CS107 / AC207

SYSTEMS DEVELOPMENT FOR COMPUTATIONAL SCIENCE

LECTURE 22

Thursday, November 18th 2021

Fabian Wermelinger
Harvard University

RECAP OF LAST TIME

- Hands-on exercises using `python` and `sqlite3`
  - Reading data into tables
  - Queries
  - Sorting
  - Selecting columns
  - Altering tables
  - Aggregation
  - Deleting rows

OUTLINE

- Hands-on exercises using `sqlite3` and `pandas`
  - Table joins in SQL
  - SQL interface in `pandas`
  - SQL-like operations in `pandas`

SQLITE AND pandas EXERCISES (II)

- The exercise sheet is located at: https://harvard-iacs.github.io/2021-CS107/lectures/lecture22/
- You may work through the tasks in a Jupyter notebook.
- Commit your completed work to your class `git` repo inside the directory `lectures/lecture22` on the `main` (or `master`) branch.

Deliverables:

1. Make a copy of the exercise notebook and call it `L22_Exercises.ipynb`.
2. Do all exercises in *code cell(s) immediately after* the "Exercise" headings, similar to lecture 21. ***Note:*** to get the `pandas` tables to display in a cell, use `display()`.
3. Save and close your database(s) when done. Be sure to upload your database(s) with the lecture exercise notebook. You must name your database `L22DB.sqlite` and `L22DB_pandas.sqlite`.

TABLE JOINS IN SQL

- Last time we were practicing common operations on table data and tables itself.
- An often useful operation is to *join* two (or more) tables into a new table given optional constraints, called the *join-predicate*.
- The SQL specification defines a number of joins, most notably:
  - Inner join (the most common variant)
  - Outer joins (left, right and full)
- SQLite supports the *inner join* and *left outer join* of the SQL specification.

THE INNER JOIN

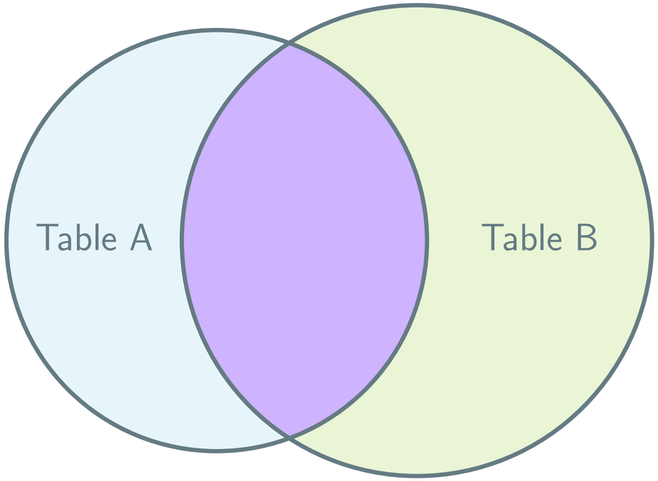
- Consider two tables A and B.
- The *inner join* is the resulting table of the *intersection* defined by the join-predicate between tables A and B.
- ***Example:*** consider the two tables:

Table A: *employees*

```
| ID | Name    | Office | Salary  |
|----|---------|--------|---------|
| 1  | Frank   | A12    | 45000.0 |
| 2  | Roberta | A10    | 80000.0 |
| 3  | Lory    | B07    | 50000.0 |
```

Table B: *bonuses*

```
| ID | Bonus     | EID |
|----|-----------|-----|
| 1  |    8000.0 | 1   |
| 2  |   10000.0 | 3   |
| 3  | 1000000.0 | 10  |
```



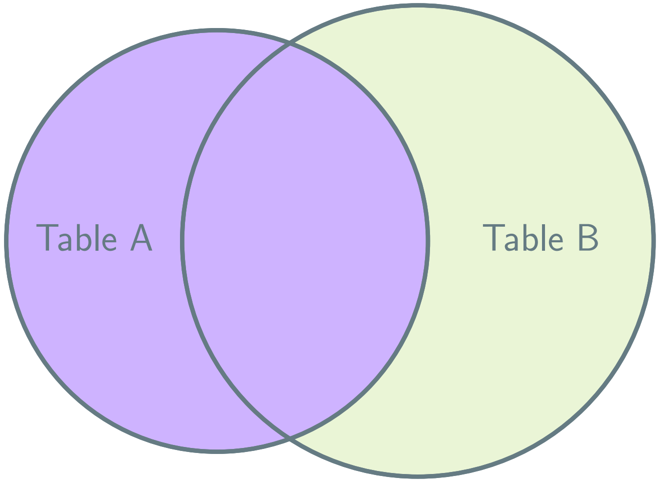
SQL command for inner join (purple region in Venn diagram):

```
SELECT * FROM A INNER JOIN B ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```

THE LEFT OUTER JOIN

- The same as inner join but also include all rows of the "left" table for which the join-predicate is false.
- Columns from the join table in rows that do not satisfy the join-predicate have `NULL` values.
- SQL command for left outer join (purple region in Venn diagram):

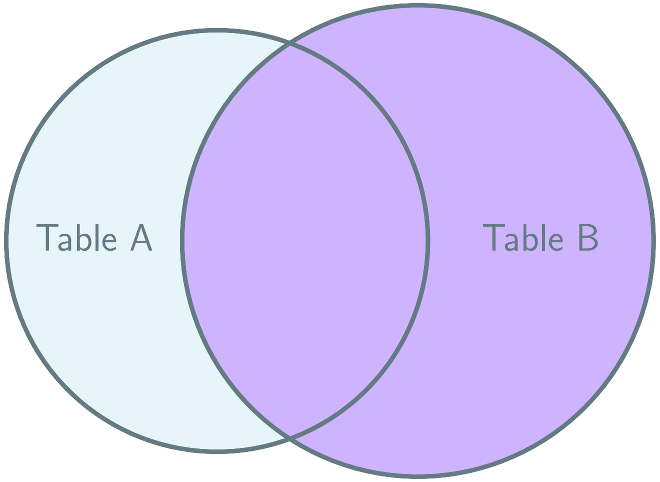
```sql
SELECT * FROM A LEFT OUTER JOIN B ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```



THE RIGHT OUTER JOIN

- The same as transposed left outer join.
- SQLite does not support this join, but can be achieved transposing the table arguments in the left outer join
- SQLite command for right outer join (purple region in Venn diagram):

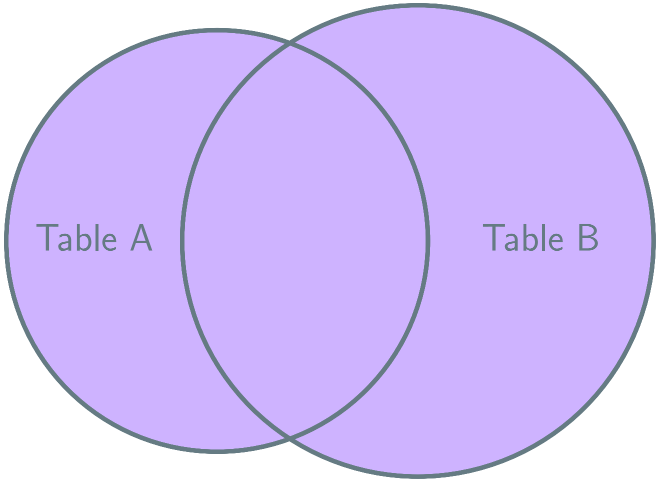
```
SELECT * FROM B LEFT OUTER JOIN A ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```



THE FULL OUTER JOIN

- The union of both tables.
- Less often used and not supported by SQLite.
- ***Careful:*** can produce large result tables.
- SQL command for full outer join (purple region in Venn diagram):

```sql
SELECT * FROM B FULL OUTER JOIN A ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```



EXAMPLE OUTPUTS FOR THE THREE SQLITE JOINS

***Table A:** employees*

```
| ID | Name    | Office | Salary  |
|----|---------|--------|---------|
| 1  | Frank   | A12    | 45000.0 |
| 2  | Roberta | A10    | 80000.0 |
| 3  | Lory    | B07    | 50000.0 |
```

***Table B:** bonuses*

```
| ID | Bonus     | EID |
|----|-----------|-----|
| 1  |    8000.0 | 1   |
| 2  |   10000.0 | 3   |
| 3  | 1000000.0 | 10  |
```

- Inner join (see `joins.sh`):

```
SELECT * FROM A INNER JOIN B ON B.EID = A.ID
```

```
ID  Name   Office  Salary   ID  Bonus    EID
--  -----  ------  -------  --  -------  ---
1   Frank  A12     45000.0  1   8000.0   1
3   Lory   B07     50000.0  2   10000.0  3
```

- Left outer join (see `joins.sh`):

```
SELECT * FROM A LEFT OUTER JOIN B ON B.EID = A.ID
```

```
ID  Name     Office  Salary   ID    Bonus    EID
--  -------  ------  -------  ----  -------  ----
1   Frank    A12     45000.0  1     8000.0   1
2   Roberta  A10     80000.0  NULL  NULL     NULL
3   Lory     B07     50000.0  2     10000.0  3
```

- Right outer join (see `joins.sh`):

```
SELECT * FROM B LEFT OUTER JOIN A ON B.EID = A.ID
```

```
ID  Bonus      EID  ID    Name   Office  Salary
--  ---------  ---  ----  -----  ------  -------
1   8000.0     1    1     Frank  A12     45000.0
2   10000.0    3    3     Lory   B07     50000.0
3   1000000.0  10   NULL  NULL   NULL    NULL
```