SQL (JOIN statement)

Learning Outcomes

- Be able to retrieve data from database and formulate queries using JOIN operators, including:
  - NATURAL JOIN
  - JOIN USING
  - JOIN ON
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN

Connecting tables with JOIN

Often need to use two or more tables as below:

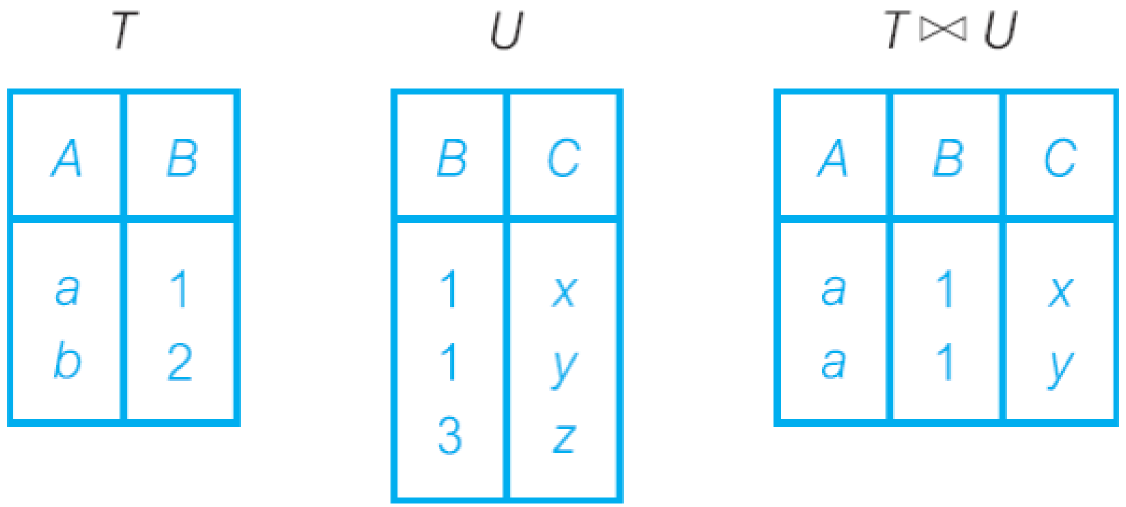
```
SELECT c.clientNo, fName, lName,
       propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

It is like the theta join $\bowtie_T$ in relational algebra with the where clause as the “predicate/condition” (T).

But Natural Join $\bowtie$ often did what we wanted easily.

Natural Join (Relational algebra reminder)

- R⋈S
  - The result of the natural join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names.



T

A	*B*
a	1
b	2

U

B	*C*
1	*x*
1	*y*
3	*z*

T ⋈ *U*

A	*B*	*C*
a	1	*x*
a	1	*y*

(g) Natural join

Natural join example

Students

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019
Hengda Yin	20161333	e-Comm	2019

Years

Year	Animal
2018	Dog
2019	Pig
2020	Rat
2021	Ox

```
SELECT * FROM
     Students NATURAL JOIN Years;
```

Students $\bowtie$ Years

Name	ID	Course	Year	Animal
Yang Wang	20184123	IoT	2018	Dog
Yixin Xi	20184113	Telecomm	2019	Pig
Zhou Li	20164888	Telecomm	2019	Pig
Hengda Yin	20161333	e-Comm	2019	Pig

SQL NATURAL JOIN

Client

Name	Cid	City
Yang Wang	4123	Beijing
Yixin Xi	4113	Beijing
Zhou Li	4888	Shanghai
Hengda Yin	1333	Xian

Meeting

Cid	City	Date
4123	Beijing	20th Mar 22
4123	Xian	21st Mar 22
4888	Shanghai	22nd Mar 22
4233	Beijing	23rd Mar 22

```
SELECT * FROM
      Client NATURAL JOIN Meeting;
```

Name	Cid	City	Date
Yang Wang	4123	Beijing	20th Mar 22
Zhou Li	4888	Shanghai	22nd Mar 22

Here a natural join may not be what we want…

NATURAL JOIN in SQL

- NATURAL JOIN – join on attributes with same name

Client(Name, Cid, City)

Meeting(Cid, City, Date)

```
SELECT * FROM
    Client NATURAL JOIN Meeting;
```

Produces a table with attributes:

Name, Cid, City, Date

SQL JOIN USING

Client

Name	Cid	City
Yang Wang	4123	Beijing
Yixin Xi	4113	Beijing
Zhou Li	4888	Shanghai
Hengda Yin	1333	Xian

Meeting

Cid	City	Date
4123	Beijing	20th Mar 22
4123	Xian	21st Mar 22
4888	Shanghai	22nd Mar 22
4233	Beijing	23rd Mar 22

```
SELECT * FROM
    Client c JOIN Meeting m USING (Cid);
```

Note brackets – could be a list e.g. `USING (Cid,City)`

Name	Cid	c.City	m.City	Date
Yang Wang	4123	Beijing	Beijing	20th Mar 22
Yang Wang	4123	Beijing	Xian	20th Mar 22
Zhou Li	4888	Shanghai	Shanghai	22nd Mar 22

SQL JOIN ON (doesn't combine column)

Client

Name	Cid	City
Yang Wang	4123	Beijing
Yixin Xi	4113	Beijing
Zhou Li	4888	Shanghai
Hengda Yin	1333	Xian

Meeting

Cid	City	Date
4123	Beijing	20th Mar 22
4123	Xian	21st Mar 22
4888	Shanghai	22nd Mar 22
4233	Beijing	23rd Mar 22

```
SELECT * FROM
    Client c JOIN Meeting m ON c.Cid=m.Cid;
```

Name	c.Cid	m.Cid	c.City	m.City	Date
Yang Wang	4123	4123	Beijing	Beijing	20th Mar 22
Yang Wang	4123	4123	Beijing	Xian	20th Mar 22
Zhou Li	4888	4888	Shanghai	Shanghai	22nd Mar 22

Same as `SELECT * FROM Client c,Meeting m`

```
WHERE c.Cid=m.Cid;
```

Other uses of JOIN

- The JOINS we looked at so far are all known as INNER JOIN (results only when columns match)
- There are two more types
- CROSS JOIN (this is the Cartesian Product)
- Outer joins (insert NULL when no match)
  - LEFT JOIN – if left table has no match add NULL
  - RIGHT JOIN – if right table has no match add NULL
  - FULL JOIN – if either table has no match add NULL

CROSS JOIN (Cartesian product)

Students

Name	ID
Yang Wang	20184123
Yixin Xi	20184113
Zhou Li	20164888

Courses

Course	Year
IoT	2018
Telecomm	2019

```
SELECT * FROM
Students CROSS JOIN Courses;
```

Students X Courses

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	IoT	2018
Zhou Li	20164888	IoT	2018
Yang Wang	20184123	Telecomm	2019
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019

SQL LEFT JOIN

Client

Name	Cid
Yang Wang	4123
Yixin Xi	4113
Zhou Li	4888
Hengda Yin	1333

Meeting

Cid	City
4123	Beijing
4123	Xian
4888	Shanghai
4233	Beijing

```
SELECT * FROM Client c
 LEFT JOIN Meeting m ON c.Cid = m.Cid;
```

Name	c.Cid	m.Cid	City
Yang Wang	4123	4123	Beijing
Yang Wang	4123	4123	Xian
Zhou Li	4888	4888	Shanghai
Yixin Xi	4113	NULL	NULL
Hengda Yin	1333	NULL	NULL

SQL RIGHT JOIN

Client

Name	Cid
Yang Wang	4123
Yixin Xi	4113
Zhou Li	4888
Hengda Yin	1333

Meeting

Cid	City
4123	Beijing
4123	Xian
4888	Shanghai
4233	Beijing

```
SELECT * FROM Client c
  RIGHT JOIN Meeting m ON c.Cid = m.Cid;
```

Name	c.Cid	m.Cid	City
Yang Wang	4123	4123	Beijing
Yang Wang	4123	4123	Xian
Zhou Li	4888	4888	Shanghai
NULL	NULL	4233	Beijing

SQL FULL JOIN

Client

Name	Cid
Yang Wang	4123
Yixin Xi	4113
Zhou Li	4888
Hengda Yin	1333

Meeting

Cid	City
4123	Beijing
4123	Xian
4888	Shanghai
4233	Beijing

```
SELECT * FROM Client c
   FULL JOIN Meeting m ON c.Cid = m.Cid;
```

Name	c.Cid	m.Cid	City
Yang Wang	4123	4123	Beijing
Yang Wang	4123	4123	Xian
Zhou Li	4888	4888	Shanghai
Yixin Xi	4113	NULL	NULL
Hengda Yin	1333	NULL	NULL
NULL	NULL	4233	Beijing

What have we learned?

- A NATURAL JOIN B joins on all duplicate attributes leaving a table with no duplicate attributes
- A JOIN B USING (c) joins on attribute c only leaving a table with no duplicate attribute c.
- A JOIN B ON A.Attribute = B.Attribute is just the same as SELECT … FROM A, B WHERE A.Attribute = B.Attribute
- A CROSS JOIN B produces the Cartesian product
- Outer JOINS (LEFT JOIN, RIGHT JOIN, FULL JOIN) allow us to add NULL entries where there is no match.