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

- To link rows between two tables, one table includes a **foreign key** column that references **primary key** in another table.
- A common query requirement is to fetch data from the referenced table by matching the foreign key in one table to the primary key in another
- For example, *"What is the name of each course instance's teacher?"*
- We need to select course instance rows from the `CourseInstance` table and **join** them with the teacher rows from the `Teacher` table based on the `teacher_number` foreign key column value
- The referential integrity and such **join operations** are the key features which distinguish the relational database management systems from other database management systems

Join clauses

> *"What is the name of each course instance's teacher?"*

- 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 }... ]
```

- **Join clause** combines **row** from one or more tables into a new table
- Rows are join based on a condition called **join condition**, which is commonly formulated to match the foreign with a primary key
- 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` statement 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

- If we want to include information, such as the name of the teacher (`first_name` and `surname` columns), we need to **join** the matching row from the `Teacher` table
- While constructing joins we need to be aware of the relationships between tables which makes the database diagram extremely useful

Join clauses

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

```
-- what is the first name and surname of each course instance teacher?
SELECT
course_code, instance_number,
-- ⚠️ note column name "Teacher.teacher_number", not just "teacher_number"
Teacher.teacher_number, first_name, surname
FROM CourseInstance
-- the join condition specifies which rows are combined
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 condition

- 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 condition with a composite foreign key

- 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 has it taken to grade each student?
SELECT
  CourseInstance.course_code,
  instance_number,
  student_number,
  -- the end_date column comes from the CourseInstance table
  DATEDIFF(DAY, end_date, grade_date) AS 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
```

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:

```sql
-- ❌ 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 tables, we can **chain multiple join clauses** together
- For example, in the following example we need information from the `CourseGrade`, `Course` and `Student` tables:

```
-- display name of the student and name of the course for each passing grade
SELECT
  Course.course_name
  CourseGrade.course_code,
  instance_number,
  CourseGrade.student_number,
  first_name,
  surname
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

course_code	instance_number	teacher_number
a290	1	🔗 h430
a290	2	⚠️ 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, first_name, 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

- ℹ️ With the `INNER JOIN` clause the order of tables in `FROM TableA INNER JOIN TableB` doesn't matter

OUTER JOIN clause

- The `OUTER JOIN` clause returns all matching rows between tables, **along with non-matching rows from the specified table**, filling unmatched columns with `NULL` values
- This is useful, if we want to select **all course instances** (including the ones with missing `teacher_number`) and the name of their teacher
- The `OUTER JOIN` clause has two variations: `LEFT OUTER JOIN` and `RIGHT OUTER JOIN`
- The difference between `INNER JOIN` and `OUTER JOIN` join opreations two lies in the **inclusion of non-matching rows**
- ❗ With the `OUTER JOIN` clauses the order of tables in `FROM TableA LEFT OUTER JOIN TableB` does matter

LEFT OUTER JOIN clause

- With the `LEFT OUTER JOIN` clause the result table has **all rows from the left table**, and the **matching rows from the right table**:

```
SELECT
course_code, instance_number,
Teacher.teacher_number, first_name, surname
--  !  order of tables matter!
FROM CourseInstance -- "left table"
LEFT OUTER JOIN Teacher -- "right table"
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

- The `RIGHT OUTER JOIN` clause operates exactly the same as the `LEFT OUTER JOIN` but **all rows from the right table** (the `RIGHT OUTER JOIN TableName` table) are included
- The following queries will have the same result:

```
SELECT teacher_number, campus_name
FROM Teacher -- "left table"
LEFT OUTER JOIN Campus -- "right table"
ON Teacher.campus_code = Campus.campus_code

-- same as RIGHT OUTER JOIN after flipping the order of tables
SELECT teacher_number, campus_name
FROM Campus -- "left table"
RIGHT OUTER JOIN Teacher -- "right table"
ON Teacher.campus_code = Campus.campus_code
```

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.color_name, ShoeSize.size_name
FROM ShoeColor
CROSS JOIN ShoeSize
```

color_name	size_name
red	Medium
red	Large
...	...

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**