MULTI-TABLE QUERIES

CARTESIAN PRODUCT

- Cartesian product takes all pairwise combinations of rows of two tables

A

a_1	...	a_m

B

b_1	...	b_n

$$A \times B = \{(a_1, \cdots, a_m, b_1, \cdots, b_m) | (a_1, \cdots, a_m) \in A, (b_1, \cdots, b_m) \in B\}$$

$A \times B$

a_1	...	a_m	b_1	...	b_n

CARTESIAN PRODUCT

- **Example:** Consider the following tables holding information of customers and offices

CUSTOMERS

CID	CITY
C1	İstanbul
C2	Ankara
C3	İzmir

OFFICES

OID	CITY
O1	İstanbul
O2	Adana

CUSTOMERS xOFFICES

CID	CITY	OID	CITY
C1	İstanbul	O1	İstanbul
C1	İstanbul	O2	Adana
C2	Ankara	O1	İstanbul
C2	Ankara	O2	Adana
C3	İzmir	O1	İstanbul
C3	İzmir	O2	Adana

CARTESIAN PRODUCT

- Unlike set theory, order of pairs is not important since order of columns is not employed in relational model

$$(a_1, \cdots, a_m, b_1, \cdots, b_m) = (b_1, \cdots, b_m, a_1, \cdots, a_m)$$

- Hence Cartesian product in relational model is commutative

$$A \times B = B \times A$$

CARTESIAN PRODUCT

- Cartesian product in SQL
- Simply list names of the table after from

```
SELECT *
FROM Customers, Offices
```

- This is same as

```
SELECT *
FROM Offices, Customers
```

CARTESIAN PRODUCT

- Cartesian product is associative

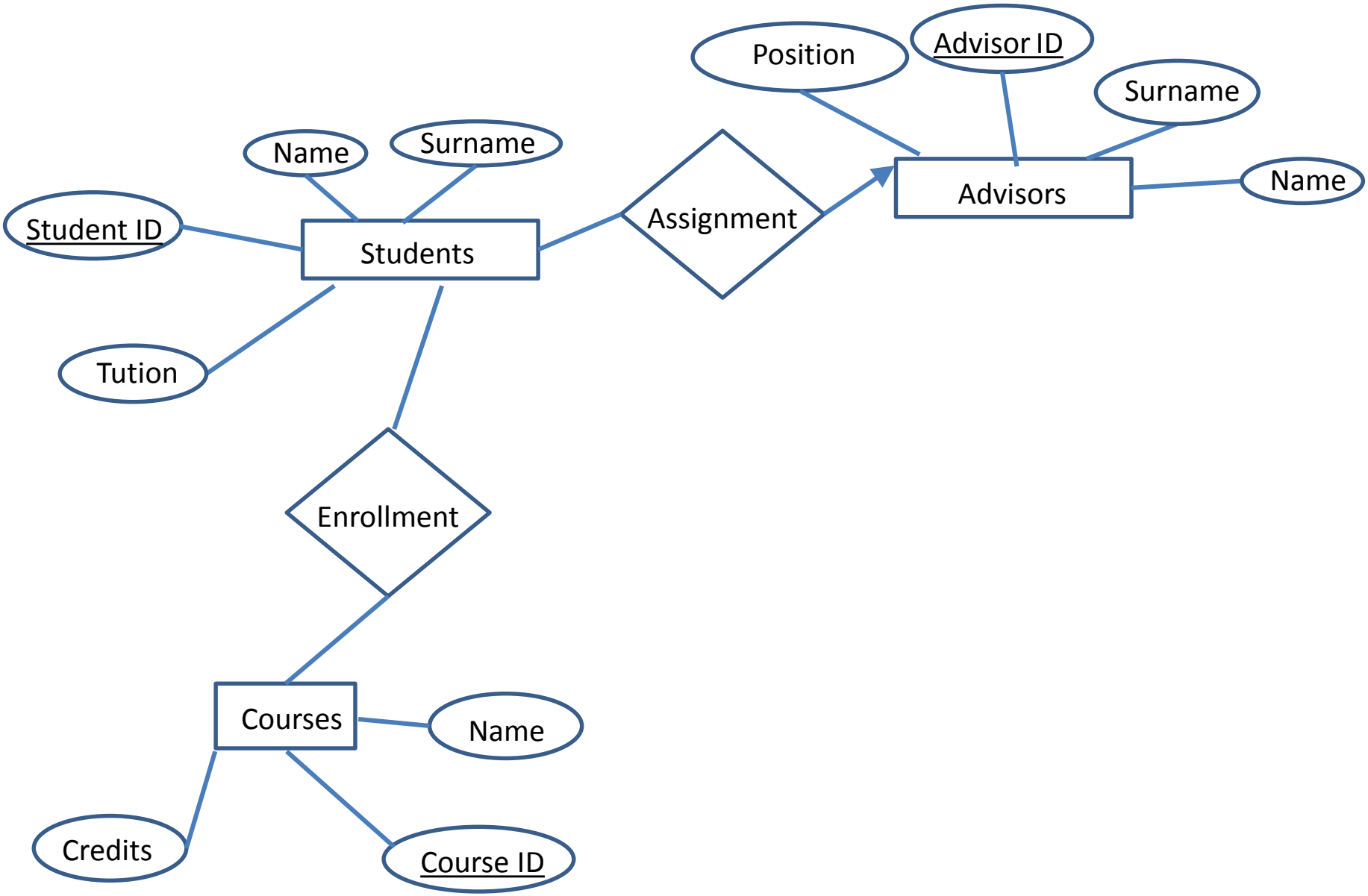
$$(A \times B) \times C = A \times (B \times C) = A \times B \times C$$

- Cartesian product of more than two tables is all possible combinations of rows of tables

JOIN

- The operations we studies until now (except Cartesian product) apply to a single table
- But frequently we need to make use of relation between different entity sets to obtain the information we want
- Therefore we need combine information from multiple tables

JOIN


Position
Advisor ID
Surname
Name
Surname
Advisors
Name
Student ID
Students
Assignment
Tution
Enrollment
Courses
Name
Credits
Course ID

JOIN

ADVISORS

AdvisorID	Name	Surname	Position
A1	Halis	Sak	Assit. Prof.
A3	Uğur	Yıldıran	Assit. Prof.
A2	Samet	Yılmaz	TA
A4	Murat	Tunç	Prof.

COURSES

CourseID	CName	Credits
SYE346	Management Information Systems	3
SYE222	Operations Research	4
SYE365	Termodynamics	3
SYE216	Cost Engineering	3

STUDENTS

Student ID	Name	Surname	Tution	AID
20570322	Ali	Ergin	0	A1
20770367	Mehmet	Kavak	12000	A4
20850286	Selin	Pekcan	24000	A4
20978909	Ercan	Sivri	12000	A3
20670346	Ahmet	İnce	0	A2
20670312	Aysu	Doğan	24000	A3
20870333	Hatice	Dündar	24000	A1

ENROLLMENT

SID	CID
20570322	SYE346
20770367	SYE346
20570322	SYE365
20978909	SYE222
20850286	SYE365
20670346	SYE216
20870333	SYE222

- List the names of students whose advisor is Uğur Yıldıran
- List the names of advisors whose students does not pay tution
- Which students are taking MIS course?

JOIN

- In general we want computer give this information automatically
- We will not look at the tables and find them
- It is very difficult task for large databases
- We will use JOIN operation for this purpose

JOIN

- Join operation can be used to combine information from two tables.
- For this operation, first we need to chose a column (or a combination of columns) from each table.
- Join simply takes pairwise combination of rows from two tables.
- But unlike Cartesian product it does not take all pairs.
- It only takes pairs for which values in the chosen columns are the same.

JOIN

A

a_1	...	a_m

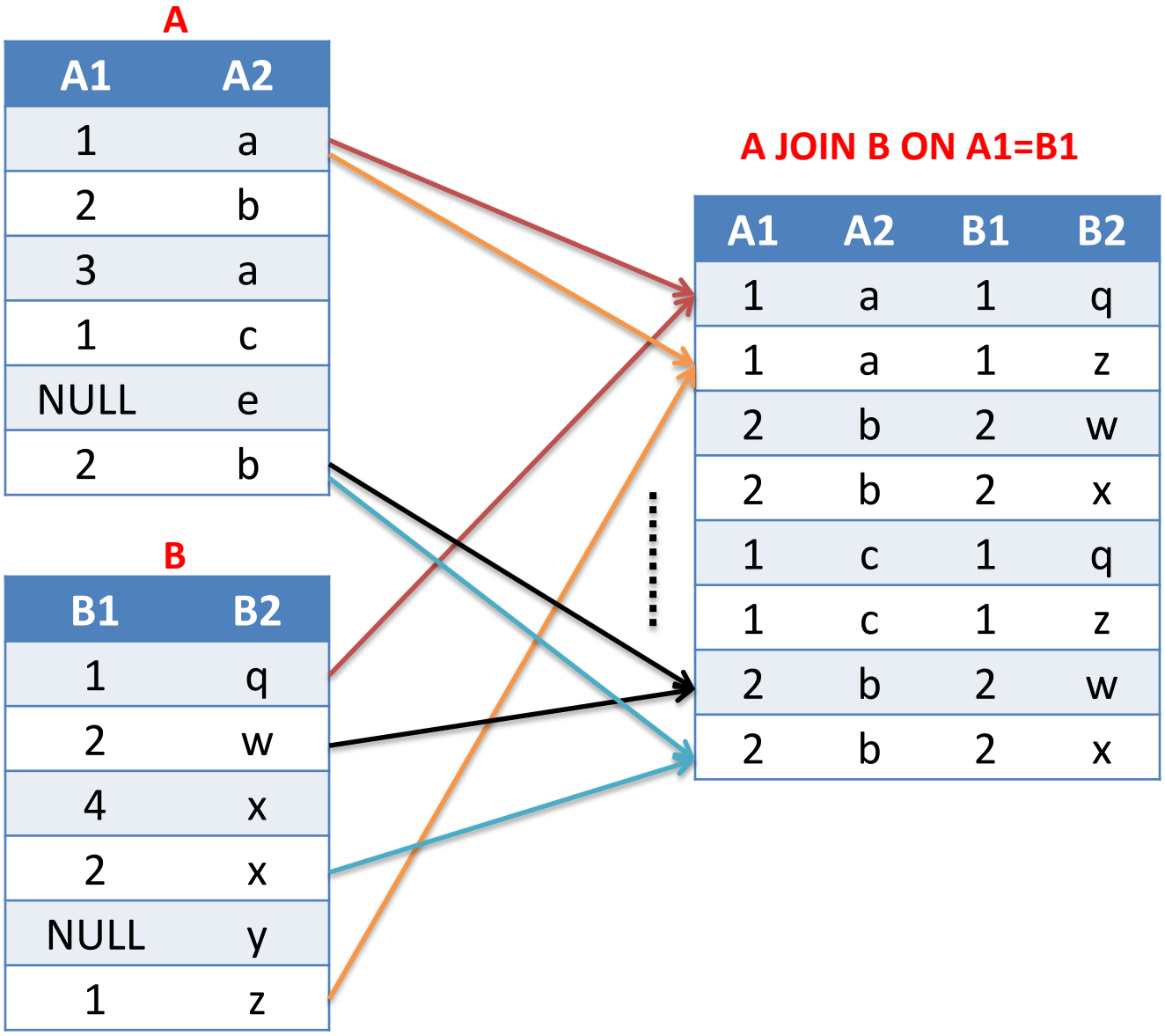
B

b_1	...	b_n

Suppose we choose columns a_i and b_j from tables A and B

$$A \text{ JOIN } B \text{ ON } (a_i = b_j) = \{(a_1, \dots, a_m, b_1, \dots, b_m) \in A \times B | a_i = b_j\}$$

JOIN


A
A1 A2
1 a
2 b
3 a
1 c
NULL e
2 b
B
B1 B2
1 q
2 w
4 x
2 x
NULL y
1 z
A JOIN B ON A1=B1
A1 A2 B1 B2
1 a 1 q
1 a 1 z
2 b 2 w
2 b 2 x
1 c 1 q
1 c 1 z
2 b 2 w
2 b 2 x

JOIN

- JOIN can be obtained by combining two operations we saw previously

$$A \text{ JOIN } B \text{ ON } (a_i = b_j) = \{(a_1, \dots, a_m, b_1, \dots, b_m) \in A \times B | a_i = b_j\}$$

JOIN

- JOIN can be obtained by combining two operations we saw previously
  - First take Cartesian product of two tables
  - Then select the rows for which the matched columns have the same value

$$A \text{ JOIN } B \text{ ON } (a_i = b_j) = \{(a_1, \ldots, a_m, b_1, \ldots, b_m) \in A \times B | a_i = b_j\}$$

JOIN

A

A1	A2
1	a
2	b
3	a
1	c
NULL	e
2	b

B

B1	B2
1	q
2	w
4	x
2	x
NULL	y
1	z

A X B

A1	A2	B1	B2
1	a	1	q
1	a	2	w
1	a	4	x
1	a	2	x
1	a	NULL	y
1	a	1	z
2	b	1	q
2	b	2	w
2	b	4	x
2	b	2	x
2	b	NULL	y
2	b	1	z
3	a	1	q
3	a	2	w
3	a	4	x
3	a	2	x
3	a	NULL	y
3	a	1	z
1	c	1	q
1	c	2	w
1	c	4	x
1	c	2	x
1	c	NULL	y
1	c	1	z
NULL	e	1	q
NULL	e	2	w
NULL	e	4	x
NULL	e	2	x
NULL	e	NULL	y
NULL	e	1	z
2	b	1	q
2	b	2	w
2	b	4	x
2	b	2	x
2	b	NULL	y
2	b	1	z

JOIN

- Based on this fact, we can write an SQL query performing join by combining Cartesian product and row selection operations
- **Example:** Join of STUDENTS and ADVISORS tables on AID and AdvisorID

JOIN

STUDENTS

Student ID	Name	Surname	Tution	AID
20570322	Ali	Ergin	0	A1
20770367	Mehmet	Kavak	12000	A4
20850286	Selin	Pekcan	24000	A4
20978909	Ercan	Sivri	12000	A3
20670346	Ahmet	İnce	0	A2
20670312	Aysu	Doğan	24000	A3
20870333	Hatice	Dündar	24000	A1

ADVISORS

AdvisorID	Name	Surname	Position
A1	Halis	Sak	Assit. Prof.
A3	Uğur	Yıldıran	Assit. Prof.
A2	Samet	Yılmaz	TA
A4	Murat	Tunç	Prof.

```
SELECT *
FROM STUDENTS, ADVISORS
WHERE  AdvisorID=AID
```

Student ID	Name	Surname	Tution	AID	AdvisorID	Name	Surname	Position
20570322	Ali	Ergin	0	A1	A1	Halis	Sak	Assit. Prof.
20770367	Mehmet	Kavak	12000	A4	A4	Murat	Tunç	Prof.
20850286	Selin	Pekcan	24000	A4	A4	Murat	Tunç	Prof.
20978909	Ercan	Sivri	12000	A3	A3	Uğur	Yıldıran	Assit. Prof.
20670346	Ahmet	İnce	0	A2	A2	Samet	Yılmaz	TA
20670312	Aysu	Doğan	24000	A3	A3	Uğur	Yıldıran	Assit. Prof.
20870333	Hatice	Dündar	24000	A1	A1	Halis	Sak	Assit. Prof.

JOIN

Student ID	Name	Surname	Tution	AID	AdvisorID	Name	Surname	Position
20570322	Ali	Ergin	0	A1	A1	Halis	Sak	Assit. Prof.
20770367	Mehmet	Kavak	12000	A4	A4	Murat	Tunç	Prof.
20850286	Selin	Pekcan	24000	A4	A4	Murat	Tunç	Prof.
20978909	Ercan	Sivri	12000	A3	A3	Uğur	Yıldıran	Assit. Prof.
20670346	Ahmet	İnce	0	A2	A2	Samet	Yılmaz	TA
20670312	Aysu	Doğan	24000	A3	A3	Uğur	Yıldıran	Assit. Prof.
20870333	Hatice	Dündar	24000	A1	A1	Halis	Sak	Assit. Prof.

- We may obtain the following information by using the result of join
  - List the names of advisors whose students does not pay tuition
  - List the names of students whose advisor is Uğur Yıldıran
  - Etc.

JOIN

- List the names of advisors whose students does not pay tuition

```
SELECT ADVISOR.Name, ADVISOR.Surame
FROM STUDENTS, ADVISORS
WHERE   AdvisorID=AID AND Tution=0
```

Name	Surname
Halis	Sak
Samet	Yılmaz

Here we use names of tables to qualify the columns having the same names

JOIN

- List the names of students whose advisor is Uğur Yıldıran

```
SELECT STUDENTS.Name, STUDENTS.Surname
FROM STUDENTS, ADVISORS
WHERE   AdvisorID=AID AND
        ADVISOR.Name='Uğur' AND
        ADVISOR.Surame='Yıldıran'
```

Name	Surname
Ercan	Sivri
Aysu	Doğan

JOIN

- Join is a commutative operation just like Cartesian product
- Hence, the following queries are the same

```
SELECT *
FROM STUDENTS, ADVISORS
WHERE  AdvisorID=AID
```

```
SELECT *
FROM ADVISORS, STUDENTS
WHERE  AdvisorID=AID
```

JOIN

- In 1992 a new syntax is introduced for JOIN
  - It does not combine Cartesian product and row selection
  - It defines JOIN as a new operation using JOIN keyword
- **Example:** Join of Advisors and Students tables in SQL92 syntax

```
SELECT *
FROM STUDENTS JOIN ADVISORS ON AdvisorID=AID
```

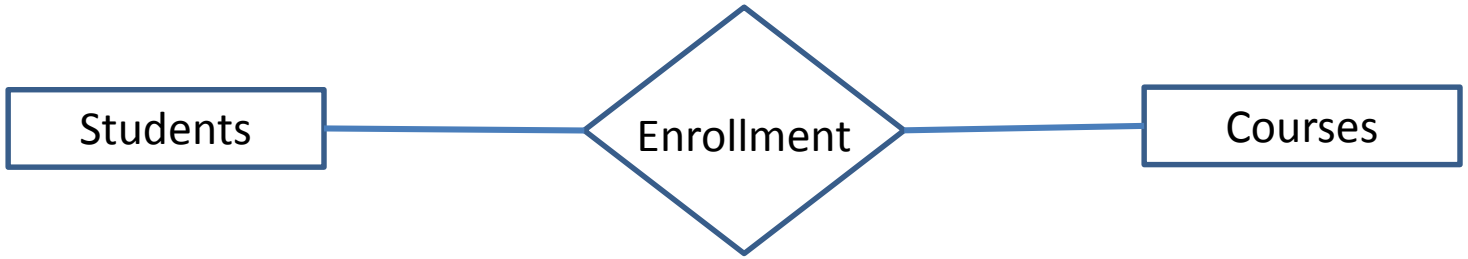
JOIN

- Row selection is performed separately using WHERE clause
- **Example:** List the names of advisors whose students does not pay tuition

```
SELECT ADVISOR.Name, ADVISOR.Surname
FROM STUDENTS JOIN ADVISORS ON AdvisorID=AID
WHERE Tution=0
```

JOIN

- We may need to apply join operation to more than two tables
- **Example:** Which students are taking the course named ‘Management Information Systems’?



JOIN

ADVISORS

AdvisorID	Name	Surname	Position
A1	Halis	Sak	Assit. Prof.
A3	Uğur	Yıldıran	Assit. Prof.
A2	Samet	Yılmaz	TA
A4	Murat	Tunç	Prof.

COURSES

CourseID	CName	Credits
SYE346	Management Information Systems	3
SYE222	Operations Research	4
SYE365	Termodynamics	3
SYE216	Cost Engineering	3

STUDENTS

Student ID	Name	Surname	Tution	AID
20570322	Ali	Ergin	0	A1
20770367	Mehmet	Kavak	12000	A4
20850286	Selin	Pekcan	24000	A4
20978909	Ercan	Sivri	12000	A3
20670346	Ahmet	İnce	0	A2
20670312	Aysu	Doğan	24000	A3
20870333	Hatice	Dündar	24000	A1

ENROLLMENT

SID	CID
20570322	SYE346
20770367	SYE346
20570322	SYE365
20978909	SYE222
20850286	SYE365
20670346	SYE216
20870333	SYE222

JOIN

STUDENTS JOIN ENROLLMENT ON StudentID=SID

Name	Surname	Tution	AID	StudentID	SID	CID
Ali	Ergin	0	A1	20570322	20570322	SYE346
Mehmet	Kavak	12000	A4	20770367	20770367	SYE346
Ali	Ergin	0	A1	20570322	20570322	SYE365
Ercan	Sivri	12000	A3	20978909	20978909	SYE222
Selin	Pekcan	24000	A4	20850286	20850286	SYE365
Ahmet	İnce	0	A2	20670346	20670346	SYE216
Hatice	Dündar	24000	A1	20870333	20870333	SYE222

COURSES

CourseID	CName	Credits
SYE346	Management Information Systems	3
SYE222	Operations Research	4
SYE365	Termodynamics	3
SYE216	Cost Engineering	3

JOIN

(STUDENTS JOIN ENROLLMENT ON StudentID=SID) JOIN COURSES ON CourseID=CID

Name	Surname	Tution	AID	StudentID	SID	CID	CourseID	CName	Credits
Ali	Ergin	0	A1	20570322	20570322	SYE346	SYE346	Management Information Systems	3
Mehmet	Kavak	12000	A4	20770367	20770367	SYE346	SYE346	Management Information Systems	3
Ali	Ergin	0	A1	20570322	20570322	SYE365	SYE365	Termodynamics	3
Ercan	Sivri	12000	A3	20978909	20978909	SYE222	SYE222	Operations Research	4
Selin	Pekcan	24000	A4	20850286	20850286	SYE365	SYE365	Termodynamics	3
Ahmet	İnce	0	A2	20670346	20670346	SYE216	SYE216	Cost Engineering	3
Hatice	Dündar	24000	A1	20870333	20870333	SYE222	SYE222	Operations Research	4

JOIN

- Join operation is associative
- Hence the following equivalent
  - (STUDENTS JOIN ENROLLMENT ON StudentID=SID) JOIN COURSES ON CourseID=CID
  - (ENROLLMENT JOIN COURSES ON CourseID=CID) JOIN STUDENTS ON StudentID=SID
- It simply takes triple combination of rows from three tables for which the chosen columns has the same value

JOIN

- What are the names and surnames of students taking the course named ‘Management Information Systems’?

JOIN

- What are the names and surnames of students taking the course named ‘Management Information Systems’?
- In classical syntax:

```
SELECT Name, Surname
FROM STUDENTS, ENROLLMENT, COURSES
WHERE StudentID=SID AND CourseID=CID AND
        CName= ‘Management Information Systems’
```

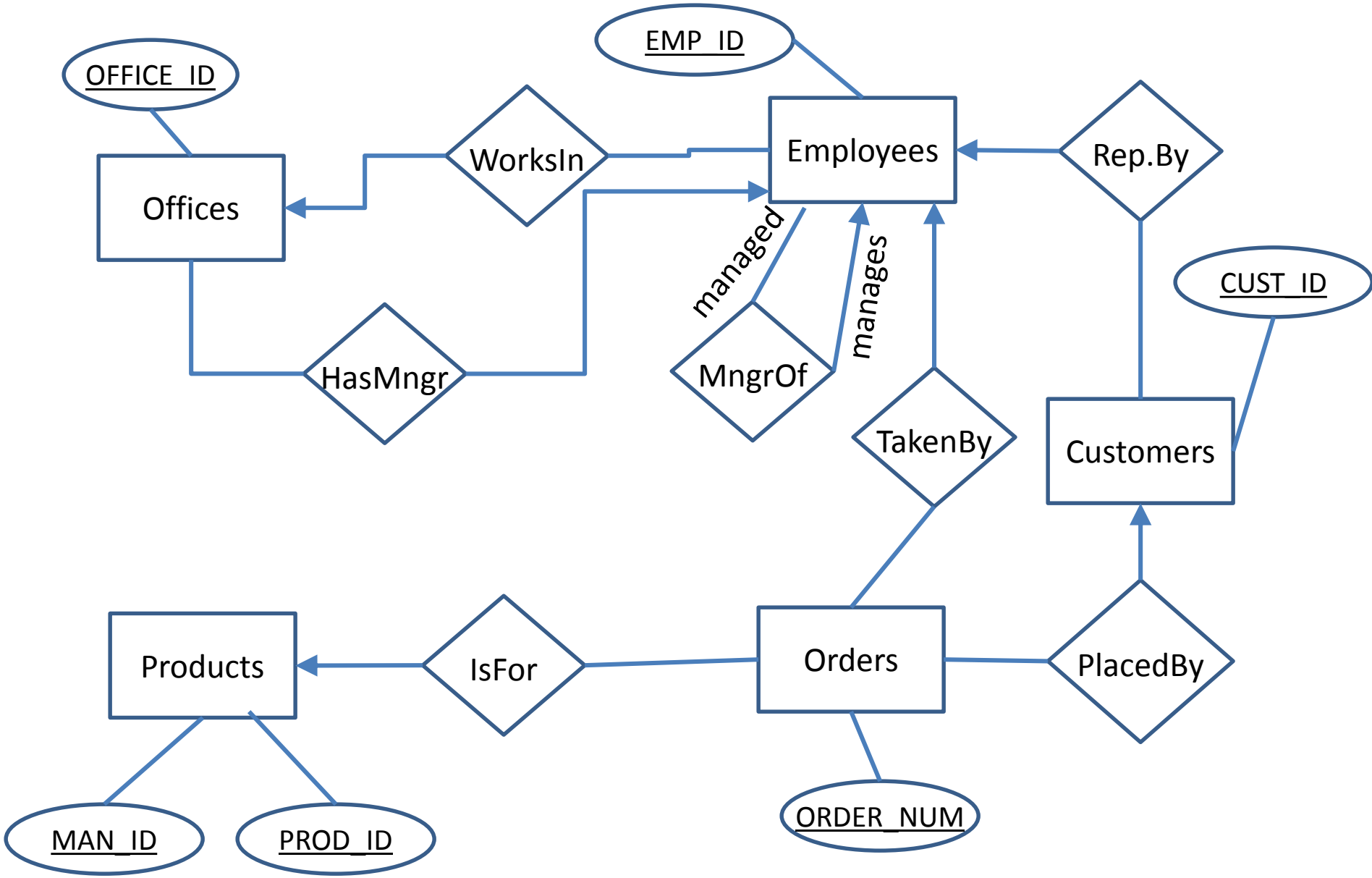
JOIN

- What are the names and surnames of students taking the course named ‘Management Information Systems’?
- In SQL92 syntax:

```
SELECT Name, Surname
FROM STUDENTS JOIN ENROLLMENT ON StudentID=SID
        JOIN COURSES ON CourseID=CID
WHERE CName= ‘Management Information Systems’
```

PRACTICE ON THE EXAMPLE DATABASE

EXAMPLE DATABASE



EXAMPLES

- List all orders showing order number, price, company name, and the customer’s credit limit

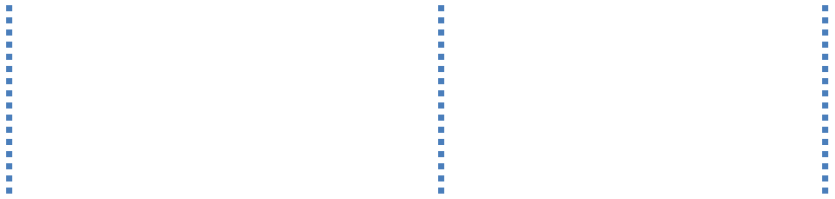
EXAMPLES

- List all orders showing order number, price, company name, and the customer's credit limit

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
            MAX_CREDIT
FROM CUSTOMERS, ORDERS
WHERE CUST_ID=CUST_NUM
```

EXAMPLES

CUST_ID	COMP_NAME	CST_ REP	MAX_ CREDIT	ORDER_ NUM	ORDER_ DATE	CUST_ NUM	REP_ NUM	MAN	PROD	QUANT	ORD_ PRICE
117	Roco Inc.	9	64000	200001	16.11.2008	117	6	CHI	45AG	15	32400
111	Tory Brothers	7	40000	200002	12.02.2009	111	5	HAM	303	25	36985
101	ACP Inc.	3	65000	200003	02.12.2011	101	6	AX	212	9	2675
118	Southeast Sys.	8	60000	200004	11.03.2009	118	8	SCR	ZTO1	3	2530
102	Second Corp.	1	50000	200005	11.09.2006	102	1	HAM	304	36	4975
107	Jack Mfg.	6	80000	200006	28.05.2003	107	10	HAM	307	6	32400



EXAMPLES

- List each employes's name together with the location and region of the office at which she/he works

EXAMPLES

- List each employes's name together with the location and region of the office at which she/he works

```
SELECT FL_NAME, LOCATION, REGION
FROM EMPLOYEES, OFFICES
WHERE OFFICE= OFFICE_ID
```

EXAMPLES

EMP_ ID	NAME	AGE	OFFICE	TITLE	HIRE_ DATE	SPV.	EMP_ TARGET	EMP_ SALES	OFFICE _ID	LOCATION	REGION	MANAGER	OFF_ TARGET	OFF_ SALES
1	Vanessa Abrams	55	42	Purchasing Mgr	2015-12-20	4	450000	567345	42	Seattle	Northern	5	675000	367995
2	Blair Waldorf	28	51	Sales Rep	2014-04-15	8	450000	376900	51	Houston	Southern	8	490000	575986
3	Carter Baizen	49	42	HR Rep	2004-03-04	4	324500	985678	42	Seattle	Northern	5	675000	367995
4	Nathaniel Archibald	35	42	Sales Mgr	2002-07-18	6	400000	525678	42	Seattle	Northern	5	675000	367995
5	Ivy Dickens	35	40	Supply-Chain Rep	2008-03-22	4	450000	789567	40	Detroit	Northern	6	750000	698345
6	Daniel Humphery	32	42	HR Mgr	2011-07-15	NULL	300000	675489	42	Seattle	Northern	5	675000	367995
7	Georgina Sparks	29	52	Purchasing Rep	2016-03-15	8	200000	167329	52	Miami	Southern	8	148500	478812
8	Charles Bass	25	51	Purchasing Rep	2016-12-12	6	150000	287956	51	Houston	Southern	8	490000	575986
9	Serena Woodsen	31	41	Sales Rep	2017-09-23	6	900000	445723	41	Boston	Northern	4	40000	123876

EXAMPLES

- List the location of offices having a target over $600,000 together with the names and titles of their managers

EXAMPLES

- List the location of offices having a target over $600,000 together with the names and titles of their managers

```
SELECT LOCATION, FL_NAME, TITLE
FROM OFFICES, EMPLOYEES
WHERE MANAGER=EMP_ID AND OFF_TARGET > 600000
```

EXAMPLES

- List the location of offices having a target over $600,000 together with the names and titles of their managers

```
SELECT LOCATION, FL_NAME, TITLE
FROM OFFICES, EMPLOYEES
WHERE MANAGER=EMP_ID AND OFF_TARGET > 600000
```

- In SQL 92 Syntax

EXAMPLES

- List the cities of offices having a target over $600,000 together with the names and titles of their managers

```
SELECT LOCATION, FL_NAME, TITLE
FROM OFFICES, EMPLOYEES
WHERE MANAGER = EMP_ID AND OFF_TARGET >
   600000
```

- In SQL 92 Syntax

```
SELECT LOCATION, FL_NAME, TITLE
FROM OFFICES JOIN EMPLOYEES ON MANAGER=EMP_ID
WHERE OFF_TARGET > 600000
```

EXAMPLES

- List names of companies whose sales representative works at office 41 or 42 or 51 and amount of sales he/she made is more than his/her target

EXAMPLES

- List names of companies whose sales representative works at office 41 or 42 or 51 and amount of sales he/she made is more than his/her target

```
SELECT COMP_NAME
FROM CUSTOMERS, EMPLOYEES
WHERE CST_REP = EMP_ID AND
        EMP_SALES > EMP_TARGET
        AND OFFICE IN(41,42,51)
```

EXAMPLES

- List all the orders, showing their order numbers, prices and product types

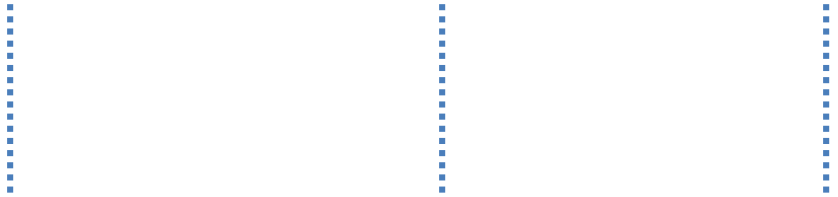
EXAMPLES

- List all the orders, showing their order numbers, prices and product types

```
SELECT ORDER_NUM, ORD_PRICE, TYPE
FROM ORDERS, PRODUCTS
WHERE MAN = MAN_ID AND PROD = PROD_ID;
```

EXAMPLES

ORDER_NUM	ORDER_DATE	CUST NUM	REP NUM	MAN	PROD	QUANT	ORD_ PRICE	MAN_ ID	PROD_ ID	TYPE	PRICE	AV_ QUANT
200001	2008-11-16	117	6	CHI	45AG	15	32400	CHI	45AG	CHISEL PIN	373	37
200002	2009-02-12	111	5	HAM	303	25	36985	HAM	303	SIZE 3 HAMMER	225	25
200003	2011-12-02	101	6	AX	212	9	2675	AX	212	LARGE AX	2580	100
200004	2009-03-11	118	8	SCR	ZTO1	3	2530	SCR	ZTO1	SCRAPER	78	99
200005	2006-09-11	102	1	HAM	304	36	4975	HAM	304	SIZE 4 HAMMER	300	0
200006	2003-05-28	107	10	HAM	307	6	32400	HAM	307	HAMMER INSTALLER	360	365
200007	2009-03-03	112	8	CHI	47AG	11	50000	CHI	47AG	RIGHT CHISEL	5246	3



EXAMPLES

- List all the orders, showing their order numbers, prices and product types

```
SELECT ORDER_NUM, ORD_PRICE, TYPE
FROM ORDERS, PRODUCTS
WHERE MAN = MAN_ID AND PROD = PROD_ID;
```

- In SQL 92 syntax

EXAMPLES

- List all the orders, showing their order numbers, prices and product types

```
SELECT ORDER_NUM, ORD_PRICE, TYPE
FROM ORDERS, PRODUCTS
WHERE MAN = MAN_ID AND PROD = PROD_ID;
```

- In SQL 92 syntax

```
SELECT ORDER_NUM, ORD_PRICE, TYPE
FROM ORDERS JOIN PRODUCTS
ON MAN = MAN_ID AND PROD = PROD_ID;
```

EXAMPLES

- List name of each company and name of its sales representative together with location and region of the office of the representative.

EXAMPLES

- List name of each company and name of its sales representative together with location and region of the office of the representative.

```
SELECT COMP_NAME, FL_NAME, LOCATION, REGION
FROM CUSTOMERS, EMPLOYEES, OFFICES
WHERE   CST_REP=EMP_ID AND
        OFFICE=OFFICE_ID
```

EXAMPLES

- List the orders whose price is over $25,000 by printing the order number, price, the name of the employee who took the order and the name of the company who placed it

EXAMPLES

- List the orders whose price is over $25,000 by printing the order number, price, the name of the employee who took the order and the name of the company who placed it

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
            FL_NAME
FROM ORDERS, CUSTOMERS, EMPLOYEES
WHERE CUST_NUM=CUST_ID AND
            REP_NUM=EMP_ID  AND
            ORD_PRICE > 25000
```

EXAMPLES

- Classical syntax

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
            FL_NAME
FROM ORDERS, CUSTOMERS, EMPLOYEES
WHERE CUST_NUM=CUST_ID AND REP_NUM=EMP_ID
            AND ORD_PRICE > 25000
```

- Using SQL92 Syntax

EXAMPLES

- Classical syntax

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
        FL_NAME
FROM ORDERS, CUSTOMERS, EMPLOYEES
WHERE CUST_NUM=CUST_ID AND REP_NUM=EMP_ID
        AND ORD_PRICE > 25000
```

- Using SQL92 Syntax

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
        FL_NAME
FROM ORDERS JOIN CUSTOMERS ON CUST_NUM = CUST_ID
        JOIN EMPLOYEES ON REP_NUM = EMP_ID
        WHERE ORD_PRICE > 25000
```

EXAMPLES

- List the orders whose priceis over $25,000 showing the order number, price, the name of the customer who placed the order and the name of the employee assigned to that customer.

EXAMPLES

- List the orders whose priceis over $25,000 showing the order number, price, the name of the customer who placed the order and the name of the employee assigned to that customer.

```
SELECT ORDER_NUM, ORD_PRICE, COMP_NAME,
            FL_NAME
FROM ORDERS, CUSTOMERS, EMPLOYEES
WHERE   CUST_NUM = CUST_ID AND
            CST_REP = EMP_ID AND
            ORD_PRICE > 25000
```

QUALIFYING *

QUALIFYING *

- List all information of salespeople together with the location and region they work
- Obvious solution is

```
SELECT EMP_ID, FL_NAME, AGE, OFFICE, TITLE,
       HIRE_DATE, SUPERVISOR, EMP_TARGET,
       EMP_SALES, LOCATION, REGION
FROM EMPLOYEES, OFFICES
WHERE OFFICE = OFFICE_ID
```

QUALIFYING *

- But it is not nice to write all column names of the EMPLOYEES table
- We can qualify * with table name instead

```
SELECT EMPLOYEES.*, LOCATION, REGION
FROM EMPLOYEES, OFFICES
WHERE OFFICE = OFFICE_ID
```

RECURSIVE RELATIONS AND SELFJOINS

RECURSIVE RELATIONS AND SELFJOINS

- If we need to make use of a one-to-many recursive relation, we need to take join of a table by itself
- **Example:** List the name of each employee and that of his/her manager
- We cannot use the following query due to confusion in column names

```
SELECT FL_NAME, FL_NAME
FROM EMPLOYEES, EMPLOYEES
WHERE SUPERVISOR = EMP_ID
```

RECURSIVE RELATIONS AND SELFJOINS

- To alleviate this problem, we can use *aliases* (tag names) for tables.
- Aliases can be given for a table by writing it just after the name of the table in FROM clause

```
SELECT EMPS.FL_NAME, MGRS.FL_NAME
FROM EMPLOYEES EMPS, EMPLOYEES MGRS
WHERE EMPS. SUPERVISOR = MGRS. EMP_ID
```

- It is just like we have two identical copies of EMPLOYEES table with different names (EMPS and MGRS)

OUTER JOINS

OUTER JOINS

- Recall the following example
- **Example:** List the names of employees and the cities where they work

FL_NAME	LOCATION
Vanessa Abrams	Seattle
Blair Waldorf	Houston
Carter Baizen	Seattle
Nathaniel Archibald	Seattle
Ivy Dickens	Detroit
Daniel Humphery	Seattle
Georgina Sparks	Miami
Charles Bass	Houston
Serena Woodsen	Boston

- We cannot see the name of Ben Donovan since he is not assigned to an office

OUTER JOINS

- What if we want to see all employees in the result even if they are not assigned to an office appearing in OFFICES table
- In such situations we can use OUTER JOINS

```
SELECT FL_NAME, LOCATION
FROM EMPLOYEES
    LEFT OUTER JOIN OFFICES
    ON OFFICE = OFFICE_ID
```

FL_NAME	LOCATION
Ben Donovan	NULL
Vanessa Abrams	Seattle
Blair Waldorf	Houston
Carter Baizen	Seattle
Nathaniel Archibald	Seattle
Ivy Dickens	Detroit
Daniel Humphery	Seattle
Georgina Sparks	Miami
Charles Bass	Houston
Serena Woodsen	Boston

OUTER JOINS

- Outer join returns the results of inner join together with the rows of a table which does not match with a row in the other table
- There are three type of outer joins

LEFT OUTER JOIN	RIGHT OUTER JOIN	FULL OUTER JOIN
INNER JOIN + Unmatched elements from left table	INNER JOIN + Unmatched elements from right table	INNER JOIN + Unmatched elements from left table + Unmatched elements from right table

OUTER JOINS

- Example:

CUSTOMERS	
NAME	CITY
Mary	Boston
John	NULL
Susan	Chicago
Sam	Chicago
James	Denver

REPRESENTATIVES	
NAME	CITY
Nancy	Boston
Henry	Boston
George	NULL
Betty	Chicago
Anne	Dallas

- List the customers and representatives living in the same city as pairs

OUTER JOINS

- Inner join

```
SELECT *
FROM CUSTOMERS JOIN REPRESENTATIVES
ON CUSTOMERS.CITY = REPRESENTATIVES.CITY
```

CUSTOMERS	
NAME	CITY
Mary	Boston
John	NULL
Susan	Chicago
Sam	Chicago
James	Denver

REPRESENTATIVES	
NAME	CITY
Nancy	Boston
Henry	Boston
George	NULL
Betty	Chicago
Anne	Dallas

INNER JOIN

NAME	CITY	NAME	CITY
Mary	Boston	Nancy	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Betty	Chicago
Sam	Chicago	Betty	Chicago

OUTER JOINS

- Left outer join

  SELECT *

  FROM CUSTOMERS LEFT OUTER JOIN REPRESENTATIVES

  ON CUSTOMERS.CITY = REPRESENTATIVES.CITY

CUSTOMERS	
NAME	CITY
Mary	Boston
John	NULL
Susan	Chicago
Sam	Chicago
James	Denver

REPRESENTATIVES	
NAME	CITY
Nancy	Boston
Henry	Boston
George	NULL
Betty	Chicago
Anne	Dallas

LEFT OUTER JOIN

NAME	CITY	NAME	CITY
Mary	Boston	Nancy	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Betty	Chicago
Sam	Chicago	Betty	Chicago
John	NULL	NULL	NULL
James	Denver	NULL	NULL

OUTER JOINS

- Right outer join

```
SELECT *
FROM CUSTOMERS RIGHT OUTER JOIN REPRESENTATIVES
ON CUSTOMERS.CITY = REPRESENTATIVES.CITY
```

CUSTOMERS	
NAME	CITY
Mary	Boston
John	NULL
Susan	Chicago
Sam	Chicago
James	Denver

REPRESENTATIVES	
NAME	CITY
Nancy	Boston
Henry	Boston
George	NULL
Betty	Chicago
Anne	Dallas

RIGHT OUTER JOIN

NAME	CITY	NAME	CITY
Mary	Boston	Nancy	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Betty	Chicago
Sam	Chicago	Betty	Chicago
NULL	NULL	George	NULL
NULL	NULL	Anne	Dallas

OUTER JOINS

- Full outer join

```
SELECT *
FROM CUSTOMERS FULL OUTER JOIN REPRESENTATIVES
ON CUSTOMERS.CITY = REPRESENTATIVES.CITY
```

CUSTOMERS	
NAME	CITY
Mary	Boston
John	NULL
Susan	Chicago
Sam	Chicago
James	Denver

REPRESENTATIVES	
NAME	CITY
Nancy	Boston
Henry	Boston
George	NULL
Betty	Chicago
Anne	Dallas

FULL OUTER JOIN

NAME	CITY	NAME	CITY
Mary	Boston	Nancy	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Betty	Chicago
Sam	Chicago	Betty	Chicago
John	NULL	NULL	NULL
James	Denver	NULL	NULL
NULL	NULL	George	NULL
NULL	NULL	Anne	Dallas