course}\}$ **R violates BCNF** since Lecturer is not a super key.

BCNF Normalization

- Problems:
 - Data redundancy: many same pairs of {lecturer, course} repeat
 - Anomaly occurs when updating only 2nd row: “Advanced DB” into “Data Mining” → Daniel is for 2 courses

<u>Student</u>	<u>Course</u>	<u>Lecturer</u>
S1	Advanced DB	Daniel
S2	Advanced DB	Daniel
S3	Advanced DB	Green
S4	Image Processing	Kevin
S5	Image Processing	Kevin
S1	Communication System	Brown
S3	Communication System	Brown

BCNF tables:

Teaching (Lecturer, Course)

Register (Student, Lecturer)

<u>Lecturer</u>	<u>Course</u>
Daniel	Advanced DB
Green	Advanced DB
Kevin	Image Processing
Brown	Communication System

<u>Student</u>	<u>Lecturer</u>
S1	Daniel
S2	Daniel
S3	Green
S4	Kevin
S5	Kevin
S1	Brown
S3	Brown

ADVANCED SQL QUERIES

-
- MSSQL Server Developer Edition 2017
 - MSSQL Management Studio

SQL data retrieval query structure

SELECT desired expressions, columns

π

ρ

[FROM one or more tables]

\bowtie_c

[WHERE Conditions about expected rows]

σ

[GROUP BY rows with the same column values]

[HAVING BY condition to restrict result]

[ORDER BY column list]

Advanced SQL QUERIES

- CASE WHEN clause
- Subqueries
- Outer JOIN
- Self JOIN
- Challenges

CASE WHEN clause

- In AdventureWorks2012:
 - Table Production.Product (ProductID, ProductNumber, ProductLine, Name, ...) where ProductLine= {'R','M','T','S', NULL}
 - How to get a list of products with 3 columns ProductNumber, Name, Category which is the meaning full alias of ProductLine

ProductNumber	Category	Name
AR-5381	Not for sale	Adjustable Race
BA-8327	Not for sale	Bearing Ball
BB-7421	Not for sale	LL Bottom Bracket
BB-8107	Not for sale	ML Bottom Bracket
BB-9108	Not for sale	HL Bottom Bracket
BC-M005	Mountain	Mountain Bottle Cage
BC-R205	Road	Road Bottle Cage
BE-2349	Not for sale	BB Ball Bearing
BE-2908	Not for sale	Headset Ball Bearings
BK-M18B-40	Mountain	Mountain-500 Black, 40

```
SELECT ProductNumber, Category =  
    CASE ProductLine  
        WHEN 'R' THEN 'Road'  
        WHEN 'M' THEN 'Mountain'  
        WHEN 'T' THEN 'Touring'  
        WHEN 'S' THEN 'Other sale items'  
        ELSE 'Not for sale'  
    END,  
    Name  
FROM Production.Product  
ORDER BY ProductNumber;
```

Another CASE WHEN example in SELECT

	ProductNumber	Name	ListPrice
1	AR-5381	Adjustable Race	0.00
2	BA-8327	Bearing Ball	0.00
3	BB-7421	LL Bottom Bracket	53.99
4	BB-8107	ML Bottom Bracket	101.24
5	BB-9108	HL Bottom Bracket	121.49
6	BC-M005	Mountain Bottle Cage	9.99
7	BC-R205	Road Bottle Cage	8.99
8	BE-2349	BB Ball Bearing	0.00
9	BE-2908	Headset Ball Bearings	0.00
10	BK-M18B-40	Mountain-500 Black, 40	539.99

	ProductNumber	Name	Price Range
1	AR-5381	Adjustable Race	Mfg item - not for resale
2	BA-8327	Bearing Ball	Mfg item - not for resale
3	BB-7421	LL Bottom Bracket	Under \$250
4	BB-8107	ML Bottom Bracket	Under \$250
5	BB-9108	HL Bottom Bracket	Under \$250
6	BC-M005	Mountain Bottle Cage	Under \$50
7	BC-R205	Road Bottle Cage	Under \$50
8	BE-2349	BB Ball Bearing	Mfg item - not for resale
9	BE-2908	Headset Ball Bearings	Mfg item - not for resale
10	BK-M18B-40	Mountain-500 Black, 40	Under \$1000

```

SELECT ProductNumber, Name, "Price Range" =
CASE
    WHEN ListPrice = 0 THEN 'Mfg item - not for resale'
    WHEN ListPrice < 50 THEN 'Under $50'
    WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'
    WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'
    ELSE 'Over $1000'
END
FROM Production.Product
ORDER BY ProductNumber ;
    
```

Use CASE WHEN in ORDER BY

- List salesmen order by countries. If salesmen lives in the USA, we sort by his region

	BusinessEntityID	LastName	TerritoryName	CountryRegionName
1	286	Tsoflias	Australia	Australia
2	278	Vargas	Canada	Canada
3	282	Saraiva	Canada	Canada
4	277	Carson	Central	United States
5	290	Varkey Chudukatil	France	France
6	288	Valdez	Germany	Germany
7	275	Blythe	Northeast	United States
8	283	Campbell	Northwest	United States
9	284	Mensa-Annan	Northwest	United States
10	280	Ansman-Wolfe	Northwest	United States
11	279	Reiter	Southeast	United States
12	276	Mitchell	Southwest	United States
13	281	Ito	Southwest	United States
14	289	Pak	United Kingdom	United Kingdom

```
SELECT BusinessEntityID, LastName, TerritoryName, CountryRegionName
FROM Sales.vSalesPerson
WHERE TerritoryName IS NOT NULL
ORDER BY CASE CountryRegionName WHEN 'United States' THEN TerritoryName
          ELSE CountryRegionName END;
```

Use CASE WHEN in HAVING clause

- Use HAVING to get JobTitle whose MAX(payrate) of men > \$40 or MAX(payrate) of men > \$42

	JobTitle	MaximumRate
1	Chief Executive Officer	125.50
2	Vice President of Production	84.1346
3	Vice President of Sales	72.1154
4	Vice President of Engineering	63.4615
5	Chief Financial Officer	60.0962
6	Research and Development Manager	50.4808
7	Information Services Manager	50.4808
8	North American Sales Manager	48.101
9	Pacific Sales Manager	48.101
10	European Sales Manager	48.101
11	Finance Manager	43.2692
12	Engineering Manager	43.2692

```
SELECT JobTitle, MAX(ph1.Rate) AS MaximumRate
FROM HumanResources.Employee AS e
     JOIN HumanResources.EmployeePayHistory AS ph1
       ON e.BusinessEntityID = ph1.BusinessEntityID
GROUP BY JobTitle
HAVING (MAX(CASE WHEN Gender = 'M'
                THEN ph1.Rate
                ELSE NULL END) > 40.00
        OR MAX(CASE WHEN Gender = 'F'
                THEN ph1.Rate
                ELSE NULL END) > 42.00)
ORDER BY MaximumRate DESC;
```

Subquery

Main Query:



Which employees have salaries greater than Abel's salary?

Subquery



What is Abel's salary?



```
SELECT last_name
FROM employees
WHERE salary > 11000
      (SELECT salary
       FROM employees
       WHERE last_name = 'Abel');
```

Correlated Sub-Query

- A correlated subquery is one way of reading every row in a table and comparing values in each row against related data.
- It is used whenever a sub-query must return a different result or set of results for each candidate row considered by the main query.

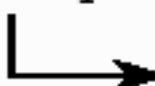
```
SELECT column1, column2, ...  
FROM   table1 outer  
WHERE  column1 operator  
        (SELECT column1, column2  
         FROM    table2  
         WHERE   expr1 =  
                outer.expr2);
```

The subquery references a column from a table in the parent query.

Correlated Sub-query Example

Find all employees who earn more than the average salary in their department.

```
SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary >
      (SELECT AVG(salary)
       FROM   employees
       WHERE  department_id =
             outer.department_id) ;
```



Each time a row from the outer query is processed, the inner query is evaluated.

Correlated Sub-query Example

Display details of those employees who have switched jobs at least twice.

```
SELECT e.employee_id, last_name, e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT(*)
              FROM   job_history
              WHERE  employee_id = e.employee_id);
```


Subqueries in FROM clauses

- Another use for subqueries is as relations in a FROM clause:
- SELECT name
FROM movieExec,
(SELECT producerC#
FROM movies, starsIn
WHERE title = movieTitle
AND year = movieYear
AND starName = 'Harrison Ford'
) as Prod
WHERE cert# = Prod.producerC#

OUTER JOIN

- List all Customers with theirs Orders, even those without any order (In Northwind)

	CustomerID	CompanyName	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	Ship
827	SIMOB	Simons bistro	11074	SIMOB	7	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL	2
828	RICSU	Richter Supermarkt	11075	RICSU	8	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL	2
829	BONAP	Bon app'	11076	BONAP	4	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL	2
830	RATTC	Rattlesnake Canyon Grocery	11077	RATTC	1	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL	2
831	PARIS	Paris spécialités	NULL	NULL	NULL	NULL	NULL	NULL	NUL
832	FISSA	FISSA Fabrica Inter. Salchi...	NULL	NULL	NULL	NULL	NULL	NULL	NUL

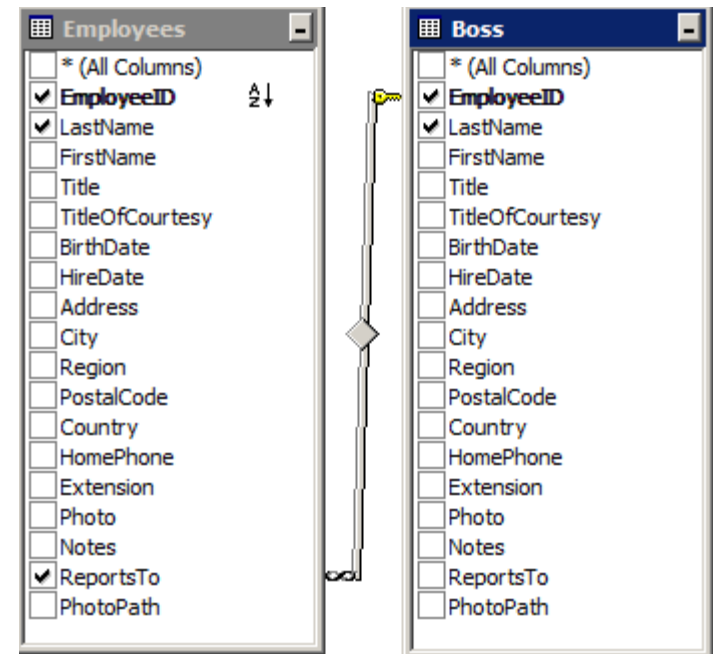
```
SELECT Customers.CustomerID, CompanyName, [Orders].*
FROM Customers LEFT OUTER JOIN [Orders] ON Customers.CustomerId = [Orders].CustomerId
WHERE OrderDate IS NULL
```

```
SELECT Customers.CustomerID, CompanyName, [Orders].*
FROM [Orders] RIGHT OUTER JOIN Customers ON Customers.CustomerId =
[Orders].CustomerId
WHERE OrderDate IS NULL
```

SELF JOIN

- List employees' fullname and his/her boss name (in Northwind DB).

	FullName	Boss
1	Nancy Davolio	Andrew Fuller
2	Janet Leverling	Andrew Fuller
3	Margaret Peacock	Andrew Fuller
4	Steven Buchanan	Andrew Fuller
5	Michael Suyama	Steven Buchanan
6	Robert King	Steven Buchanan
7	Laura Callahan	Andrew Fuller
8	Anne Dodsworth	Steven Buchanan



```
SELECT E.FirstName + ' ' + E.LastName AS FullName, B.FirstName + ' ' + B.LastName Boss
FROM Employees E INNER JOIN Employees B ON E.ReportsTo = B.EmployeeID
```

Challenge I



- In Northwind, list categories whose average prices are greater than 30\$, sort by those values

	CategoryName	AvgPrice
1	Produce	32.37
2	Beverages	37.9791
3	Meat/Poultry	54.0066

```
SELECT C.CategoryID, C.CategoryName, AVG(P.UnitPrice) AvgPrice
FROM Categories C INNER JOIN Products P ON C.CategoryID = P.CategoryID
GROUP BY C.CategoryID, C.CategoryName
HAVING AVG(P.UnitPrice) > 30
ORDER BY AVG(P.UnitPrice)
```

Challenge 2



- In Northwind, find the 3rd highest Unit Price of Products

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1

```
SELECT P1.*  
FROM Products P1 WHERE 3 = (SELECT COUNT(DISTINCT UnitPrice)  
                             FROM Products P2  
                             WHERE P1.UnitPrice <= P2.UnitPrice  
                             )
```