Join clauses

- During this week we will learn:
  - What are join clauses and what kind of query problems can they solve
  - How to use the `INNER JOIN`, `OUTER JOIN` and `CROSS JOIN` clauses to perform different kind of joins operations

Join clauses

- Instead of combining rows, like set operators (e.g. `UNION`), a *join clause* combines *columns* from one or more tables into a new table
- There's three different kind of join operations which operate in different ways: *inner join*, *outer join* and *cross join*
- In the relational model a table can have a foreign key referencing primary key in another table
- A common query problem is to combine columns from the primary key table with the columns of the foreign key table
- For example, *what is the name of each course instance teacher?*

Join clauses

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

```
-- what is the teacher number of each course instance teacher?
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 *doesn't* have to compare primary key to a foreign key, any kind of condition can be used

Join clauses

- With join clauses, it is a good idea to specify the table name before the column name to avoid *ambiguous column names*:

```
-- ❌ 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
```

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

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

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

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 `CROSS JOIN` clause selects rows from *both* tables *without a join condition*