Join clauses

- The learning objectives for this week are:
  - Knowing what *join clauses* are and what kind of query problems can they solve
  - Knowing how to use the `INNER JOIN`, `OUTER JOIN` and `CROSS JOIN` clauses to perform different kind of joins operations
  - Knowing the difference between `INNER JOIN` and `OUTER JOIN` clauses

Join clauses

- In the relational model a table can have a foreign key referencing primary key in another table
- A common query problem is to select colums from the referenced table based on the foreign key value
- For example, "*What is the name of each course instance's teacher?*"
- We would need to select course instance columns from the `CourseInstance` table and *join* them with the teacher columns from the `Teacher` table based on the `teacher_number` foreign key column value
- The referential integrity and *join operations* are the key features which dinstuingish the relational database management systems from other database management systems

Join clauses

- The first table resembles the `CourseInstance` table row, the second table the `Teacher` table row and the third table the desired result table row:

course_code	instance_number	teacher_number
a290	1	🔑 h430

teacher_number	first_name	surname
🔑 h430	Emma	Virta

course_code	instance_number	teacher_number	first_name	surname
a290	1	🔑 h430	Emma	Virta

Join clauses

```
SELECT [ DISTINCT ] {
  -- ...
}
FROM table_name [ [ AS ] table_alias ]
-- JOIN clause
[ { [ INNER ] JOIN table_name [ [ AS ] table_alias ] ON join_condition }... ]
```

- Instead of combining rows, like set operators (e.g. `UNION`), a *join clause* combines *columns* from one or more tables into a new table
- Rows are join based on a condition called *join condition*
- There's three different kind of *join operations* which operate in different ways: *inner join*, *outer join* and *cross join*

Join clauses

- With a `SELECT` stament we get the `teacher_number` foreign key column value:

```
SELECT course_code, instance_number, teacher_number
FROM CourseInstance
```

course_code	instance_number	teacher_number
a290	1	🔑 h430

Join clauses

- We can use the `INNER JOIN` clause to combine the matching columns from the `Teacher` table:

```
-- what is the first name and surname of each course instance teacher?
SELECT
CourseInstance.course_code, CourseInstance.instance_number, Teacher.teacher_number, Teacher.first_name, Teacher.surname
FROM CourseInstance
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	🔑 h430	Emma	Virta

Join clauses

- In the example each row of the `CourseInstance` table is combined with a row from the `Teacher` table based on the *join condition*:

```
-- the teacher_number of column in the CourseIntance table
-- must match the teacher_number column of the Teacher table
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

- ⚠️ The join condition *does not* have to compare primary key to a foreign key, any kind of condition can be used

Join clauses

- With join clauses we operate on multiple tables, which can have columns with the same name
- This leads to errors caused by *ambiguous column names*, which can be avoided by specifying the table name before the column name using the `table_name.column_name` syntax:

```
-- ❌ teacher_number column name is ambiguous because
-- both CourseInstance and Teacher table have the teacher_number column
SELECT teacher_number
FROM CourseInstance
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number

-- ✅ we specify that the teacher_number column
-- of the CourseInstance table should be selected
SELECT CourseInstance.teacher_number
FROM CourseInstance
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

Join clauses

- If we want to get columns from more than two table, we can chain multiple join clauses together
- For example, in the following example we need information from the `CourseGrade`, `Course` and `Teacher` tables:

```
-- display course and student information for each passing grade
SELECT CourseGrade.course_code, instance_number, CourseGrade.student_number first_name, surname, credits, grade
FROM CourseGrade
INNER JOIN Course ON CourseGrade.course_code = Course.course_code
INNER JOIN Student ON CourseGrade.student_number = Student.student_number
WHERE grade > 0
```

INNER JOIN clause

- The `INNER JOIN` clause (or `JOIN` in short) *only* selects rows that have matching values in *both* tables based on the join condition
- If we consider the previous example, this means that course instances without teacher number (`teach_number` column value is `NULL`) won't be included in the result table
- This is because we can't match `teacher_number` of value `NULL` with a row in the `Teacher` table because the primary key value can't be `NULL`

INNER JOIN clause

- Let's consider the following rows in `CourseInstance` and `Teacher` tables:

course_code	instance_number	teacher_number
a290	1	h430
a290	2	⚠️ NULL
a450	1	h303

teacher_number	first_name	surname
h430	Emma	Virta
h303	Veli	Ponteva
h777	Mauri	Matikka

INNER JOIN clause

- The result table only has rows that have the corresponding `teacher_number` column value in the `Teacher` table

```
SELECT
CourseInstance.course_code, CourseInstance.instance_number,
Teacher.teacher_number, Teacher.first_name, Teacher.surname
FROM CourseInstance
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	h430	Emma	Virta
a450	1	h303	Veli	Ponteva

INNER JOIN clause

- If the foreign key is a composite key, the join condition must include all columns of the composite key
- For example, the `CourseInstance` table has a composite key consisting of `course_code` and `instance_number` columns

```
-- how long does it take to grade each course on average?
SELECT
  CourseInstance.course_code,
  AVG(DATEDIFF(DAY, end_date, grade_date)) AS average_grading_time
FROM CourseGrade
-- join condition must include all columns of the composite key
INNER JOIN CourseInstance ON CourseGrade.course_code = CourseInstance.course_code
AND CourseGrade.instance_number = CourseInstance.instance_number
GROUP BY CourseInstance.course_code
```

OUTER JOIN clause

- The `OUTER JOIN` clause selects *matching* and *non-matching rows* from either or both tables
- The `OUTER JOIN` clause has two variations: `LEFT OUTER JOIN` and `RIGHT OUTER JOIN`
- The difference between these two lies in the inclusion of non-matching rows
- The `LEFT OUTER JOIN` clause (or `LEFT JOIN` in short) includes the non-matching rows from the table which is on the *left* of the join clause
- The `RIGHT OUTER JOIN` clause (or `RIGHT JOIN` in short) includes the non-matching rows from the table which is on the *right* of the join clause

OUTER JOIN clause

- The "left table" is before the join clause and the "right table" after it:

```
SELECT -- ...
FROM LeftTable
LEFT OUTER JOIN RightTable
ON -- ...
```

LEFT OUTER JOIN clause

- With the `LEFT OUTER JOIN` clause the result table has *all* rows from the `CourseInstance` table *and the matching rows* from the `Teacher` table

```
SELECT
CourseInstance.course_code, CourseInstance.instance_number, Teacher.teacher_number, Teacher.first_name, Teacher.surname
FROM CourseInstance
LEFT OUTER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	h430	Emma	Virta
a290	2	⚠️ NULL	⚠️ NULL	⚠️ NULL
a450	1	h303	Veli	Ponteva

RIGHT OUTER JOIN clause

- With the `RIGHT OUTER JOIN` clause the result table has *all* rows from the `Teacher` table *and the matching rows* from the `CourseIntance` table

```
SELECT
CourseInstance.course_code, CourseInstance.instance_number, Teacher.teacher_number, Teacher.first_name, Teacher.surname
FROM CourseInstance
RIGHT OUTER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	h430	Emma	Virta
a450	1	h303	Veli	Ponteva
⚠️ NULL	⚠️ NULL	h777	Mauri	Matikka

OUTER JOIN clause

- Technically, every `RIGHT OUTER JOIN` clause can be handled with a `LEFT OUTER JOIN` clause
- This is because `TableA RIGHT OUTER JOIN TableB` is the same as `TableB LEFT OUTER JOIN TableA`
- It might be easier to think every outer join operation as a `LEFT OUTER JOIN` clause and not to use `RIGHT OUTER JOIN` clause

Difference between INNER JOIN and OUTER JOIN

- The difference between `INNER JOIN` and `OUTER JOIN` lies in the way the *non-matching rows are handled*

course_code	instance_number	teacher_number
a290	1	h430
a290	2	⚠️ NULL
a450	1	h303

teacher_number	first_name	surname
h430	Emma	Virta
h303	Veli	Ponteva
h777	Mauri	Matikka

Difference between INNER JOIN and OUTER JOIN

- The `INNER JOIN` clause *omits the rows that don't have a matching row* in the other table:

```
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	h430	Emma	Virta
a450	1	h303	Veli	Ponteva

- ⚠️ The course instance with the `teacher_number` columns value as `NULL` is omitted from the result table because it doesn't have a matching row in the `Teacher` table

Difference between INNER JOIN and OUTER JOIN

- `OUTER JOIN` *includes the rows that don't have a matching row* in the other table and the corresponding columns values are `NULL`

```
LEFT OUTER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

course_code	instance_number	teacher_number	first_name	surname
a290	1	h430	Emma	Virta
a290	2	⚠️ NULL	⚠️ NULL	⚠️ NULL
a450	1	h303	Veli	Ponteva

Join clause shorthands

- We usually prefer the shorthand versions of the join clauses
- The `INNER JOIN` clause can be written as `JOIN`
- The `LEFT OUTER JOIN` clause can be written as `LEFT JOIN`
- The `RIGHT OUTER JOIN` clause can be written as `RIGHT JOIN`

```
INNER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number

LEFT OUTER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
LEFT JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number

RIGHT OUTER JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
RIGHT JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
```

CROSS JOIN clause

- The `CROSS JOIN` clause selects rows from *both* tables *without a join condition*
- The `CROSS JOIN` clause operates similarly as the *cartesian product*
- The result table has every possible combination of rows of the first and the second table
- ⚠️ The result table can potentially have a very large number of rows

```
-- what are all possible shoe color and size combinations?
SELECT ShoeColor.name, ShoeSize.name
FROM ShoeColor
CROSS JOIN ShoeSize
```

Examples of join clauses

- What do we get from the following queries?

```sql
-- ❓ what do we get from this query?
SELECT course_code, instance_number
FROM CourseInstance
JOIN Teacher ON CourseInstance.teacher_number = Teacher.teacher_number
JOIN Campus ON Teacher.campus_code = Campus.campus_code
WHERE Campus.campus_name = 'Pasila'

-- ❓ what do we get from this query?
SELECT CourseGrade.course_code, CourseGrade.instance_number, AVG(grade) as average_grade
FROM CourseGrade
JOIN CourseInstance ON CourseGrade.course_code = CourseInstance.course_code
AND CourseGrade.instance_number = CourseInstance.instance_number
WHERE participants > 15
GROUP BY CourseGrade.course_code, CourseGrade.instance_number
```

Summary

- *Join clauses* combines *columns* from one or more tables into a new table
- The `INNER JOIN` clause *only* selects rows that have matching values in *both* tables based on the join condition
- The `LEFT OUTER JOIN` clause includes the non-matching rows from the table which is on the *left* of the join clause and the matching rows from the table on the right
- The `RIGHT OUTER JOIN` clause includes the non-matching rows from the table which is on the *right* of the join clause and the matching rows from the table on the left
- The difference between `INNER JOIN` and `OUTER JOIN` lies in the way the *non-matching rows are handled*
- The `CROSS JOIN` clause selects rows from *both* tables *without a join condition*