Chapter 4

How to retrieve data from two tables

© 2008, Mike Murach & Associates, Inc.

Objectives

Applied

- Use the explicit syntax to code an inner join that returns data from a single table or multiple tables.
- Use the explicit syntax to code an outer join.
- Code a union that combines data from a single table or multiple tables.

© 2008, Mike Murach & Associates, Inc.

Objectives (continued)

Knowledge

- Explain when column names need to be qualified.
- Describe the proper use of a table alias.
- Describe the differences between an inner join, a left outer join, a right outer join, a full outer join, and a cross join.
- Explain why you don’t need to use right outer joins.
- Describe the use of the implicit syntax for coding joins.
- Describe the use of the USING and NATURAL keywords for coding joins.
- Describe the use of unions including the use of the MINUS and INTERSECT operators.

© 2008, Mike Murach & Associates, Inc.

The explicit syntax for an inner join

```
SELECT select_list
FROM table_1
    [INNER] JOIN table_2
        ON join_condition_1
   [[INNER] JOIN table_3
        ON join_condition_2]...
```

A SELECT statement that joins two tables

```
SELECT invoice_number, vendor_name
FROM vendors INNER JOIN invoices
    ON vendors.vendor_id = invoices.vendor_id
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	VENDOR_NAME
1	0-2058	Malloy Lithographing Inc
2	0-2060	Malloy Lithographing Inc
3	0-2436	Malloy Lithographing Inc
4	1-200-5164	Federal Express Corporation

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

The syntax for an inner join that uses table aliases

```
SELECT select_list
FROM table_1 n1
    [INNER] JOIN table_2 n2
        ON n1.column_name operator n2.column_name
   [[INNER] JOIN table_3 n3
        ON n2.column_name operator n3.column_name]...
```


© 2008, Mike Murach & Associates, Inc.

An inner join with aliases for all tables

```
SELECT invoice_number, vendor_name, invoice_due_date,
    (invoice_total - payment_total - credit_total)
        AS balance_due
FROM vendors v JOIN invoices i
    ON v.vendor_id = i.vendor_id
WHERE (invoice_total - payment_total - credit_total) > 0
ORDER BY invoice_due_date DESC
```

The result set

	INVOICE_NUMBER	VENDOR_NAME	INVOICE_DUE_DATE	BALANCE_DUE
1	40318	Data Reproductions Corp	20-JUL-08	21842
2	39104	Data Reproductions Corp	20-JUL-08	85.31
3	0-2436	Malloy Lithographing Inc	17-JUL-08	10976.06

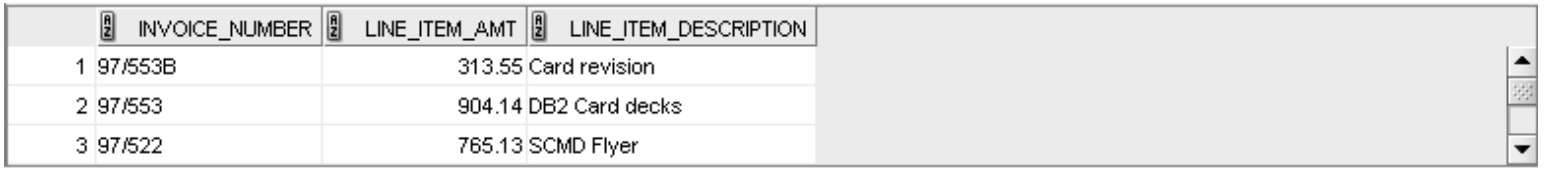
(40 rows selected)

© 2008, Mike Murach & Associates, Inc.

An inner join with an alias for only one table

```
SELECT invoice_number, line_item_amt,
    line_item_description
FROM invoices JOIN invoice_line_items line_items
    ON invoices.invoice_id = line_items.invoice_id
WHERE account_number = 540
ORDER BY invoice_date
```

The result set



	INVOICE_NUMBER	LINE_ITEM_AMT	LINE_ITEM_DESCRIPTION
1	97/553B	313.55	Card revision
2	97/553	904.14	DB2 Card decks
3	97/522	765.13	SCMD Flyer

```
(8 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

The syntax of a table name that's qualified with a schema name

```
schema_name.table_name
```

A SQL statement that grants the SELECT permission in the OM schema to the AP schema

```
GRANT SELECT ON customers TO ap
```


© 2008, Mike Murach & Associates, Inc.

A join with a table from another schema

```
SELECT vendor_name, customer_last_name,
    customer_first_name, vendor_state AS state,
    vendor_city AS city
FROM vendors v
    JOIN om.customers c
    ON v.vendor_zip_code = c.customer_zip
ORDER BY state, city
```

The result set

	VENDOR_NAME	CUSTOMER_LAST_NAME	CUSTOMER_FIRST_NAME	STATE	CITY
1	Wells Fargo Bank	Marissa	Kyle	AZ	Phoenix
2	Aztek Label	Irvin	Ania	CA	Anaheim
3	Lou Gentile's Flower Basket	Damien	Deborah	CA	Fresno
4	Shields Design	Damien	Deborah	CA	Fresno
5	Costco	Neftaly	Thalia	CA	Fresno
6	Costco	Holbrooke	Rashad	CA	Fresno
7	Gary McKeighan Insurance	Holbrooke	Rashad	CA	Fresno
8	Zylka Design	Neftaly	Thalia	CA	Fresno
9	Zylka Design	Holbrooke	Rashad	CA	Fresno

(37 rows)

© 2008, Mike Murach & Associates, Inc.

An inner join with two conditions

```
SELECT invoice_number, invoice_date,
    invoice_total, line_item_amt
FROM invoices i JOIN invoice_line_items li
    ON (i.invoice_id = li.invoice_id) AND
       (i.invoice_total > li.line_item_amt)
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL	LINE_ITEM_AMT
1	97/522	30-APR-08	1962.13	765.13
2	97/522	30-APR-08	1962.13	1197
3	I77271-O01	05-JUN-08	662	75.6
4	I77271-O01	05-JUN-08	662	58.4

(6 rows selected)

© 2008, Mike Murach & Associates, Inc.

The same join with one condition in a WHERE clause

```
SELECT invoice_number, invoice_date,
    invoice_total, line_item_amt
FROM invoices i JOIN invoice_line_items li
    ON i.invoice_id = li.invoice_id
WHERE i.invoice_total > li.line_item_amt
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL	LINE_ITEM_AMT
1	97/522	30-APR-08	1962.13	765.13
2	97/522	30-APR-08	1962.13	1197
3	I77271-O01	05-JUN-08	662	75.6
4	I77271-O01	05-JUN-08	662	58.4

```
(6 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

A self-join that returns vendors from cities in common with other vendors

```
SELECT DISTINCT v1.vendor_name, v1.vendor_city,
    v1.vendor_state
FROM vendors v1 JOIN vendors v2
    ON (v1.vendor_city = v2.vendor_city) AND
       (v1.vendor_state = v2.vendor_state) AND
       (v1.vendor_id <> v2.vendor_id)
ORDER BY v1.vendor_state, v1.vendor_city
```

The result set

	VENDOR_NAME	VENDOR_CITY	VENDOR_STATE
1	AT&T	Phoenix	AZ
2	Computer Library	Phoenix	AZ
3	Wells Fargo Bank	Phoenix	AZ
4	Aztek Label	Anaheim	CA
5	Blue Shield of California	Anaheim	CA
6	ASC Signs	Fresno	CA
7	Abbey Office Furnishings	Fresno	CA
8	BFI Industries	Fresno	CA

```
(84 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

A SELECT statement that joins four tables

```
SELECT vendor_name, invoice_number, invoice_date,
    line_item_amt, account_description
FROM vendors v
    JOIN invoices i ON v.vendor_id = i.vendor_id
    JOIN invoice_line_items li
        ON i.invoice_id = li.invoice_id
    JOIN general_ledger_accounts gl
        ON li.account_number = gl.account_number
WHERE (invoice_total - payment_total - credit_total) > 0
ORDER BY vendor_name, line_item_amt DESC
```

The result set

	VENDOR_NAME	INVOICE_NUMBER	INVOICE_DATE	LINE_ITEM_AMT	ACCOUNT_DESCRIPTION
1	Abbey Office Furnishings	203339-13	02-MAY-08	17.5	Office Supplies
2	Blue Cross	547481328	20-MAY-08	224	Group Insurance
3	Blue Cross	547480102	19-MAY-08	224	Group Insurance
4	Blue Cross	547479217	17-MAY-08	116	Group Insurance
5	Cardinal Business Media, Inc.	134116	01-JUN-08	90.36	Card Deck Advertising
6	Coffee Break Service	109596	14-JUN-08	41.8	Meals
7	Compuserve	21-4748363	09-MAY-08	9.95	Books, Dues, and Subscriptions
8	Computerworld	367447	31-MAY-08	2433	Card Deck Advertising

(44 rows selected)

© 2008, Mike Murach & Associates, Inc.

The implicit syntax for an inner join

```
SELECT select_list
FROM table_1, table_2 [, table_3]...
WHERE table_1.column_name operator table_2.column_name
  [AND table_2.column_name operator table_3.column_name]...
```

Implicit syntax that joins two tables

```
SELECT invoice_number, vendor_name
FROM vendors v, invoices i
WHERE v.vendor_id = i.vendor_id
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	VENDOR_NAME
1	0-2058	Malloy Lithographing Inc
2	0-2060	Malloy Lithographing Inc
3	0-2436	Malloy Lithographing Inc
4	1-200-5164	Federal Express Corporation
5	1-202-2978	Federal Express Corporation

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

Implicit syntax that joins four tables

```
SELECT vendor_name, invoice_number, invoice_date,
    line_item_amt, account_description
FROM  vendors v, invoices i, invoice_line_items  li,
    general_ledger_accounts gl
WHERE v.vendor_id = i.vendor_id
  AND i.invoice_id = li.invoice_id
  AND li.account_number = gl.account_number
  AND (invoice_total - payment_total - credit_total) > 0
ORDER BY vendor_name, line_item_amt DESC
```

The result set

	VENDOR_NAME	INVOICE_NUMBER	INVOICE_DATE	LINE_ITEM_AMT	ACCOUNT_DESCRIPTION
1	Abbey Office Furnishings	203339-13	02-MAY-08	17.5	Office Supplies
2	Blue Cross	547480102	19-MAY-08	224	Group Insurance
3	Blue Cross	547481328	20-MAY-08	224	Group Insurance
4	Blue Cross	547479217	17-MAY-08	116	Group Insurance
5	Cardinal Business Media, Inc.	134116	01-JUN-08	90.36	Card Deck Advertising

```
(44 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

Terms to know

- Join
- Join condition
- Inner join
- Ad hoc relationship
- Qualified column name
- Table alias
- Self join
- Explicit syntax (SQL-92)
- Implicit syntax

© 2008, Mike Murach & Associates, Inc.

The explicit syntax for an outer join

```
SELECT select_list
FROM table_1
    {LEFT|RIGHT|FULL} [OUTER] JOIN table_2
        ON join_condition_1
   [{LEFT|RIGHT|FULL} [OUTER] JOIN table_3
        ON join_condition_2]...
```

What outer joins do

Join	Keeps unmatched rows from
Left	The left table
Right	The right table
Full	Both tables

© 2008, Mike Murach & Associates, Inc.

A SELECT statement that uses a left outer join

```
SELECT vendor_name, invoice_number, invoice_total
FROM vendors LEFT JOIN invoices
    ON vendors.vendor_id = invoices.vendor_id
ORDER BY vendor_name
```

The result set

	VENDOR_NAME	INVOICE_NUMBER	INVOICE_TOTAL
1	ASC Signs	(null)	(null)
2	AT&T	(null)	(null)
3	Abbey Office Furnishings	203339-13	17.5
4	American Booksellers Assoc	(null)	(null)
5	American Express	(null)	(null)

(202 rows selected)

© 2008, Mike Murach & Associates, Inc.

The Departments table

DEPARTMENT_NUMBER	DEPARTMENT_NAME
1	Accounting
2	Payroll
3	Operations
4	Personnel
5	Maintenance

The Employees table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NUMBER
1	Cindy	Smith	2
2	Elmer	Jones	4
3	Ralph	Simonian	2
4	Olivia	Hernandez	1
5	Robert	Aaronsen	2
6	Denise	Watson	6
7	Thomas	Hardy	5
8	Rhea	O'Leary	4
9	Paulo	Locario	6

© 2008, Mike Murach & Associates, Inc.

A left outer join

```
SELECT department_name AS dept_name,
    d.department_number AS dept_no,
    last_name
FROM departments d
    LEFT JOIN employees e
    ON d.department_number = e.department_number
ORDER BY department_name
```

	DEPT_NAME	DEPT_NO	LAST_NAME
1	Accounting	1	Hernandez
2	Maintenance	5	Hardy
3	Operations	3	(null)
4	Payroll	2	Smith
5	Payroll	2	Simonian
6	Payroll	2	Aaronsen
7	Personnel	4	Jones
8	Personnel	4	O'Leary

© 2008, Mike Murach & Associates, Inc.

A right outer join

```
SELECT department_name AS dept_name,
    e.department_number AS dept_no,
    last_name
FROM departments d
    RIGHT JOIN employees e
    ON d.department_number = e.department_number
ORDER BY department_name
```

	DEPT_NAME	DEPT_NO	LAST_NAME
1	Accounting	1	Hernandez
2	Maintenance	5	Hardy
3	Payroll	2	Aaronsen
4	Payroll	2	Simonian
5	Payroll	2	Smith
6	Personnel	4	Jones
7	Personnel	4	O'Leary
8	(null)	6	Locario
9	(null)	6	Watson

© 2008, Mike Murach & Associates, Inc.

A full outer join

```
SELECT department_name AS dept_name,
    d.department_number AS d_dept_no,
    e.department_number AS e_dept_no,
    last_name
FROM departments d
    FULL JOIN employees e
    ON d.department_number = e.department_number
ORDER BY department_name
```

	DEPT_NAME	D_DEPT_NO	E_DEPT_NO	LAST_NAME
1	Accounting	1	1	Hernandez
2	Maintenance	5	5	Hardy
3	Operations	3	(null)	(null)
4	Payroll	2	2	Simonian
5	Payroll	2	2	Smith
6	Payroll	2	2	Aaronsen
7	Personnel	4	4	O'Leary
8	Personnel	4	4	Jones
9	(null)	(null)	6	Watson
10	(null)	(null)	6	Locario

© 2008, Mike Murach & Associates, Inc.

The Departments table

DEPARTMENT_NUMBER	DEPARTMENT_NAME
1	Accounting
2	Payroll
3	Operations
4	Personnel
5	Maintenance

The Employees table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NUMBER
1	Cindy	Smith	2
2	Elmer	Jones	4
3	Ralph	Simonian	2
4	Olivia	Hernandez	1
5	Robert	Aaronsen	2
6	Denise	Watson	6
7	Thomas	Hardy	5
8	Rhea	O'Leary	4
9	Paulo	Locario	6

© 2008, Mike Murach & Associates, Inc.

The Projects table

PROJECT_NUMBER	EMPLOYEE_ID
P1011	8
P1011	4
P1012	3
P1012	1
P1012	5
P1013	6
P1013	9
P1014	10

© 2008, Mike Murach & Associates, Inc.

A SELECT statement that uses left outer joins

```
SELECT department_name, last_name, project_number AS proj_no
FROM departments d
    LEFT JOIN employees e
        ON d.department_number = e.department_number
    LEFT JOIN projects p
        ON e.employee_id = p.employee_id
ORDER BY department_name, last_name, project_number
```

	DEPARTMENT_NAME	LAST_NAME	PROJ_NO
1	Accounting	Hernandez	P1011
2	Maintenance	Hardy	(null)
3	Operations	(null)	(null)
4	Payroll	Aaronsen	P1012
5	Payroll	Simonian	P1012
6	Payroll	Smith	P1012
7	Personnel	Jones	(null)
8	Personnel	O'Leary	P1011

© 2008, Mike Murach & Associates, Inc.

A SELECT statement that uses full outer joins

```
SELECT department_name, last_name, project_number AS proj_no
FROM departments dpt
    FULL JOIN employees emp
        ON dpt.department_number = emp.department_number
    FULL JOIN projects prj
        ON emp.employee_id = prj.employee_id
ORDER BY department_name
```

	DEPARTMENT_NAME	LAST_NAME	PROJ_NO
1	Accounting	Hernandez	P1011
2	Maintenance	Hardy	(null)
3	Operations	(null)	(null)
4	Payroll	Simonian	P1012
5	Payroll	Smith	P1012
6	Payroll	Aaronsen	P1012
7	Personnel	Jones	(null)
8	Personnel	O'Leary	P1011
9	(null)	Locario	P1013
10	(null)	(null)	P1014
11	(null)	Watson	P1013

© 2008, Mike Murach & Associates, Inc.

The implicit syntax for an outer join

```
SELECT select_list
FROM table_1, table_2 [, table 3]...
WHERE table_1.column_name [(+)] table_2.column_name [(+)]
    [table_2.column_name [(+)] table_3.column_name [(+)]]...
```


© 2008, Mike Murach & Associates, Inc.

Implicit syntax with a left outer join

```
SELECT department_name AS dept_name,
    dpt.department_number AS dept_no,
    last_name
FROM departments dpt, employees emp
WHERE dpt.department_number = emp.department_number (+)
ORDER BY department_name
```

The result set

	DEPT_NAME	DEPT_NO	LAST_NAME
1	Accounting	1	Hernandez
2	Maintenance	5	Hardy
3	Operations	3	(null)
4	Payroll	2	Simonian
5	Payroll	2	Aaronsen
6	Payroll	2	Smith
7	Personnel	4	Jones
8	Personnel	4	O'Leary

```
(8 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

Implicit syntax with a right outer join

```
SELECT department_name AS dept_name,
    emp.department_number AS dept_no,
    last_name
FROM departments dpt, employees emp
WHERE dpt.department_number (+) = emp.department_number
ORDER BY department_name
```

The result set

	DEPT_NAME	DEPT_NO	LAST_NAME
1	Accounting	1	Hernandez
2	Maintenance	5	Hardy
3	Payroll	2	Aaronsen
4	Payroll	2	Simonian
5	Payroll	2	Smith
6	Personnel	4	Jones
7	Personnel	4	O'Leary
8	(null)	6	Locario
9	(null)	6	Watson

```
(9 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

The Departments table

DEPARTMENT_NUMBER	DEPARTMENT_NAME
1	Accounting
2	Payroll
3	Operations
4	Personnel
5	Maintenance

The Employees table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NUMBER
1	Cindy	Smith	2
2	Elmer	Jones	4
3	Ralph	Simonian	2
4	Olivia	Hernandez	1
5	Robert	Aaronsen	2
6	Denise	Watson	6
7	Thomas	Hardy	5
8	Rhea	O'Leary	4
9	Paulo	Locario	6

© 2008, Mike Murach & Associates, Inc.

The Projects table

PROJECT_NUMBER	EMPLOYEE_ID
P1011	8
P1011	4
P1012	3
P1012	1
P1012	5
P1013	6
P1013	9
P1014	10

© 2008, Mike Murach & Associates, Inc.

A SELECT statement with an outer and inner join

```
SELECT department_name AS dept_name,
       last_name, project_number
FROM departments dpt
    JOIN employees emp
        ON dpt.department_number = emp.department_number
    LEFT JOIN projects prj
        ON emp.employee_id = prj.employee_id
ORDER BY department_name
```

The result set

	DEPT_NAME	LAST_NAME	PROJECT_NUMBER
1	Accounting	Hernandez	P1011
2	Maintenance	Hardy	(null)
3	Payroll	Simonian	P1012
4	Payroll	Smith	P1012
5	Payroll	Aaronsen	P1012
6	Personnel	Jones	(null)
7	Personnel	O'Leary	P1011

```
(7 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

The syntax for a join with the USING keyword

```
SELECT select_list
FROM table_1
    [{LEFT|RIGHT|FULL} [OUTER]] JOIN table_2
        USING(join_column_1[, join_column_2]...)
   [[{LEFT|RIGHT|FULL} [OUTER]] JOIN table_3
        USING (join_column_2[, join_column_2]...)]...
```

A SELECT statement with the USING keyword

```
SELECT invoice_number, vendor_name
FROM vendors
    JOIN invoices USING (vendor_id)
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	VENDOR_NAME
1	0-2058	Malloy Lithographing Inc
2	0-2060	Malloy Lithographing Inc
3	0-2436	Malloy Lithographing Inc
4	1-200-5164	Federal Express Corporation

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

A SELECT statement with the USING keyword

```
SELECT department_name AS dept_name, last_name,
project_number
FROM departments
    JOIN employees USING (department_number)
    LEFT JOIN projects USING (employee_id)
ORDER BY department_name
```

The result set

	DEPT_NAME	LAST_NAME	PROJECT_NUMBER
1	Accounting	Hernandez	P1011
2	Maintenance	Hardy	(null)
3	Payroll	Simonian	P1012
4	Payroll	Smith	P1012
5	Payroll	Aaronsen	P1012
6	Personnel	Jones	(null)
7	Personnel	O'Leary	P1011

(7 rows selected)

© 2008, Mike Murach & Associates, Inc.

The syntax for a join with the NATURAL keyword

```
SELECT select_list
FROM table_1
    NATURAL JOIN table_2
    [NATURAL JOIN table_3]...
```

A SELECT statement with the NATURAL keyword

```
SELECT invoice_number, vendor_name
FROM vendors
    NATURAL JOIN invoices
ORDER BY invoice_number
```

The result set

	INVOICE_NUMBER	VENDOR_NAME
1	0-2058	Malloy Lithographing Inc
2	0-2060	Malloy Lithographing Inc
3	0-2436	Malloy Lithographing Inc
4	1-200-5164	Federal Express Corporation

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

A SELECT statement with the NATURAL keyword

```
SELECT department_name AS dept_name, last_name,
    project_number
FROM departments
    NATURAL JOIN employees
    LEFT JOIN projects USING (employee_id)
ORDER BY department_name
```

The result set

	DEPT_NAME	LAST_NAME	PROJECT_NUMBER
1	Accounting	Hernandez	P1011
2	Maintenance	Hardy	(null)
3	Payroll	Simonian	P1012
4	Payroll	Smith	P1012
5	Payroll	Aaronsen	P1012
6	Personnel	Jones	(null)
7	Personnel	O'Leary	P1011

```
(7 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

How to code a cross join with the explicit syntax

The explicit syntax for a cross join

```
SELECT select_list
FROM table_1 CROSS JOIN table_2
```

A cross join that uses the explicit syntax

```
SELECT departments.department_number, department_name,
    employee_id, last_name
FROM departments CROSS JOIN employees
ORDER BY departments.department_number
```

The result set

	DEPARTMENT_NUMBER	DEPARTMENT_NAME	EMPLOYEE_ID	LAST_NAME
1	1	Accounting	4	Hernandez
2	1	Accounting	3	Simonian
3	1	Accounting	9	Locario
4	1	Accounting	8	O'Leary
5	1	Accounting	7	Hardy
6	1	Accounting	6	Watson
7	1	Accounting	5	Aaronsen

```
(45 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

How to code a cross join with the implicit syntax

The implicit syntax for a cross join

```
SELECT select_list
FROM table_1, table_2
```

A cross join that uses the implicit syntax

```
SELECT departments.department_number, department_name,
    employee_id, last_name
FROM departments, employees
ORDER BY departments.department_number
```

The result set

	DEPARTMENT_NUMBER	DEPARTMENT_NAME	EMPLOYEE_ID	LAST_NAME
1	1	Accounting	4	Hernandez
2	1	Accounting	3	Simonian
3	1	Accounting	9	Locario
4	1	Accounting	8	O'Leary
5	1	Accounting	7	Hardy
6	1	Accounting	6	Watson
7	1	Accounting	5	Aaronsen

(45 rows selected)

© 2008, Mike Murach & Associates, Inc.

Terms to know

- Outer join
- Left outer join
- Right outer join
- Equi-join
- Natural join
- Cross join

© 2008, Mike Murach & Associates, Inc.

The syntax for a union

```
    SELECT_statement_1
UNION [ALL]
    SELECT_statement_2
[UNION [ALL]
    SELECT_statement_3]...
[ORDER BY order_by_list]
```

Rules for a union

- The number of columns must be the same in all SELECTs.
- The column data types must be compatible.
- The column names are taken from the first SELECT statement.

© 2008, Mike Murach & Associates, Inc.

A union with data from two different tables

```
    SELECT 'Active' AS source, invoice_number, invoice_date,
        invoice_total
    FROM active_invoices
    WHERE invoice_date >= '01-JUN-2008'
UNION
    SELECT 'Paid' AS source, invoice_number, invoice_date,
        invoice_total
    FROM paid_invoices
    WHERE invoice_date >= '01-JUN-2008'
ORDER BY invoice_total DESC
```

The result set

	SOURCE	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	Active	40318	18-JUL-08	21842
2	Paid	P02-3772	03-JUN-08	7125.34
3	Paid	10843	04-JUN-08	4901.26
4	Paid	77290	04-JUN-08	1750
5	Paid	RTR-72-3662-X	04-JUN-08	1600
6	Paid	75C-90227	06-JUN-08	1367.5
7	Paid	P02-88D77S7	06-JUN-08	856.92
8	Active	I77271-O01	05-JUN-08	662
9	Active	9982771	03-JUN-08	503.2

(22 rows selected)

© 2008, Mike Murach & Associates, Inc.

A union with data from just the Invoices table

```
    SELECT 'Active' AS source, invoice_number, invoice_date,
        invoice_total
    FROM invoices
    WHERE (invoice_total – payment_total – credit_total) > 0
UNION
    SELECT 'Paid' AS source, invoice_number, invoice_date,
        invoice_total
    FROM invoices
    WHERE (invoice_total – payment_total – credit_total)
          <= 0
ORDER BY invoice_total DESC
```

The result set

	SOURCE	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	Paid	0-2058	08-MAY-08	37966.19
2	Paid	P-0259	16-APR-08	26881.4
3	Paid	0-2060	08-MAY-08	23517.58
4	Active	40318	18-JUL-08	21842
5	Active	P-0608	11-APR-08	20551.18
6	Active	0-2436	07-MAY-08	10976.06

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

A union with payment data from the same tables

```
    SELECT invoice_number, vendor_name,
        '33% Payment' AS payment_type,
        invoice_total AS total,
        (invoice_total * 0.333) AS payment
    FROM invoices JOIN vendors
        ON invoices.vendor_id = vendors.vendor_id
    WHERE invoice_total > 10000
UNION
    SELECT invoice_number, vendor_name,
        '50% Payment' AS payment_type,
        invoice_total AS total,
        (invoice_total * 0.5) AS payment
    FROM invoices JOIN vendors
        ON invoices.vendor_id = vendors.vendor_id
    WHERE invoice_total BETWEEN 500 AND 10000
```


© 2008, Mike Murach & Associates, Inc.

The union (continued)

```
UNION
    SELECT invoice_number, vendor_name,
        'Full amount' AS payment_type,
        invoice_total AS Total, invoice_total AS Payment
    FROM invoices JOIN vendors
        ON invoices.vendor_id = vendors.vendor_id
    WHERE invoice_total < 500
ORDER BY payment_type, vendor_name, invoice_number
```

The result set

	INVOICE_NUMBER	VENDOR_NAME	PAYMENT_TYPE	TOTAL	PAYMENT
1	40318	Data Reproductions Corp	33% Payment	21842	7273.386
2	0-2058	Malloy Lithographing Inc	33% Payment	37966.19	12642.74127
3	0-2060	Malloy Lithographing Inc	33% Payment	23517.58	7831.35414
4	0-2436	Malloy Lithographing Inc	33% Payment	10976.06	3655.02798
5	P-0259	Malloy Lithographing Inc	33% Payment	26881.4	8951.5062
6	P-0608	Malloy Lithographing Inc	33% Payment	20551.18	6843.54294
7	509786	Bertelsmann Industry Svcs. Inc	50% Payment	6940.25	3470.125

(114 rows selected)

© 2008, Mike Murach & Associates, Inc.

The syntax for MINUS and INTERSECT operations

```
    SELECT_statement_1
{MINUS|INTERSECT}
    SELECT_statement_2
[ORDER BY order_by_list]
```


© 2008, Mike Murach & Associates, Inc.

The Customers table

CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME
Maria	Anders
Ana	Trujillo
Antonio	Moreno
Thomas	Hardy
Christina	Berglund
Hanna	Moos

(24 rows selected)

The Employees table

FIRST_NAME	LAST_NAME
Cindy	Smith
Elmer	Jones
Ralph	Simonian
Olivia	Hernandez
Robert	Aaronsen
Denise	Watson

(9 rows selected)

© 2008, Mike Murach & Associates, Inc.

A query that excludes rows from the first query if they also occur in the second query

```
    SELECT customer_first_name, customer_last_name
    FROM customers
MINUS
    SELECT first_name, last_name
    FROM employees
ORDER BY customer_last_name
```

The result set

	CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME
1	Maria	Anders
2	Christina	Berglund
3	Art	Braunschweiger
4	Donna	Chelan

```
(23 rows selected)
```


© 2008, Mike Murach & Associates, Inc.

A query that only includes rows that occur in both queries

```
    SELECT customer_first_name, customer_last_name
    FROM customers
INTERSECT
    SELECT first_name, last_name
    FROM employees
```

The result set

	CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME
1	Thomas	Hardy

```
(1 rows selected)
```


© 2008, Mike Murach & Associates, Inc.