Advanced Queries

IS240 – DBMS

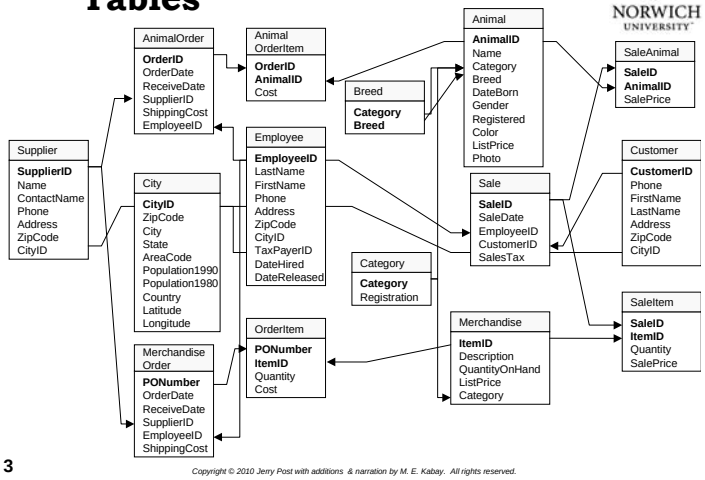
Lecture # 8 – 2010-03-08

M. E. Kabay, PhD, CISSP-ISSMP
Assoc. Prof. Information Assurance
Division of Business & Management, Norwich University
mailto:mkabay@norwich.edu **V: 802.479.7937**

Topics

- Sub-query for Calculation
- Query Sets (IN)
- Using IN with a Sub-query
- Left Outer Join
- Older Syntax for Left Join
- SubQuery for Computation
- Correlated Subquery
- UNION Operator
- Multiple JOIN Columns
- CASE Function
- Inequality Join
- SQL SELECT & SQL Mnemonic

Tables



Harder Questions

- How many cats are "in-stock" on 10/1/04?
- Which cats sold for more than the average price?
- Which animals sold for more than the average price of animals in their category?
- Which animals have *not* been sold?
- Which customers (who bought something at least once) did *not* buy anything between 11/1/04 and 12/31/04?
- Which customers who bought Dogs also bought products for Cats (at any time)?

Sub-query for Calculation

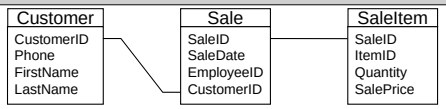
- Which cats sold for more than the average sale price of cats?
  - Assume we know the average price is $170.
  - Usually we need to compute it first.

```
SELECT SaleAnimal.AnimalID, Animal.Category,
SaleAnimal.SalePrice
FROM Animal
INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE ((Animal.Category='Cat') AND (SaleAnimal.SalePrice>170));
```

```
SELECT SaleAnimal.AnimalID, Animal.Category, SaleAnimal.SalePrice
FROM Animal
INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE ((Animal.Category='Cat') AND (SaleAnimal.SalePrice>
  (  SELECT AVG(SalePrice)
     FROM Animal
     INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
     WHERE (Animal.Category='Cat')
  )
 ) );
```


Query Sets (IN)

```
SELECT Customer.LastName, Customer.FirstName, SaleItem.ItemID
FROM (Customer INNER JOIN Sale ON Customer.CustomerID = Sale.CustomerID)
  INNER JOIN SaleItem ON Sale.SaleID = SaleItem.SaleID
WHERE (SaleItem.ItemID In (1,2,30,32,33))
ORDER BY Customer.LastName, Customer.FirstName;
```



Field	LastName	FirstName	ItemID
Table	Customer	Customer	SaleItem
Sort	Ascending	Ascending	
Criteria			In (1,2,30,32,33)
Or			

List all customers (Name) who purchased one of the following items: 1, 2, 30, 32, 33.

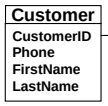
Using IN with a Sub-query

- List all customers who bought items for cats.

```
SELECT Customer.LastName, Customer.FirstName, SaleItem.ItemID
FROM (Customer
      INNER JOIN Sale ON Customer.CustomerID = Sale.CustomerID)
      INNER JOIN SaleItem ON Sale.SaleID = SaleItem.SaleID
WHERE (SaleItem.ItemID In
      (SELECT ItemID FROM Merchandise WHERE Category='Cat')
      );
```


SubQuery (IN: Look up a Set)

```
SELECT Customer.LastName, Customer.FirstName
FROM Customer INNER JOIN Sale ON Customer.CustomerID = Sale.CustomerID
WHERE ((Month([SaleDate])=3)) And Customer.CustomerID In
  (SELECT CustomerID
   FROM Sale
   WHERE (Month([SaleDate])=5) );
```



LastName	First
Adkins	Inga
McCain	Sam
Grimes	Earl

Field	LastName	FirstName	Month(SaleDate)	CustomerID
Table	Customer	Customer	Sale	Customer
Sort	Ascending	Ascending		
Criteria			3	In (SELECT CustomerID FROM State WHERE (Month(SaleDate)=5)
Or				

List all of the customers who bought something in March and who also bought something in May. (Two tests on the same data!)

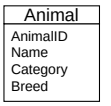
SubQuery (ANY, ALL)

Find animals that sold for *more than any of the prices* of cats
(= find animals that sold for *more than the greatest price* of any cat)

```
SELECT Animal.AnimalID, Name, SalePrice, ListPrice
FROM Animal
INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE ((SalePrice > Any
  (SELECT ListPrice
   FROM Animal
   INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
   WHERE Category = 'Cat'))
AND (Category='Cat');
```

- **Any: value is compared to each item in the list. If it is True for any of the items, the statement is evaluated to True.**
- **All: value is compared to each item in the list. If it is True for every item in the list, the statement is evaluated to True (much more restrictive than *any*).**

SubQuery: NOT IN (Subtract)




```
SELECT Animal.AnimalID, Animal.Name, Animal.Category
FROM Animal
WHERE (Animal.AnimalID Not In
  (SELECT AnimalID From SaleAnimal));
```

Field	AnimalID	Name	Category
Table	Animal	Animal	Animal
Sort			
Criteria	Not In (SELECT AnimalID FROM SaleAnimal)		
Or			

AnimalID	Name	Category
12	Leisha	Dog
19	Gene	Dog
25	Vivian	Dog
34	Rhonda	Dog
88	Brandy	Dog
181		Fish

~~SELECT Animal.AnimalID, Name, SalePrice, ListPrice~~
~~FROM Animal~~
~~INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID~~
~~....~~

- **Which animals have *not* been sold?**
  - **Start with list of all animals.**
  - **Subtract out list of those who were sold.**

SubQuery: NOT IN (Data)

Animal

ID	Name	Category	Breed
~~2~~		~~Fish~~	~~Angel~~
~~4~~	~~Gary~~	~~Dog~~	~~Dalmation~~
5		**Fish**	**Shark**
~~6~~	~~Rosie~~	~~Cat~~	~~Oriental Shorthair~~
~~7~~	~~Eugene~~	~~Cat~~	~~Bombay~~
~~8~~	~~Miranda~~	~~Dog~~	~~Norfolk Terrier~~
9		**Fish**	**Guppy**
~~10~~	~~Sherri~~	~~Dog~~	~~Siberian Huskie~~
~~11~~	~~Susan~~	~~Dog~~	~~Dalmation~~
12	**Leisha**	**Dog**	**Rottweiler**

SaleAnimal

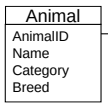
ID	SaleID	SalePrice
2	35	$10.80
4	80	$156.66
6	27	$173.99
7	25	$251.59
8	4	$183.38
10	18	$150.11
11	17	$148.47

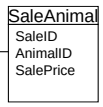
```
SELECT Animal.AnimalID, Animal.Name, Animal.Category
FROM Animal
WHERE (Animal.AnimalID Not In
  (SELECT AnimalID From SaleAnimal));
```

- **Which animals have *not* been sold?**
  - **Start with list of all animals.**
  - **Subtract out list of those who were sold.**

Left Outer Join

```
SELECT Animal.AnimalID, Animal.Name, Animal.Category, SaleAnimal.SalePrice
FROM Animal LEFT JOIN SaleAnimal
ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE (SaleAnimal.SaleID Is Null);
```





Field	AnimalID	SaleID	Name	Category
Table	Animal	SaleAnimal	Animal	Animal
Sort				
Criteria		Is Null		
Or				

AnimalID	Name	Category
12	Leisha	Dog
19	Gene	Dog
25	Vivian	Dog
34	Rhonda	Dog
88	Brandy	Dog
181		Fish

~~*FROM*~~
~~*Animal INNER JOIN SaleAnimal*~~
~~*ON*~~
~~*Animal.AnimalID = SaleAnimal.AnimalID*~~

- **Which animals have not been sold?**
- **LEFT JOIN includes all rows from left table (Animal)**
- **But only those from right table (SaleAnimal) *that match a row in Animal on AnimalID*.**
- **Thus rows in Animal without matching data in Sale Animal will have Null data for SaleAnimal.AnimalID and SaleAnimal.SalePrice**

Left Outer Join (Example)

ID	Name	Category	Breed	ID	SaleID	SalePrice
2		Fish	Angel	2	35	$10.80
4	Gary	Dog	Dalmation	4	80	$156.66
5		**Fish**	**Shark**	**Null**	**Null**	**Null**
6	Rosie	Cat	Oriental Shorthair	6	27	$173.99
7	Eugene	Cat	Bombay	7	25	$251.59
8	Miranda	Dog	Norfolk Terrier	8	4	$183.38
9		**Fish**	**Guppy**	**Null**	**Null**	**Null**
10	Sherri	Dog	Siberian Huskie	10	18	$150.11
11	Susan	Dog	Dalmation	11	17	$148.47
12	**Leisha**	**Dog**	**Rottweiler**	**Null**	**Null**	**Null**

SubQuery for Computation

```
SELECT SaleAnimal.AnimalID, Animal.Category, SaleAnimal.SalePrice
FROM Animal
INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE ((Animal.Category='Cat') AND (SaleAnimal.SalePrice>
   ( SELECT AVG(SalePrice)
     FROM Animal
     INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
     WHERE (Animal.Category='Cat')          ) ) );
```

Animal
AnimalID
Name
Category
Breed

SaleAnimal
SaleID
AnimalID
SalePrice

- ***Don't know the average, so use a subquery to compute it.***
- ***Watch parentheses.***

Field	AnimalID	Name	Category	SalePrice
Table	**Animal**	**Animal**	**Animal**	**SaleAnimal**
Sort				**Descending**
Criteria			**3**	**> (SELECT Avg(SalePrice) FROM Animal INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID WHERE Animal.Category = 'Cat')**
Or				

Correlated Subquery

- **List the Animals that have sold for a price higher than the *average* for animals in that Category.**

```
SELECT AnimalID, Name, Category, SalePrice
FROM Animal INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
WHERE (SaleAnimal.SalePrice>
   (SELECT Avg(SaleAnimal.SalePrice)
    FROM Animal INNER JOIN SaleAnimal ON Animal.AnimalID = SaleAnimal.AnimalID
    WHERE (Animal.Category = Animal.Category) ) )
ORDER BY SaleAnimal.SalePrice DESC;
```

- **The subquery needs to compute the average for a given category.**
- **Problem: Which category?**
- **Answer: the category that matches the category from the main part of the query.**
- **Problem: How do we refer to it? Both tables are called Animal. This query will not work yet.**

Correlated SubQuery (Avoid)

- **Match category in subquery with top level**
  - **Rename tables (As)**
- **Correlated Subquery**
  - **Recompute subquery for every row in top level--slow!**
  - **Better to compute and save Subquery, then use in join.**

```
SELECT A1.AnimalID, A1.Name, A1.Category, SaleAnimal.SalePrice
FROM Animal As A1 INNER JOIN SaleAnimal
        ON A1.AnimalID = SaleAnimal.AnimalID
WHERE (SaleAnimal.SalePrice>
   (SELECT Avg(SaleAnimal.SalePrice)
    FROM Animal As A2  INNER JOIN SaleAnimal
        ON A2.AnimalID = SaleAnimal.AnimalID
    WHERE (A2.Category = A1.Category) ) )
ORDER BY SaleAnimal.SalePrice DESC;
```

This is recomputed for every new category

List the Animals that have sold for a price higher than the average for animals in that Category.

Correlated Subquery Problem

Animal + SaleAnimal

Category	SalePrice	
Fish	$10.80	→ Compute Avg: $37.78
Dog	$156.66	→ Compute Avg: $174.20
Fish	$19.80	→ Compute Avg: $37.78
Cat	$173.99	→ Compute Avg: $169.73
Cat	$251.59	→ Compute Avg: $169.73
Dog	$183.38	
Fish	$1.80	
Dog	$150.11	
Dog	$148.47	

Recompute average for every row in the main query!

- **Assume small query**
  - **100,000 rows**
  - **5 categories of 20,000 rows**
- **100,000 * 20,000 = 1 billion rows to read!**

More Efficient Solution: 2 queries

Animal + SaleAnimal

Category	SalePrice
Fish	$10.80
Dog	$156.66
Fish	$19.80
Cat	$173.99
Cat	$251.59
Dog	$183.38
Fish	$1.80
Dog	$150.11
Dog	$148.47

JOIN

Animal.Category = Query1.Category

Saved Query

Category	AvgOfSalePrice
Bird	$176.57
Cat	$169.73
Dog	$174.20
Fish	$37.78
Mammal	$80.72
Reptile	$181.83
Spider	$118.16

- **Compute the averages once and save query**
- **JOIN saved query to main query**
- **Two passes through table: 1 billion / 200,000 => 10,000**

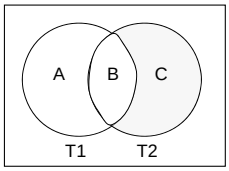
UNION Operator

```
SELECT EID, Name, Phone, Salary, 'East' AS Office
FROM EmployeeEast
UNION
SELECT EID, Name, Phone, Salary, 'West' AS Office
FROM EmployeeWest
```

EID	Name	Phone	Salary	Office
352	Jones	3352	45,000	East
876	Inez	8736	47,000	East
372	Stoiko	7632	38,000	East
890	Smythe	9803	62,000	West
361	Kim	7736	73,000	West

- **Offices in Los Angeles and New York.**
- **Each has an Employee table (East and West).**
- **Need to search data from both tables.**
- **Columns in the two SELECT lines must match.**

UNION, INTERSECT, EXCEPT

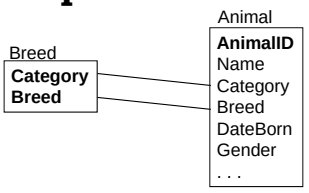


List the name of any employee who has worked for both the East and West regions.

T1 UNION T2	A + B + C
T1 INTERSECT T2	B
T1 EXCEPT T2	A

```
SELECT EID, Name
FROM EmployeeEast
INTERSECT
SELECT EID, Name
FROM EmployeeWest
```


Multiple JOIN Columns




```
SELECT *
FROM Breed INNER JOIN Animal
ON Breed.Category = Animal.Category
AND Breed.Breed = Animal.Breed
```

- **Sometimes need to JOIN tables on more than one column.**
- **PetStore: Category and Breed.**

Reflexive Join

SQL

```
SELECT Employee.EID,
Employee.Name,
Employee.Manager, E2.Name
FROM Employee INNER JOIN
Employee AS E2
ON Employee.Manager = E2.EID
```

Employee

EID	Name	. . .	Manager
115	Sanchez		765
462	Miller		115
523	Hawk		115
765	Munoz		886

Result

EID	Name	Manager	Name
115	Sanchez	765	Munoz
462	Miller	115	Sanchez
523	Hawk	115	Sanchez

- **Need to connect a table to itself.**
- **Common example: Employee(EID, Name, . . ., Manager)**
  - **A manager is also an employee.**
  - **Use a second copy of the table and an alias.**

CASE Function

```
Select AnimalID,
    CASE
        WHEN Date()-DateBorn < 90 Then "Baby"
        WHEN Date()-DateBorn >= 90
          AND Date()-DateBorn < 270 Then "Young"
        WHEN Date()-DateBorn >= 270
          AND Date()-DateBorn < 365 Then "Grown"
        ELSE "Experienced"
    END
FROM Animal;
```

Not available in Microsoft Access. It is in SQL Server and Oracle.

- **Used to change data to a different context.**
- **Example: Define age categories for the animals.**
  - **Less than 3 months**
  - **Between 3 months and 9 months**
  - **Between 9 months and 1 year**
  - **Over 1 year**

Inequality Join

- **AccountsReceivable ("AR")**
- **Categorize by Days Late**
  - **30, 90, 120+**
- **Three queries?**
- **Better: JOIN to new table for business rules**

AR(TransactionID, CustomerID, Amount, DateDue)

LateCategory(Category, MinDays, MaxDays, Charge, …)

Month	30	90	3%
Quarter	90	120	5%
Overdue	120	9999	10%

```
SELECT *
FROM AR INNER JOIN LateCategory
 ON ((Date() - AR.DateDue) >= LateCategory.MinDays)
 AND ((Date() - AR.DateDue) < LateCategory.MaxDays)
```

SQL SELECT: REVIEW

SELECT DISTINCT Table.Column {AS alias} , . . .
FROM Table/Query
INNER JOIN Table/Query ON T1.ColA = T2.ColB
WHERE (condition)
GROUP BY Column
HAVING (group condition)
ORDER BY Table.Column
{ Union second select }

SQL Mnemonic

Someone	SELECT
From	FROM
Ireland	INNER JOIN
Will	WHERE
Grow	GROUP BY
Horseradish and	HAVING
Onions	ORDER BY

SQL is picky about putting the commands in the proper sequence.

If you have to memorize the sequence, this mnemonic may be helpful.

Homework

- **Study Chapter 5 pp 220-244 carefully using SQ3R**
- **Use the Review Questions 1-8 on pp 251-252 to challenge yourselves.**
- **Recommended exercises:**
  - **Sally's Pets: previous classes completed all 25 of these exercises within two weeks and sweated blood to do so**
  - **You can tackle them without pressure. Do at least the first 10 to apply the theory**
  - ***USE THE SQL INTERPRETER IN MS-ACCESS TO TEST ALL YOUR QUERIES***
  - ***Solutions will be distributed after the Spring Break.***

DISCUSSION