Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

SQL Commands

Joins and it's Types

Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales

www.uwtsd.ac.uk

Objectives

Relational Algebra Operators

- Joins
- Implementation in SQL

Cartesian Product

- result of combining all rows from one table with all rows from another table
- cross join (X)

Customers Table:

CustomerID	CustomerName
1	Alice
2	Bob
3	Charlie

Orders Table:

OrderID	OrderDate
101	2023-01-10
102	2023-01-15
103	2023-01-20

Cartesian Product

- result of combining all rows from one table with all rows from another table
- cross join (X)

Customers Table:

CustomerID	CustomerName	OrderID	OrderDate
1	Alice	101	2023-01-10
		102	2023-01-15
		103	2023-01-20
2	Bob	101	2023-01-10
		102	2023-01-15
		103	2023-01-20
3	Charlie		

Cartesian Product

- result of combining all rows from one table with all rows from another table
- cross join (X)

CustomerID	CustomerName	OrderID	OrderDate
1	Alice	101	2023-01-10
1	Alice	102	2023-01-15
1	Alice	103	2023-01-20
2	Bob	101	2023-01-10
2	Bob	102	2023-01-15
2	Bob	103	2023-01-20
3	Charlie	101	2023-01-10
3	Charlie	102	2023-01-15
3	Charlie	103	2023-01-20

Cartesian Product

Venues Table:

VenueID	VenueName	Capacity
1	Grand Ballroom	200
2	Garden Pavilion	100
3	Lakeside Lodge	150

Catering Options Table:

CateringID	Caterer	Cuisine
101	Gourmet Delight	Italian
102	Taste of India	Indian
103	Savory Bites	Mediterranean

Cartesian Product

Result (Cartesian Product):

VenueID	VenueName	Capacity	CateringID	Caterer	Cuisine
1	Grand Ballroom	200	101	Gourmet Delight	Italian
1	Grand Ballroom	200	102	Taste of India	Indian
1	Grand Ballroom	200	103	Savory Bites	Mediterranean
2	Garden Pavilion	100	101	Gourmet Delight	Italian
2	Garden Pavilion	100	102	Taste of India	Indian
2	Garden Pavilion	100	103	Savory Bites	Mediterranean
3	Lakeside Lodge	150	101	Gourmet Delight	Italian
3	Lakeside Lodge	150	102	Taste of India	Indian
3	Lakeside Lodge	150	103	Savory Bites	Mediterranean

Cartesian Product - SQL

Employees Table:

EmployeeID	EmployeeName
1	Alice
2	Bob
3	Charlie

Departments Table:

DepartmentID	DepartmentName
101	Sales
102	Marketing

Cartesian Product - SQL

Employees Table:

EmployeeID	EmployeeName
1	Alice
2	Bob
3	Charlie

Departments Table:

DepartmentID	DepartmentName
101	Sales
102	Marketing

```
-- Create the Employees table
CREATE TABLE Employees (
   EmployeeID serial PRIMARY KEY,
   EmployeeName VARCHAR(255)
);

-- Insert sample data into the Employees table
INSERT INTO Employees (EmployeeName) VALUES
   ('Alice'),
   ('Bob'),
   ('Charlie');

-- Create the Departments table
CREATE TABLE Departments (
   DepartmentID serial PRIMARY KEY,
   DepartmentName VARCHAR(255)
);

-- Insert sample data into the Departments table
INSERT INTO Departments (DepartmentName) VALUES
   ('Sales'),
   ('Marketing');
```

Cartesian Product - SQL

Result (Cartesian Product):

EmployeeID	EmployeeName	DepartmentID	DepartmentName
1	Alice	101	Sales
1	Alice	102	Marketing
2	Bob	101	Sales
2	Bob	102	Marketing
3	Charlie	101	Sales
3	Charlie	102	Marketing

Cartesian Product - SQL

Result (Cartesian Product):

EmployeeID	EmployeeName	DepartmentID	DepartmentName
1	Alice	101	Sales
1	Alice	102	Marketing
2	Bob	101	Sales
2	Bob	102	Marketing
3	Charlie	101	Sales
3	Charlie	102	Marketing

```
SELECT *  FROM Employees, Departments;
```

Joins

- Cartesian product creates mess results.
- Need to ensure that they are done using specific conditions.
- **Join** operation does it for us.
- Join is a **derivative of Cartesian product**, equivalent to performing a **Selection** operation, using the join predicate as the selection formula, **over the Cartesian product** of the two operand relations.

Types of Joins

- There are various forms of Join operation, each with subtle differences, some more useful than others:
- Theta join
- Equi-join (a particular type of Theta join)
- Natural join
- Outer join

  - Our Area of Focus
- Semi-join

Natural Join ⋈

- R ⋈ S The Natural join is an **Equijoin** of the two relations R and S over **all common attributes *x***.

Sells		
Bar	**Beer**	**Price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars	
Bar	**Address**
Joe's	Maple St.
Sue's	River Rd.

Natural Join ⋈ - Step 1 Perform X Join

Sells		
Bar	**Beer**	**Price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

X

Bars	
Bar	**Address**
Joe's	Maple St.
Sue's	River Rd.

Natural Join ⋈ - Step 1 Perform X Join

Sells		
Bar	**Beer**	**Price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars	
Bar	**Address**
Joe's	Maple St.
Sue's	River Rd.

BarInfo – Step 1				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Joe's	Maple St.
Sue's	Coors	3.00	Joe's	Maple St.
Joe's	Bud	2.50	Sue's	River Rd.
Joe's	Miller	2.75	Sue's	River Rd.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Natural Join ⋈ - Step 2 Apply selection

Sells		
Bar	**Beer**	**Price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars	
Bar	**Address**
Joe's	Maple St.
Sue's	River Rd.

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
~~Sue's~~	~~Bud~~	~~2.50~~	~~Joe's~~	~~Maple St.~~
~~Sue's~~	~~Coors~~	~~3.00~~	~~Joe's~~	~~Maple St.~~
~~Joe's~~	~~Bud~~	~~2.50~~	~~Sue's~~	~~River Rd.~~
~~Joe's~~	~~Miller~~	~~2.75~~	~~Sue's~~	~~River Rd.~~
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Natural Join ⋈ - Step 2 Apply selection

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
~~Sue's~~	~~Bud~~	~~2.50~~	~~Joe's~~	~~Maple St.~~
~~Sue's~~	~~Coors~~	~~3.00~~	~~Joe's~~	~~Maple St.~~
~~Joe's~~	~~Bud~~	~~2.50~~	~~Sue's~~	~~River Rd.~~
~~Joe's~~	~~Miller~~	~~2.75~~	~~Sue's~~	~~River Rd.~~
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Natural Join ⋈ - Step 3 Remove duplicates

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
~~Sue's~~	~~Bud~~	~~2.50~~	~~Joe's~~	~~Maple St.~~
~~Sue's~~	~~Coors~~	~~3.00~~	~~Joe's~~	~~Maple St.~~
~~Joe's~~	~~Bud~~	~~2.50~~	~~Sue's~~	~~River Rd.~~
~~Joe's~~	~~Miller~~	~~2.75~~	~~Sue's~~	~~River Rd.~~
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Natural Join

Sells		
Bar	**Beer**	**Price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars	
Bar	**Address**
Joe's	Maple St.
Sue's	River Rd.

$$\text{BarInfo} = \text{Sells} \bowtie \text{Bars}$$

BarInfo			
Bar	**Beer**	**Price**	**Address**
Joe's	Bud	2.50	Maple St.
Joe's	Miller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

Natural Join – How ?

BarInfo – Step 1				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Joe's	Maple St.
Sue's	Coors	3.00	Joe's	Maple St.
Joe's	Bud	2.50	Sue's	River Rd.
Joe's	Miller	2.75	Sue's	River Rd.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

BarInfo – Step 2				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

BarInfo – Step 3			
Bar	**Beer**	**Price**	**Address**
Joe's	Bud	2.50	Maple St.
Joe's	Miller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

Natural Join – How ? Steps

BarInfo				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
				St.
Joe's	Bud	2.50	Sue's	River Rd.
Joe's	Miller	2.75	Sue's	River Rd.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Step 1: BarInfo = Sells x Bars

BarInfo				
Bar	**Beer**	**Price**	**Bar**	**Address**
Joe's	Bud	2.50	Joe's	Maple
				Rd.
Sue's	Coors	3.00	Sue's	River Rd.

Step 2: BarInfo = $\sigma_{\text{Sells.Bar=Bars.Bar}}$(BarInfo)

BarInfo			
Bar	**Beer**	**Price**	**Address**
Joe's	Bud	2.50	Maple St.
Sue's	Coors	3.00	River Rd.

Step 3: $\pi_{\text{Bar,Beer,Price,Address}}$(BarInfo)

Natural Join – SQL

Try to perform natural join with SQL Query.

Example:

Query: *List the names of all clients who have viewed a property, along with any comments supplied.*

SELECT clientNo, fName, lName, propertyNo, comment

FROM Client **NATURAL JOIN** Viewing;

Example:

Query: *List the names of all clients who have viewed a property, along with any comments supplied.*

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

- Sorting a join

Query: *For each branch office, list the staff numbers and names of staff who manage properties and the properties that they manage.*

```
SELECT s.branchNo, s.staffNo, fName, lName, propertyNo
FROM (Branch b JOIN Staff s USING branchNo) AS bs
        JOIN PropertyForRent p USING staffNo
```

- Sorting a join

Query: *For each branch office, list the staff numbers and names of staff who manage properties and the properties that they manage.*

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

- **Three-table join**

Query: *For each branch, list the staff numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manage.*

- The result table requires columns from three tables:
- Branch (branchNo and city), Staff (staffNo, fName and lName), and PropertyForRent (propertyNo)
- Branch and Staff details are joined using the condition

(b.branchNo = s.branchNo)

- Staff and PropertyForRent details are joined using the condition

(s.staffNo = p.staffNo)

- **Three-table join**

Query: *For each branch, list the staff numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manage.*

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo
FROM (Branch b JOIN Staff s USING branchNo) AS bs
        JOIN PropertyForRent p USING staffNo
```

- **Three-table join**

Query: *For each branch, list the staff numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manage.*

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

Outer Join

- In outer join, matched pairs are retained

unmatched values in other tables are omitted.

- More widely available in relational systems.
- Information is preserved, that is, the Outer join preserves tuples that would have been lost by other types of join.
- Types:
    - Full Outer Join: R ⟗ S
    - Left Outer Join: R ⟕ S
    - Right Outer Join: R ⟖ S

Outer Join – Types - Example

R	
Emp	**Dept**
Smith	Sales
Black	Production
White	Production

S	
Dept	**Head**
Production	Mori
Purchasing	Brown

R ⟕ S (left outer join)		
Emp	Dept	Head
Smith	Sales	Null
Black	Production	Mori
White	Production	Mori

R	
Emp	**Dept**
Smith	Sales
Black	Production
White	Production

S	
Dept	**Head**
Production	Mori
Purchasing	Brown

R ⟖ S (right outer join)		
Emp	Dept	Head
Black	Production	Mori
White	Production	Mori
Null	Purchasing	Brown

R	
Emp	**Dept**
Smith	Sales
Black	Production
White	Production

S	
Dept	**Head**
Production	Mori
Purchasing	Brown

R ⟗ S (full outer join)		
Emp	Dept	Head
Smith	Sales	Null
Black	Production	Mori
White	Production	Mori
Null	Purchasing	Brown

Left Outer Join – Types - SQL

Enrollments Table:

EnrollmentID	StudentID	CourseName
1	1	Mathematics
2	2	Physics
3	1	Chemistry
4	3	Biology

Students Table:

StudentID	FirstName	LastName
1	John	Doe
2	Alice	Smith
3	Bob	Johnson

Left Outer Join – Types - SQL

Enrollments Table:

EnrollmentID	StudentID	CourseName
1	1	Mathematics
2	2	Physics
3	1	Chemistry
4	3	Biology

Students Table:

StudentID	FirstName	LastName
1	John	Doe
2	Alice	Smith
3	Bob	Johnson

```
SELECT Students.FirstName, Students.LastName, Enrollments.CourseName
FROM
Students LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

Right Outer Join – Types - SQL

Enrollments Table:

EnrollmentID	StudentID	CourseName
1	1	Mathematics
2	2	Physics
3	1	Chemistry
4	3	Biology
5	4	History

Students Table:

StudentID	FirstName	LastName
1	John	Doe
2	Alice	Smith
3	Bob	Johnson

```
SELECT Students.FirstName, Students.LastName, Enrollments.CourseName
FROM
Students RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

Full Outer Join – Types - SQL

Enrollments Table:

EnrollmentID	StudentID	CourseName
1	1	Mathematics
2	2	Physics
3	1	Chemistry
4	3	Biology
5	4	History

Students Table:

StudentID	FirstName	LastName
1	John	Doe
2	Alice	Smith
3	Bob	Johnson

```
SELECT Students.FirstName, Students.LastName, Enrollments.CourseName
FROM
Students FULL OUTER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

Practice exercises

- Consider the following two simplified Branch and PropertyForRent tables, which we refer to as **Branch1** and **PropertyForRent1**, respectively:

Branch1

branchNo	**bCity**
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	**pCity**
PA14	Aberdeen
PL94	London
PG4	Glasgow

The (Inner) join of these two tables:

```
SELECT b.*, p.*
FROM Branch1 b, PropertyForRent1 p
WHERE b.bCity = p.pCity;
```

The result table has two rows where the cities are the same. In particular, note
that there is no row corresponding to the branch office in Bristol and there is
no row corresponding to the property in Aberdeen

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

- Left Outer join

Query: List all branch offices and any properties that are in the same cit

SELECT b.*, p.*

FROM Branch1 b **LEFT JOIN** PropertyForRent1 p **ON**

b.bCity = p.pCity; y.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

- **Right Outer join**

Query: *List all properties and any branch offices that are in the same city.*

SELECT b.*, p.*

FROM Branch1 b **RIGHT JOIN** PropertyForRent1 p **ON** b.bCity = p.pCity;

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

- **Full Outer join**
- **Query:** *List the branch offices and properties that are in the same city along with any unmatched branches or properties.*

SELECT b.*, p.*

FROM Branch1 b **FULL JOIN** PropertyForRent1 p **ON** b.bCity = p.pCity;

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

Further reading

- Chapter 6 from Connolly,T.,&begg,C.(2005). Database systems: a practical approach to design, Implementation and management, Boston, MA: pearson Addison-Wesley

Next Session

ACID Properties