Murach Chapter 4 Part 4

How to Retrieve Data From Two or More Tables

Week 4, Lecture 7

Knowledge Points in this lecture

- OUTER JOIN of 3 tables
- Mix of OUTER JOIN and INNER JOIN
- Explicit/Implicit syntax for JOINs
  - CROSS JOIN, INNER JOIN, OUTER JOIN
- JOIN USING
- NATURAL JOIN
- Set Operators in SQL
  - UNION, INTERSECT, MINUS
- In-Class Practice – Outer Join

Query that uses 2 left outer joins of 3 tables

```
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

Query that uses 2 right outer joins of 3 tables

```
SELECT department_name, last_name, project_number AS proj_no
FROM departments d
    RIGHT JOIN employees e
        ON d.department_number = e.department_number
    RIGHT 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	Payroll	Aaronsen	P1012
3	Payroll	Simonian	P1012
4	Payroll	Smith	P1012
5	Personnel	O'Leary	P1011
6	(null)	Locario	P1013
7	(null)	Watson	P1013
8	(null)	(null)	P1014

Query that uses 2 full outer joins of 3 tables

```
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	Aaronsen	P1012
6	Payroll	Smith	P1012
7	Personnel	Jones	(null)
8	Personnel	O'Leary	P1011
9	(null)	Locario	P1013
10	(null)	(null)	P1014
11	(null)	Watson	P1013

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

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]...
```

- [A]: A is optional
- {A|B}: A or B

What outer joins do

Join	Keeps unmatched rows (Extra information) from
Left	The left table
Right	The right table
Full	Both tables

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
```

Implicit cross join:

- Omit keyword JOIN
- No WHERE condition

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

Explicit CROSS JOIN Equivalent to Previous Slide

- Use keyword: CROSS JOIN
- No WHERE condition

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

Implicit syntax that inner-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)

Implicit inner join:

- No keyword JOIN in FROM clause
- MUST have join condition in WHERE clause

General Implicit syntax for 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 inner-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-14	17.5	Office Supplies
2	Blue Cross	547481328	20-MAY-14	224	Group Insurance
3	Blue Cross	547480102	19-MAY-14	224	Group Insurance
4	Blue Cross	547479217	17-MAY-14	116	Group Insurance
5	Cardinal Business Media, Inc.	134116	01-JUN-14	90.36	Card Deck Advertising

(44 rows selected)

The implicit syntax for an outer join

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

- NO keyword JOIN in FROM clause
- Place (+) next to the table with missing (less) information in join condition
  - E.g. table1.column_name = table2.column_name (+)
  - + means: we need to ADD some information to table 2 for a matching value in table 1
- **CanNOT place (+) on both sides of a comparison operator**
  - **So NO Implicit Full outer join in Oracle.**

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

Explicit LEFT OUTER JOIN Equivalent to Previous Slide

- Use keyword: LEFT JOIN

```
SELECT department_name, AS dept_name,
    dpt.department_number AS dept_no,
    last_name
FROM  departments dpt LEFT JOIN employees emp
    ON dpt.department_number =
 emp.department_number
ORDER BY department_name;
```

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)

Explicit RIGHT OUTER JOIN Equivalent to Previous Slide

- Use keyword: RIGHT JOIN

```
SELECT department_name, AS dept_name,
    dpt.department_number AS dept_no,
    last_name
FROM  departments dpt RIGHT JOIN employees emp
    ON dpt.department_number =
 emp.department_number
ORDER BY department_name;
```

A SELECT statement with JOIN and the USING keyword keyword

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

Join tables based on matching values in specified common (i.e. same name) column(s)

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)
```

JOIN ON Equivalent to Previous Slide

```
SELECT invoice_number, vendor_name
FROM vendors v
     JOIN invoices i
         ON v.vendor_id = i.vendor_id
ORDER BY invoice_number;
```

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]: A is optional
- {A|B}: A or B

More Complex SELECT with JOIN and USING

```
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
```

Note: for self-reading, not required in the course

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)

NATRUAL JOIN

- Join two tables based on matching values in ALL columns in the two tables that have the same names.
  - Don’t need to explicitly specify what columns to use in natural join
- Only works correctly if the database is designed in a certain way.
  - Natural join may not work if database design changes
- May have unexpected results for complex queries
- More common to use JOIN ON or JOIN USING

A SELECT statement with JOIN and 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)

Natural JOIN - Join two tables based on matching values in ALL columns in the two tables that have the same names

JOIN ON Equivalent to Previous Slide

- Only 1 common column vendor_id

```
SELECT invoice_number, vendor_name
FROM vendors v
     JOIN invoices i
         ON v.vendor_id = i.vendor_id
ORDER BY invoice_number;
```

Set Operators in SQL

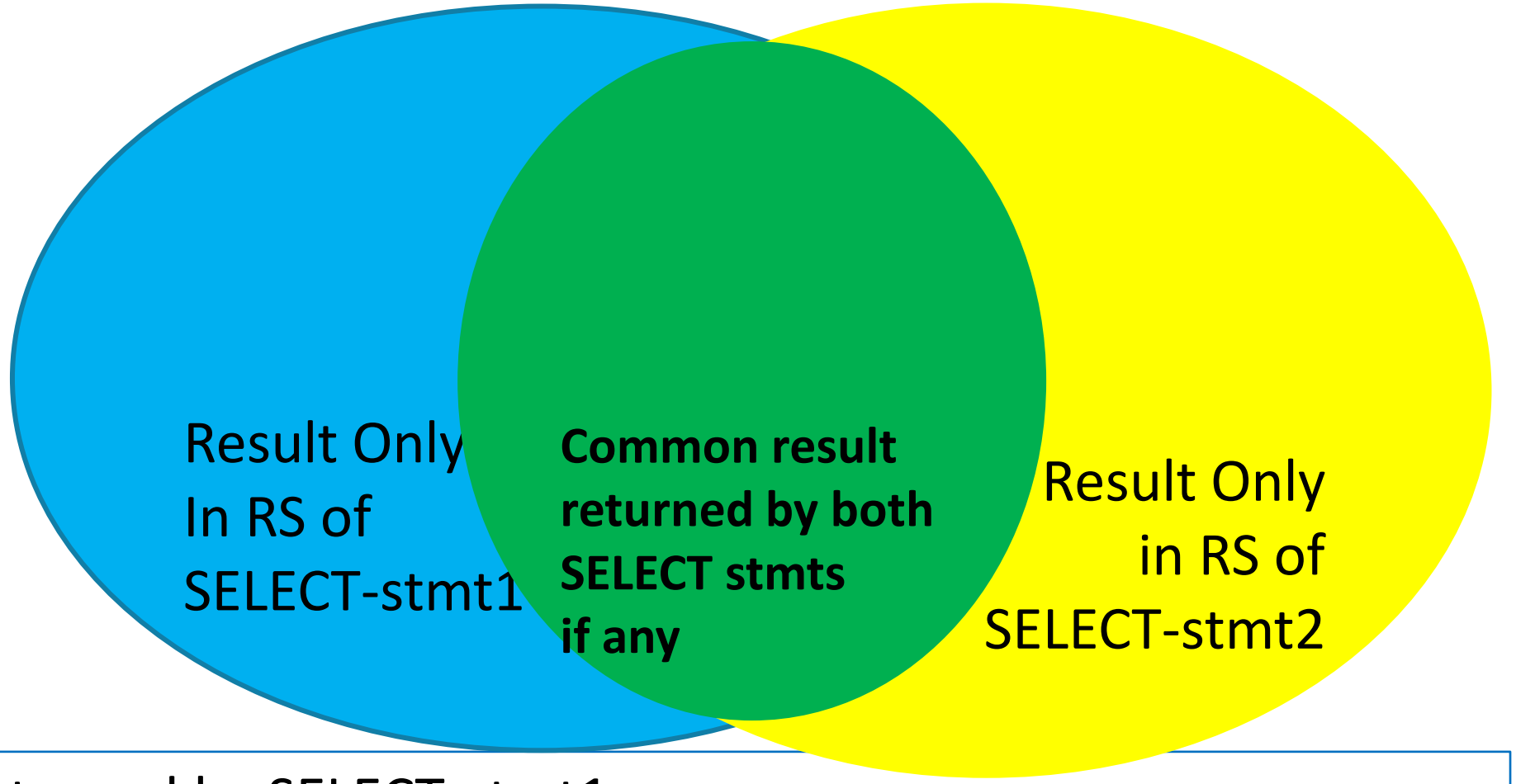
- Used to combine query results from multiple SELECT statements
- Each SELECT statement returns a set of rows: ResultSet.
- Set operators in SQL
  - UNION, INTERSECT, MINUS
- UNION operator
  - Same as set operator ∪ in Discrete Math
- INTERSECT operator
  - Same as set operator ∩ in Discrete Math
- MINUS operator
  - Same as set operator − in Discrete Math

Review: Set Operators in Discrete Math

- Sets A = {1, 2, 3}, B = {3, 4, 5}

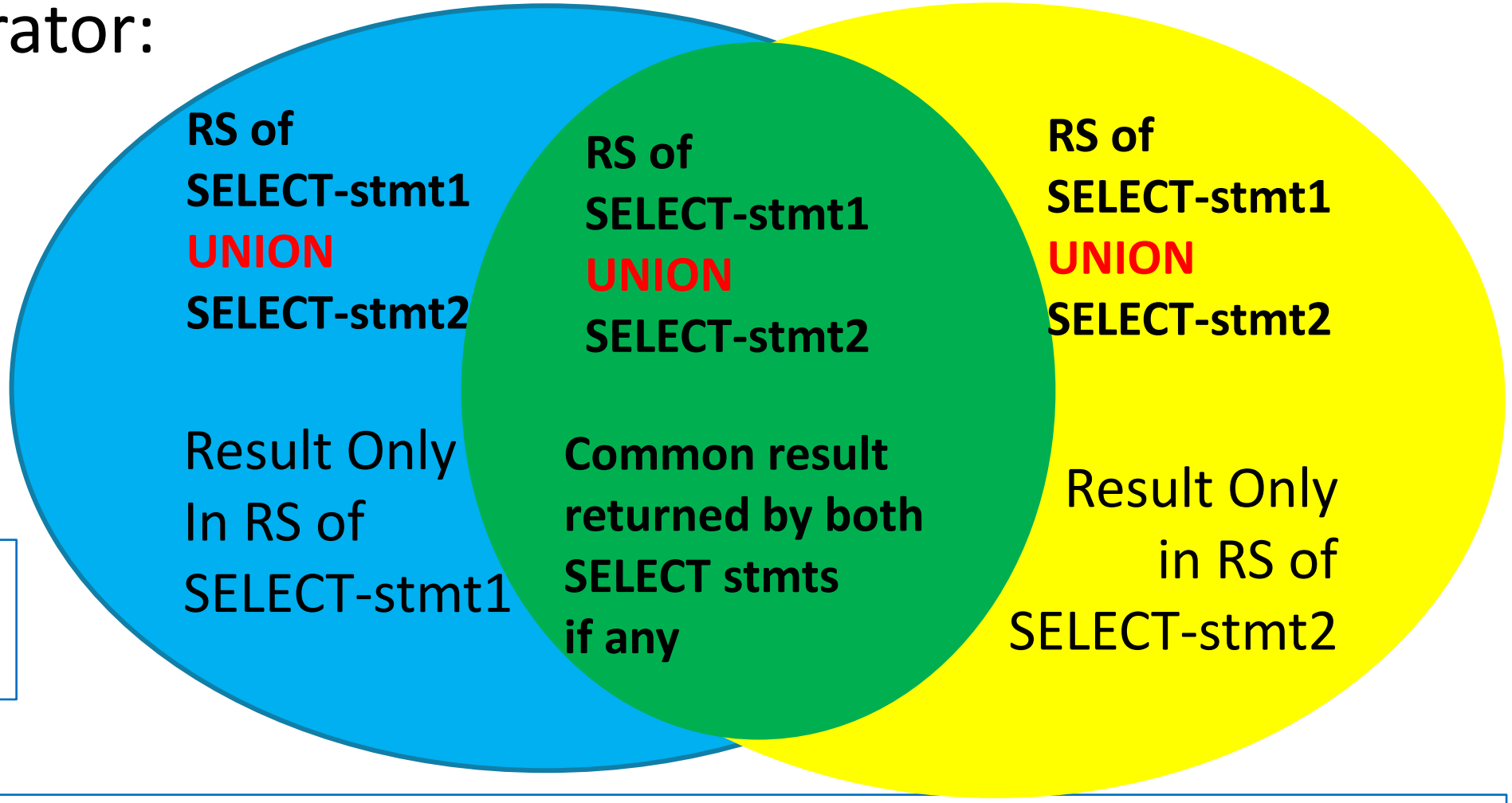
- A ∪ B = {1, 2, 3, 4, 5}
- $A \cap B$ = {3}
- $A - B$ = {1, 2}
- $B - A$ = {4, 5}

RS: result set
stmt: statement



Blue: result only returned by SELECT-stmt1
Yellow: result only returned by SELECT-stmt2
Green: result returned by both SELECT-stmt1 and SELECT-stmt2 if there is any.
Result Set of SELECT-stmt1 = Blue-Area + Green-Area
Result Set of SELECT-stmt2 = Yellow-Area + Green-Area

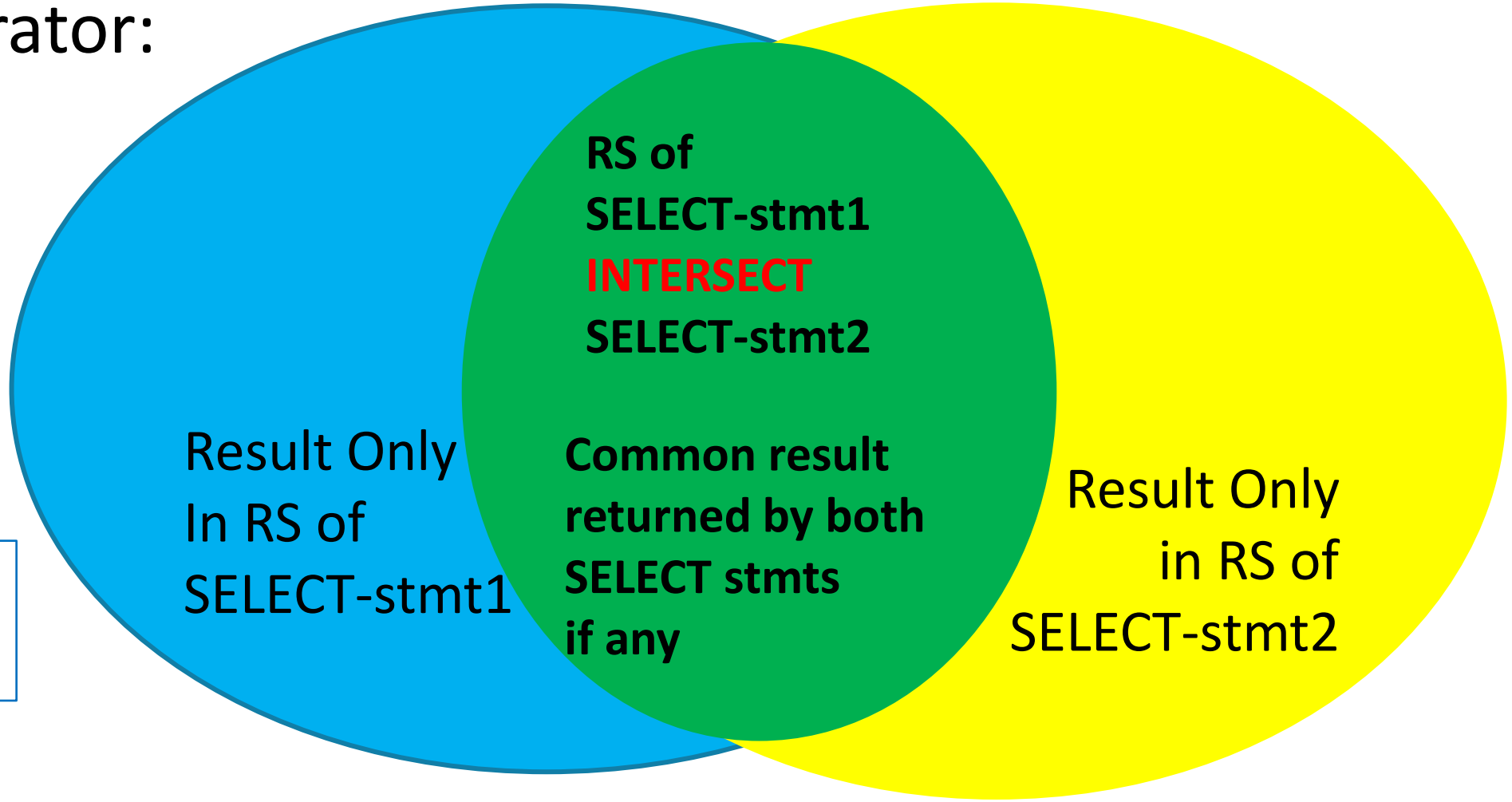
SQL set operator: UNION



RS: result set
stmt: statement

Result Set of (SELECT-stmt1 UNION SELECT-stmt2) = Blue + Green + Yellow
= ALL in Result Set of SELECT-stmt1 (Blue + Green)
+ ALL in Result Set of SELECT-stmt2 (Yellow + Green) **– DuplicatesInGreen**

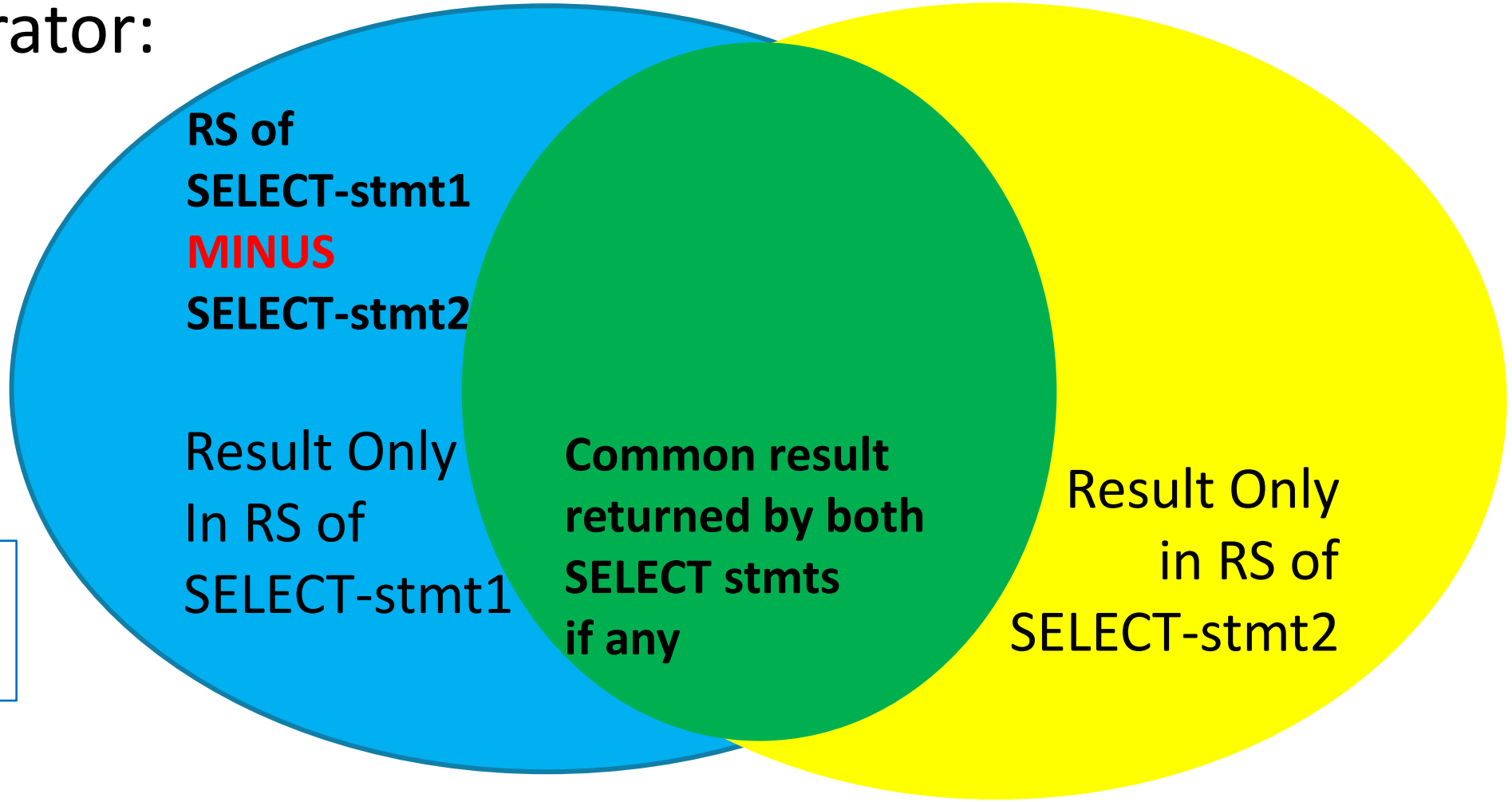
SQL set operator: INTERSECT



RS: result set
stmt: statement

Result Set of (SELECT-stmt1 INTERSECT SELECT-stmt2) = Green
= Common result in BOTH Result Set of SELECT-stmt1 (Blue + Green)
AND Result Set of SELECT-stmt2 (Yellow + Green)

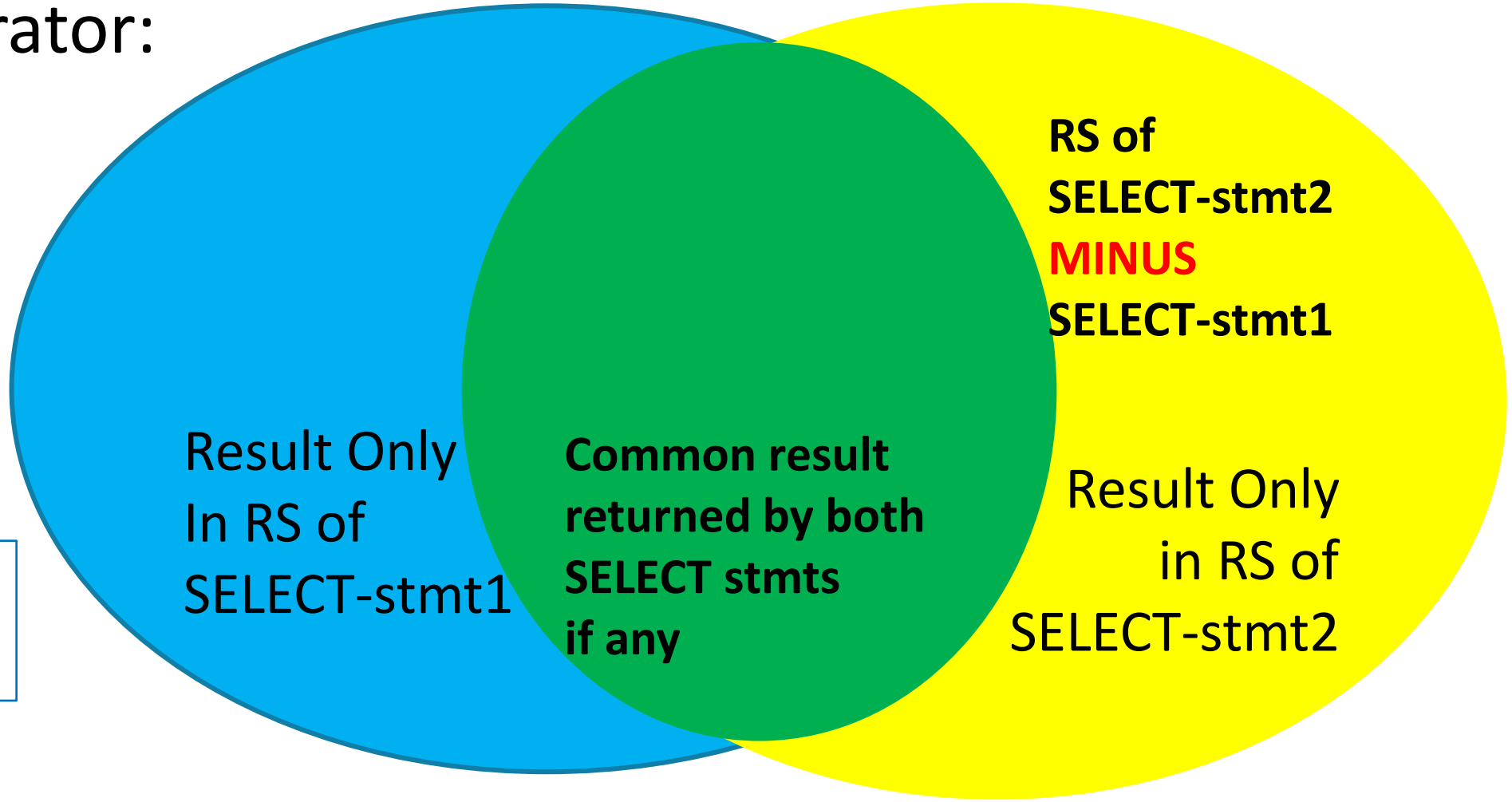
SQL set operator: MINUS



RS: result set
stmt: statement

Result Set of (SELECT-stmt1 MINUS SELECT-stmt2) = Blue
= Result ONLY in Result Set of SELECT-stmt1 (Blue + Green), but
NOT in Result Set of SELECT-stmt2 (Yellow + Green)

SQL set operator: MINUS



RS: result set
stmt: statement

Result Set of (SELECT-stmt2 MINUS SELECT-stmt1) = YELLOW
= Result ONLY in Result Set of SELECT-stmt2 (Yellow + Green), but
NOT in Result Set of SELECT-stmt1 (Blue + Green)

The Customers table

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

(24 rows selected)

The Employees table

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

(9 rows selected)

A query that includes rows that occur in first query or second query

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

Different column names in result sets of two SELECT statements

The result set

	CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME
1	Robert	Aaronsen
2	Maria	Anders
3	Christina	Berglund
4	Art	Braunschweiger
5	Donna	Chelan
6	Fred	Citeaux
7	Thomas	Hardy
8	Olivia	Hernandez

```
(32 rows selected)
```

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
ORDER BY customer_last_name
```

Different column names in result sets of two SELECT statements

The result set

	CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME
1	Thomas	Hardy

```
(1 rows selected)
```

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
```

Different column names in result sets of two SELECT statements

The result set

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

(23 rows selected)

The syntax for a union

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

[A]: A is optional

ALL - Include duplicates in result set of UNION
Default: no duplicate

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 in the result are taken from the first SELECT statement.
- Column names in ORDER BY must be same as those in first SELECT statement.

Different SELECT statements may have different column names in their SELECT clauses.

The syntax for MINUS and INTERSECT operations

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

- [A]: A is optional
- {A|B}: A or B

Same rules as for a union

- The number of columns must be the same in all SELECTs.
- The column data types must be compatible.
- The column names in the result are taken from the first SELECT statement.
- Column names in ORDER BY must be same as those in first SELECT statement. (Can also use column position in ORDER BY)

Different SELECT statements may have different column names in their SELECT clauses.