SYSTEMS DEVELOPMENT FOR COMPUTATIONAL SCIENCE

LECTURE 23

Fabian Wermelinger
Harvard University
CS107 / AC207

Thursday, November 17th 2022

LAST TIME

- Hands-on exercises using Python and SQLite:
  - Reading data into tables
  - Queries
  - Sorting
  - Selecting columns
  - Altering tables
  - Aggregation
  - Deleting rows

TODAY

Main topics: ***Databases, SQL and SQLite, Joins and Pandas***

Details:

- Hands-on exercises using SQLite and Pandas in Python:
  - Table joins in SQL
  - SQL interface in Pandas
  - SQL-like operations in Pandas

AGENDA CHECK:

- Milestone 2 deadline has been moved to *Tuesday, November 22nd 11:59pm*. You can find the milestone details at https://harvard-iacs.github.io/2022-CS107/project/M2.

SQLITE AND PANDAS EXERCISE II

- The exercise sheet and data is located in the class repository: https://code.harvard.edu/CS107/main/tree/master/lecture/code/lecture23
- We will work in a Jupyter notebook (`lecture23.ipynb`) for this exercise.

Deliverables:

1. Copy the Jupyter notebook (and the `fig` directory) into `lab/pp12` in your private Git repository and commit on your default branch. You should already have the `candidates.txt` and `contributors.txt` data files in this directory from last time.
2. For each exercise in the notebook, there are instructions labeled ***"Do the following:"***. Put all the code for those instructions in a *code cell(s) immediately following the instructions*. The code in that cell should be regular Python code. You should place comments where appropriate that describe your intentions.
   → ***Note:*** to get the Pandas tables to display in a cell, use `display()`.
3. Save and close your database. Be sure to upload your databases in `lab/pp12` as well. Please name your databases `lecture23.sqlite` and `lecture23_pandas.sqlite`.

SQLITE AND PANDAS EXERCISE II

- When you have completed both exercises (`lecture22.ipynb` and `lecture23.ipynb`) you can register for PP12 attendance and completion by using the form: https://forms.gle/B6jDg6qjFwwrq3eTA
- When you have registered on this form, ***you are not required to join your lab session for PP12***. *You can register on the form until 11:59pm tonight*.
- ***The requirements are:***
  1. In-class exercises for databases (Lecture 22 and 23) are completed and pushed to your private class repository in the directory `lab/pp12` on your default branch.
  2. Provide a link to the final commit for PP12 that has a *timestamp before 2022-11-17 11:59pm*. When you register on this form, commits pushed later than the deadline above will ***not be considered*** for submission grading. If the work is incomplete you may not obtain the completion credit.
  3. Please include both notebooks (`lecture22.ipynb` and `lecture23.ipynb`), the `*.txt` data files and the `*.sqlite` databases in your submission. To make `lecture23.ipynb` render correctly, you need to copy the `fig` directory provided in the handout as well.

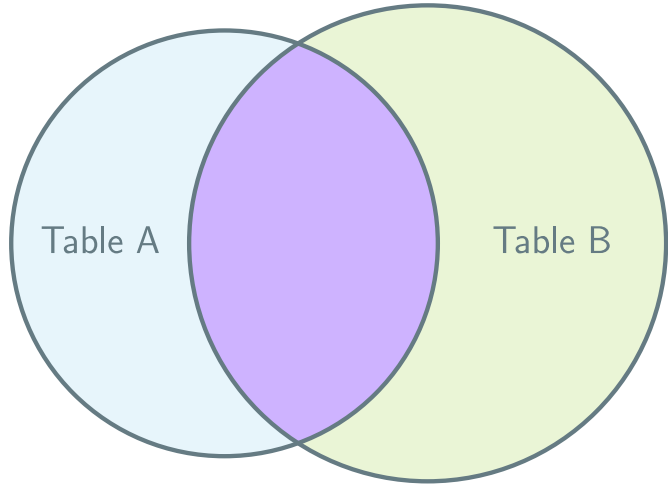
TABLE JOINS IN SQL

- Last time we were practicing common operations on table data and tables itself *(single tables only)*.
- Often you will need 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 only (sufficient for most operations → SQLite is *lightweight*).

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  |
```

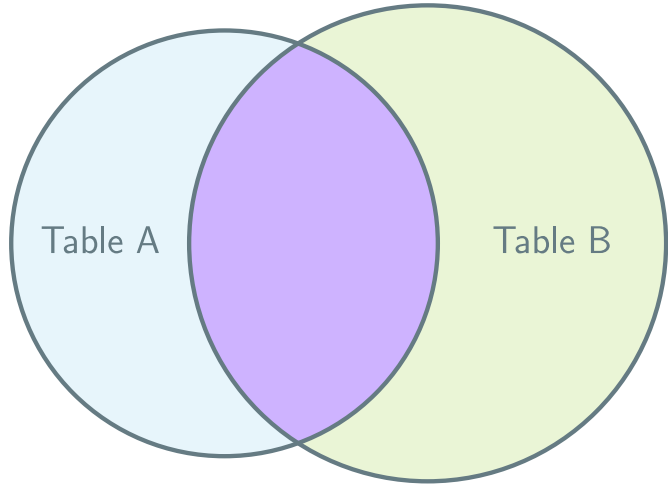
SQL command for inner join (purple region in Venn diagram):

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

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:*** alternative form

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

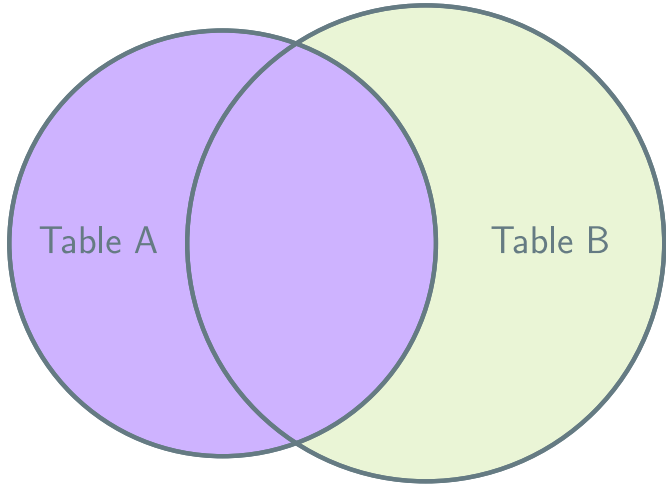
```
1  SELECT * FROM A INNER JOIN B ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```

The inner join is often written like this:

```
1  SELECT * FROM A, B WHERE 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.
- The left table may contain rows with columns joined from the right table for which the join-predicate is not satisfied. These column values are set to NULL *(see example below)*.

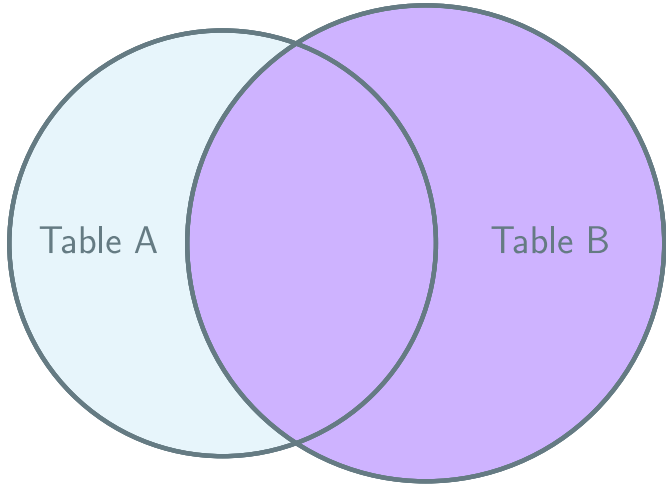


- *SQL command for left outer join (purple region in Venn diagram):*

```
1  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 a *transposed left outer join*.
- *SQLite does not support this join*, but can be achieved by ***transposing the table arguments*** in the left outer join.

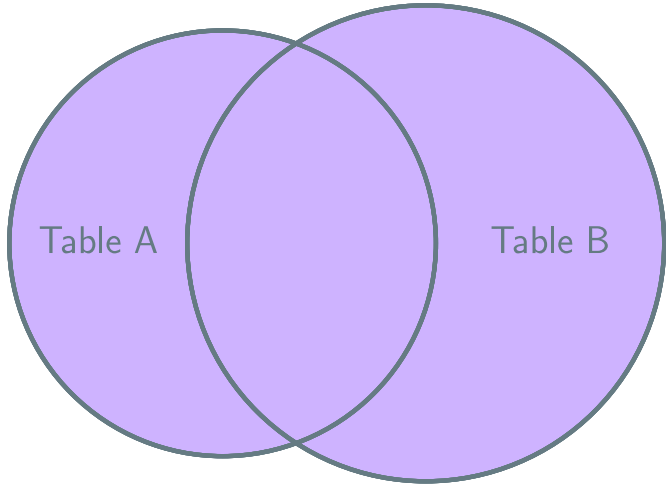


- *SQLite command for right outer join (purple region in Venn diagram):*

```
1  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 in SQLite*.
- ***Careful:*** can produce very large result tables.



- *SQL command for full outer join (purple region in Venn diagram):*

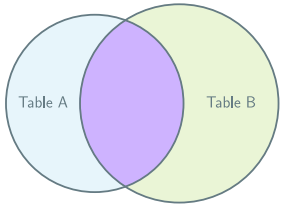
```
1  SELECT * FROM B FULL OUTER JOIN A ON B.EID = A.ID -- B.EID = A.ID is join-predicate
```

EXAMPLE OUTPUTS FOR THE THREE SQLITE JOINS

```
Table A:      | ID | Name    | Office | Salary  |
(employees)   |----|---------|--------|---------|
              | 1  | Frank   | A12    | 45000.0 |
              | 2  | Roberta | A10    | 80000.0 |
              | 3  | Lory    | B07    | 50000.0 |
```

```
Table B:    | ID | Bonus     | EID |
(bonuses)   |----|-----------|-----|
            | 1  |    8000.0 | 1   |
            | 2  |   10000.0 | 3   |
            | 3  | 1000000.0 | 10  |
```

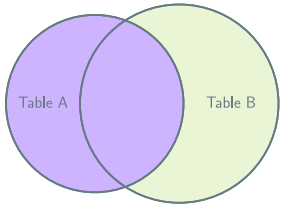
- Inner join (see `joins.sh`):




```
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
```

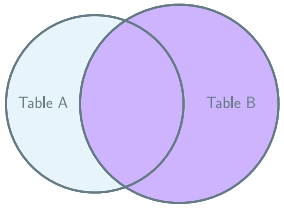
- Left outer join (see `joins.sh`):




```
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
```

RECAP

- Hands-on exercises using SQLite and Pandas in Python:
  - Table joins in SQL
  - SQL interface in Pandas
  - SQL-like operations in Pandas

Further reading:

- Additional SQL practice resource: https://www.sqlteaching.com/
- PostgreSQL tutorial: https://www.postgresqltutorial.com/
- Pandas comparison with SQL: https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_sql.html
- SQLite docs: https://www.sqlite.org/doclist.html
- SQLite3 module in Python (and tutorial): https://docs.python.org/3/library/sqlite3.html
- James R. Groff, Paul N. Weinberg and Andrew J. Oppel, *"SQL The Complete Reference"*, McGraw-Hill, 3rd edition, 2010