Outer Joins and Unions

Outer Joins

- The outer join includes all the rows from 1 or both table(s) (as specified by full, left or right) and any matching rows from another table.
- With the outer join, it is possible to have rows for which no data exists in the opposite table.
- There shall be as many rows as there are in the primary table plus any multiple rows from the secondary table for a given primary row.

Outer Joins

- This kind of join is useful for generating a comprehensive list when some of the entities in the main table do not have related rows in the secondary table.
- Consider that we have a number of computers and updates have applied to some of them.
- Not all computers in the Computers table have had updates applied to it. In a left join select statement, these non-updated computers would still show up in the result set, with null values filling any column with no value.

Illustrating Inner and Outer Joins

- Consider that we have the 2 tables below

Course Table	
Course Name	**Course ID**
Intro To Management	MNGT 1111
Marketting Strategy	MARK 1020
Computer Programming	COMP 1100
Accounting Principles	ACTG 1211
English Literature	ENGL 2101
Event Driven Programming	COMP2112
Database Systems	COMP 1204
Machine Learning	COMP 4123
Web Development	COMP 3121

Instructor-Course Table	
Instructor	**Course Id**
Jonny Whitney	ACTG 1211
Jonny Whitney	MARK 1020
Harry Willow	COMP 1100
Harry Willow	COMP 2112
Christina Venus	Null
Jenny Sanders	COMP 2112
Kris Martian	COMP 2112
Steph Parland	COMP1100
Jemina Maryland	Null
Steph Parland	COMP 1204
Cathy Walner	ENGl 2101
Sandy Bard	COMP 1204
Melvin Farmer	ENGL 2101
Hailey Lucas	Null

Illustrating Inner and Outer Joins

- Looking at the 2 given tables, we note that some courses do not have instructors
  - The courses may not be on offer during the current term
- Some instructors do not have courses
  - They may be on leave for the term or working on other programs

Inner Join on Course ID		
CourseTable.Course Name	**CourseTable.Course ID**	**Instructor-CourseTable.Instructor**
Marketting Strategy	MARK 1020	Jonny Whitney
Computer Programming	COMP 1100	Harry Willow
Computer Programming	COMP 1100	Steph Parland
Accounting Principles	ACTG 1211	Jonny Whitney
English Literature	ENGL 2101	Cathy Walner
English Literature	ENGL 2101	Melvin Farmer
Event Driven Programming	COMP2112	Harry Willow
Event Driven Programming	COMP2112	Jenny Sanders
Event Driven Programming	COMP2112	Kris Martian
Database Systems	COMP 1204	Steph Parland

Illustrating Inner and Outer Joins

Course Table Left Outer Join Instructor Table on Course ID		
CourseTable.Course Name	CourseTable.Course ID	Instructor-CourseTable.Instructor
Intro To Management	MNGT 1111	Null
Marketting Strategy	MARK 1020	Jonny Whitney
Computer Programming	COMP 1100	Harry Willow
Computer Programming	COMP 1100	Steph Parland
Accounting Principles	ACTG 1211	Jonny Whitney
English Literature	ENGL 2101	Cathy Walner
English Literature	ENGL 2101	Melvin Farmer
Event Driven Programming	COMP2112	Harry Willow
Event Driven Programming	COMP2112	Jenny Sanders
Event Driven Programming	COMP2112	Kris Martian
Database Systems	COMP 1204	Steph Parland
Machine Learning	COMP 4123	Null
Web Development	COMP 3121	Null

Illustrating Inner and Outer Joins

Course Table Right Outer Join Instructor Table on Course ID		
CourseTable.Course Name	**CourseTable.Course ID**	**Instructor-courseTable.Instructor**
Marketting Strategy	MARK 1020	Jonny Whitney
Computer Programming	COMP 1100	Harry Willow
Computer Programming	COMP 1100	Steph Parland
Accounting Principles	ACTG 1211	Jonny Whitney
English Literature	ENGL 2101	Cathy Walner
English Literature	ENGL 2101	Melvin Farmer
Event Driven Programming	COMP2112	Harry Willow
Event Driven Programming	COMP2112	Jenny Sanders
Event Driven Programming	COMP2112	Kris Martian
Database Systems	COMP 1204	Steph Parland
Null	Null	Christina Venus
Null	Null	Jemina Maryland
Null	Null	Hailey Lucas

Illustrating Inner and Outer Joins

Course Table Full Outer Join Instructor Table on Course ID		
CourseTable.Course Name	CourseTable.Course ID	Instructor-CourseTable.Instructor
Intro To Management	MNGT 1111	**Null**
Marketting Strategy	MARK 1020	Jonny Whitney
Computer Programming	COMP 1100	Harry Willow
Computer Programming	COMP 1100	Steph Parland
Accounting Principles	ACTG 1211	Jonny Whitney
English Literature	ENGL 2101	Cathy Walner
English Literature	ENGL 2101	Melvin Farmer
Event Driven Programming	COMP2112	Harry Willow
Event Driven Programming	COMP2112	Jenny Sanders
Event Driven Programming	COMP2112	Kris Martian
Database Systems	COMP 1204	Steph Parland
Machine Learning	COMP 4123	**Null**
Web Development	COMP 3121	**Null**
Null	**Null**	Christina Venus
Null	**Null**	Jemina Maryland
Null	**Null**	Hailey Lucas

Illustrating Inner and Outer Joins

ANSI left outer join

```
select      <[tableName.]columnName, …>
from        <leftTableName>
    left outer join  <rightTableName>
    on      <joinCondition>
[where      <searchCondition(s)>]
[order by <orderingCondition(s)>]
```

Pubs Schema – left outer join

- The PUBS database has a discount table with 3 rows of which 2 have a null stor_id and the stores table which has 6 rows. The left outer join looks like:

```
select      s.stor_name as 'Store', d.stor_id as 'Store ID', d.discount as 'Discount'
from        PublishersDatabase.dbo.stores as s      -- left position
 left outer join  PublishersDatabase.dbo.discounts as d
 on         d.stor_id = s.stor_id
```

ANSI right outer join

```
select  <[tableName.]columnName, …>
from    <leftTableName>
right outer join   <rightTableName>
on      <joinCondition>
[where  <searchCondition(s)>]
[order by   <orderingCondition(s)>]
```

Pubs Schema – right outer join

- The right outer join will show all discounts, including the store name that has a discount

```
select      discount as 'Discount',
            stor_name as 'Store',
            d.stor_id as 'Store ID'
from        PublishersDatabase.dbo.stores as s
right outer join  PublishersDatabase.dbo.discounts as d
on          d.stor_id = s.stor_id
```

Exercise 1 – Outer Joins

- Generate a list of all books of type ‘popular_comp’ and the royalty ranges (hi range, lo range and royalty).
- (10 rows – 9 with range values).

Exercise – Outer Joins Solution

```
--ANSI
select      t.title as 'Title',
            r.royalty as 'Royalty',
            r.lorange as 'Low Range',
            r.hirange as 'High Range'
from        PublishersDatabase.titles as t
left outer join PublishersDatabase.roysched as r
on    t.title_id = r.title_id
where       type like 'popular_comp'
```

Exercise 2

- Use PublishersDatabase
- Write a query to select the store name, order date, quantity ordered and titles of all business books ordered by all stores.

union

The union operator allows you to combine the results of two or more select statements into one result set

The result sets that are combined must have the same structure

The sets must have the same number of columns, and the columns must have compatible data types

union

Table 1	
Col A – char(4)	Col B - int
Abc	12
Def	34

Table 2	
Col C – char(4)	Col D – int
Efg	56
Hij	78

union

```
select * from Table1
union
select * from Table2
```

Abc	12	Table A
Def	34	
Efg	56	Table B
Hij	78	

union

- The union operator will remove all duplicates from the result set.
- union all will produce all duplicated rows

```
select * from Table1
union all
select * from Table2
```

* UNION ALL performs faster than UNION *

union Example

Build a list of companies and phone numbers for all the companies dealt with by Northwind. Look at the Customers, Shippers, and Suppliers.

Add a company type field to categorize the output. (124 rows)

union Example

```
select      CompanyName,   Phone,
            'Customer' as 'Company Type'
from        NorthwindTraders.dbo.Customers
union all
select      CompanyName,  Phone,
            'Shipper' as 'Company Type'
from        NorthwindTraders.dbo.Shippers
union all
select      CompanyName,  Phone,
            'Supplier' as 'Company Type'
from        NorthwindTraders.dbo.Suppliers
order by    CompanyName
```

Exercise 3

- Write a select query to list the first name and last name of all students and instructors from the ClassTrak database. Have a column for ‘Status’ which will be ‘Student’ or ‘Instructor’

Self Join

- Sometimes we need to join a table to itself.
- In that case we need to use 2 different aliases for the same table.
- Our join condition can also be a little more complex
- E.g. from Publishers Database, we want to obtain the first name and last name of all authors who live in the same state as at least one other author.

Self Join

```
Use publishersDatabase

Select distinct au1.au_fname,au1.au_lname,au1.[state]
from authors au1
join authors au2
on (au1.[state]=au2.[state])
and (au1.au_id != au2.au_id)
```

Optional JOIN Syntax

If we use “join” alone, it means inner join.

For outer join, we use “left join”, “right join” or “full join”

Use of Joins with Union

- We can combine joins with Unions, for example we may want a list of all customers and all suppliers who have interests in the same kind of products.
- We'll proceed in steps through the following exercises

Exercise 4

- Use the NorthwindTraders database
- Write a query to display the company name of all suppliers of beverages as well as the product name of the beverages they supply.

CompanyName	ProductName
Exotic Liquids	Chai
Exotic Liquids	Chang
Refrescos Americanas LTDA	Guaraná Fantástica
Bigfoot Breweries	Sasquatch Ale
Bigfoot Breweries	Steeleye Stout
Aux joyeux ecclésiastiques	Côte de Blaye
Aux joyeux ecclésiastiques	Chartreuse verte
Leka Trading	Ipoh Coffee
Bigfoot Breweries	Laughing Lumberjack Lager
Pavlova, Ltd.	Outback Lager
Plutzer Lebensmittelgroßmärkte AG	Rhönbräu Klosterbier
Karkki Oy	Lakkalikööri
Bigfoot Breweries	Romulan Ale

(13 rows affected)

Exercise 5

- Use the NorthwindTraders database
- Write a query to display the company name of all customers who have ordered beverages as well as the product name of the beverages they have ordered. Your results should not contain repetitions.

CompanyName	ProductName
Alfreds Futterkiste	Chartreuse verte
Alfreds Futterkiste	Lakkalikööri
Ana Trujillo Emparedados y helados	Outback Lager
Antonio Moreno Taquería	Chang
Antonio Moreno Taquería	Ipoh Coffee
Antonio Moreno Taquería	Rhönbräu Klosterbier
Antonio Moreno Taquería	Sasquatch Ale
Around the Horn	Chang
Around the Horn	Guaraná Fantástica
Around the Horn	Laughing Lumberjack Lager
Around the Horn	Outback Lager

(312 rows affected)

Exercise 6

- Combine the queries in exercises 4 and 5 into a single query where we display all companies having an interest in beverages. We'll assign a Type to distinguish between customers and suppliers.

CompanyName	ProductName	Type
Alfreds Futterkiste	Chartreuse verte	Customer
Alfreds Futterkiste	Lakkalikööri	Customer
Ana Trujillo Emparedados y helados	Outback Lager	Customer
Antonio Moreno Taquería	Chang	Customer
Antonio Moreno Taquería	Ipoh Coffee	Customer
Antonio Moreno Taquería	Rhönbräu Klosterbier	Customer
Antonio Moreno Taquería	Sasquatch Ale	Customer
Around the Horn	Chang	Customer
Around the Horn	Guaraná Fantástica	Customer
Around the Horn	Laughing Lumberjack Lager	Customer
Around the Horn	Outback Lager	Customer
Around the Horn	Steeleye Stout	Customer
Aux joyeux ecclésiastiques	Chartreuse verte	Supplier
Aux joyeux ecclésiastiques	Côte de Blaye	Supplier

Wolski	Zajazd	Guaraná Fantástica	Customer
Wolski	Zajazd	Rhönbräu Klosterbier	Customer
Wolski	Zajazd	Steeleye Stout	Customer

(325 rows affected)

Exercise 7

- Write a query to extract the names of all students (first_name + last_name) and the course description of all courses not assigned to a class. Your query should have a type column specifying whether the name/description belongs to a course or student

Results

Students/Courses not assigned to a class	Type
Andrew Russell	Student
Brana Basara	Student
Christopher Brooks	Student
David Tham	Student
David Tucker	Student
Devin Mulrooney	Student
James Parsons	Student
John Dale	Student
Matthew Ng	Student
Nathan Hamaluik	Student
Shane Barclay	Student
Tyler Smith	Student
c++	Courses
Internet Programming	Courses
Introduction to C/C++	Courses

(15 rows affected)

Union of Queries involving **Order by** clause

- We have previously seen that we can have a union of queries then order the *overall results* using an **order by** clause.
- What happens if each individual query already involves an **order by** clause?
- If the individual queries must have individual order by clauses, these queries must become subqueries (aka temporary tables)

Union of Queries involving **Order by** clause

- We are going to use the Publishers database
- Consider that we want to find the top 3 titles in terms of revenues for the year.

```
select top 3 title as 'Title',
        price*ytd_sales as 'Revenues'

from titles
order by price*ytd_sales desc
```

```
Title                                                                            Revenues
-------------------------------------------------------------------------------- ---------------------
But Is It User Friendly?                                                         201501.00
Fifty Years in Buckingham Palace Kitchens                                        180397.20
Secrets of Silicon Valley                                                        81900.00

(3 rows affected)
```

Union of Queries involving **Order by** clause

- Now let us say, we want the 3 stores from which we have got the most revenues.

```
select top 3 stor_name,
        sum(qty*price) as 'Total'
       from stores st
       inner join sales s
       on s.stor_id= st.stor_id
       inner join titles t
        on s.title_id=t.title_id
group by stor_name
order by sum(qty*price) desc
```

```
stor_name                                Total
---------------------------------------- ---------
Barnum's                                 1821.25
News & Brews                             1486.30
Doc-U-Mat: Quality Laundry and Books     1400.15

(3 rows affected)
```

Union of Queries involving **Order by** clause

- Now, we want to combine both

```
Use publishersdatabase

Select 'Title' as 'Category', t.Title as 'Title/Store Name', t.Revenues as 'Total Revenues'
from
(
select top 3 title as Title, price*ytd_sales as Revenues
from titles order by price*ytd_sales desc) as t

Union

Select 'Stores' as 'Category', st.[store name] as 'Title/Store Name',
        st.Total as 'Total Revenues'
from
(select top 3 stor_name as [Store Name],
       sum(qty*price) as Total   from stores st
      inner join sales s on s.stor_id= st.stor_id
      inner join titles t  on s.title_id=t.title_id
group by stor_name
order by sum(qty*price) desc) as st

order by Category desc
```

Results of the query

Category	Title/Store Name	Total Revenues
Title	But Is It User Friendly?	201501.00
Title	Fifty Years in Buckingham Palace Kitchens	180397.20
Title	Secrets of Silicon Valley	81900.00
Stores	Barnum's	1821.25
Stores	Doc-U-Mat: Quality Laundry and Books	1400.15
Stores	News & Brews	1486.30

(6 rows affected)