HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 12 – Access Databases using Python

Yuantao Fan
yuantao.fan@hh.se

Halmstad University

Overview

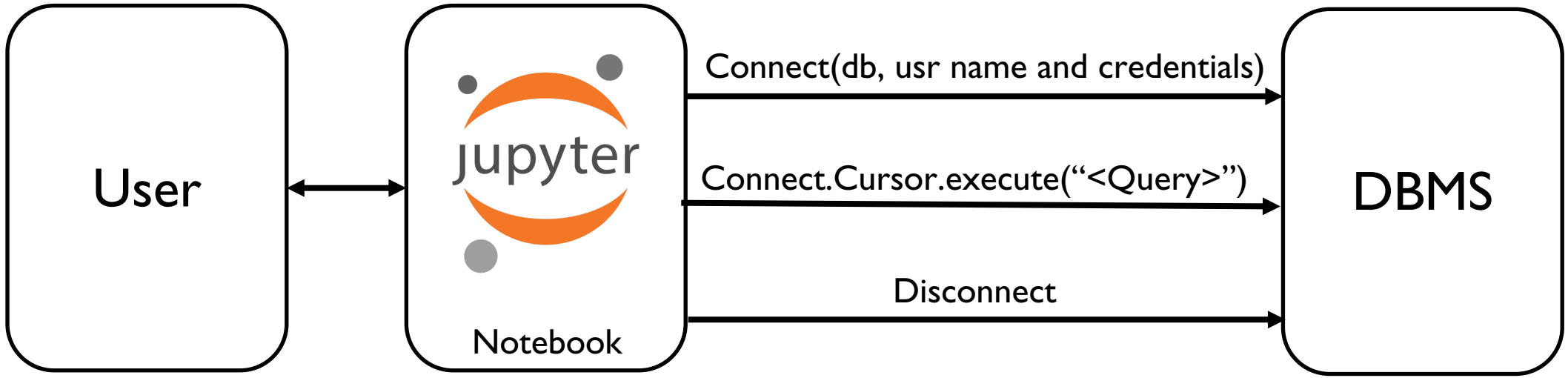
- Accessing databse using python
  - Basic concepts on accessing databases using Python
  - Load database and query data using SQL
  - Analyze and Visualize Data in Jupyter notebooks
- Practical tips on querying data

Why Accessing Databases with Python

- Well developed and supported packages for data analytics
  - Numpy, Scipy, Pandas, Scikit-learn, pytorch, matplotlib, seaborn, plotly…
- Widely accessible
- Fast prototyping
- Packages support accessing relational databases
- Database API (DB-API)
- Well documented
- Resources, e.g. tutorials

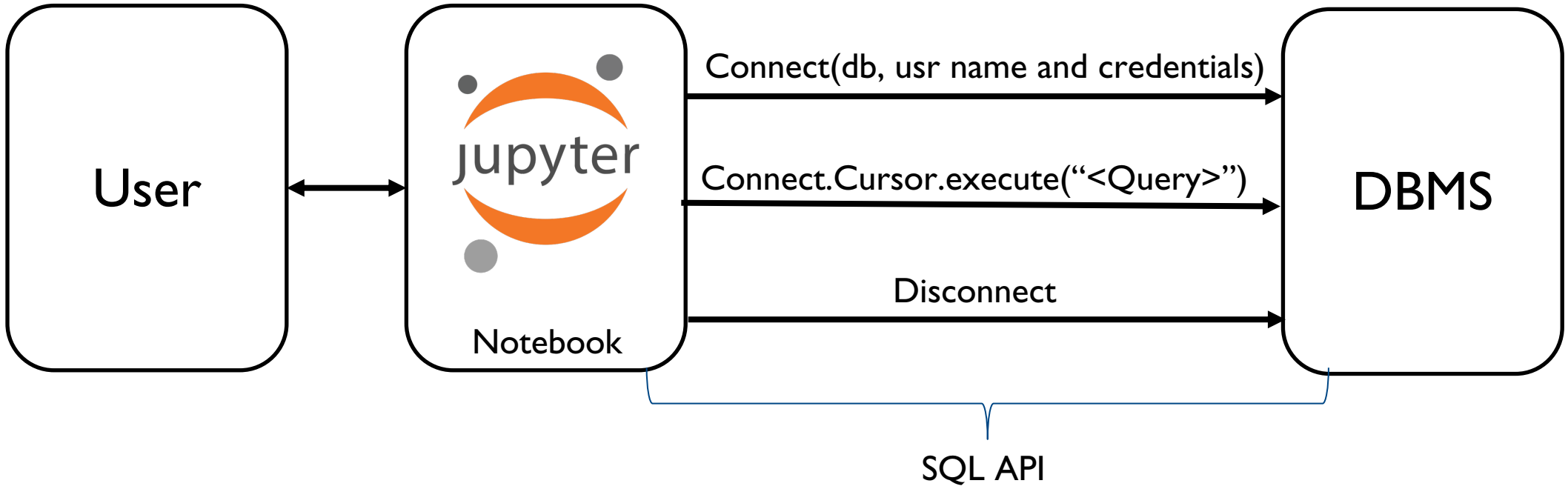
Why Jupyter Notebook?

- Interactive computing
- Visualization, e.g. results (e.g. plots) are embeded, along with the code
- Collaboration

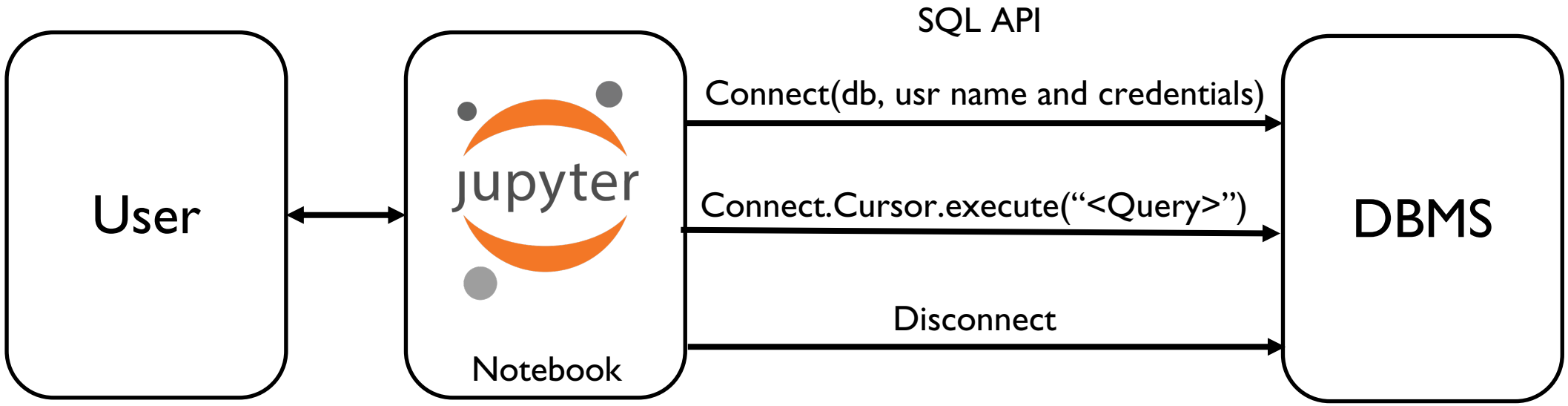


Why Jupyter Notebook?

- Interactive computing
- Visualization, e.g. results (e.g. plots) are embeded, along with the code
- Collaboration



DBMS SQL-based APIs



DBMS	SQL API
SQLite	sqlite3
PostgreSQL	psycopg2
MySQL	MySQL C API
IBM DB2	ibm_db
Oracle	OCI

Python DB-API

- Access relational databases with standardized API
  - Functions apply to any databases
  - Consistent and portable
- Allow accessing multiple relational databases

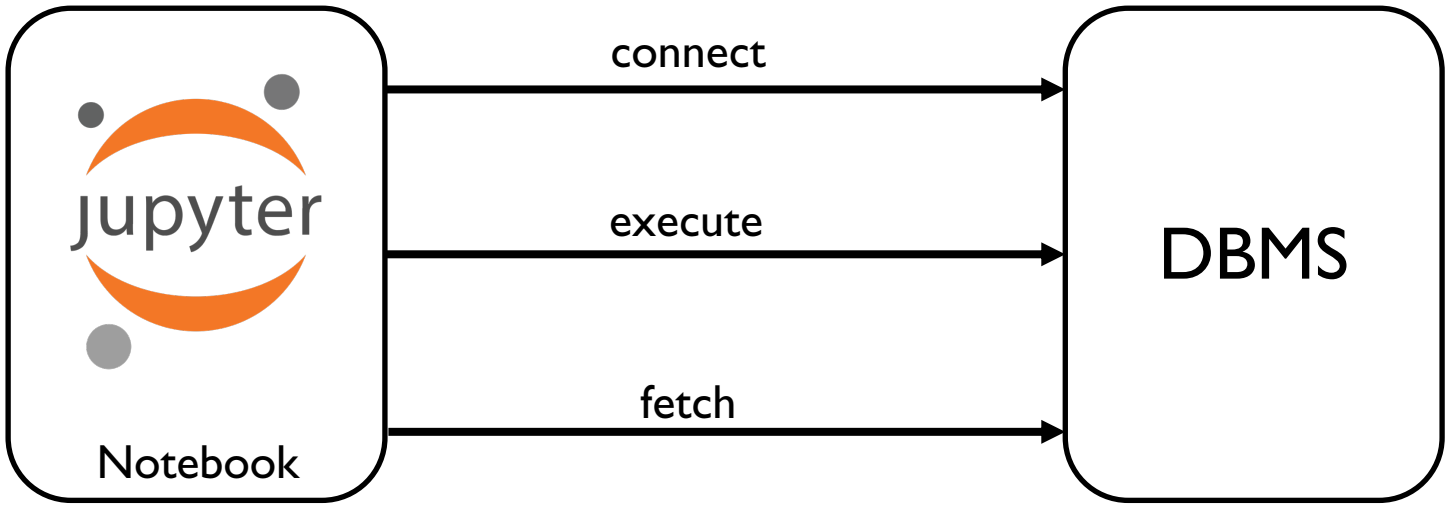
- Important concepts
  - Connection Objects
    - Database Connections
    - Transactions of data
  - Cursor Objects
    - Execute database queries
    - Scroll through and retrieve results

SQL API

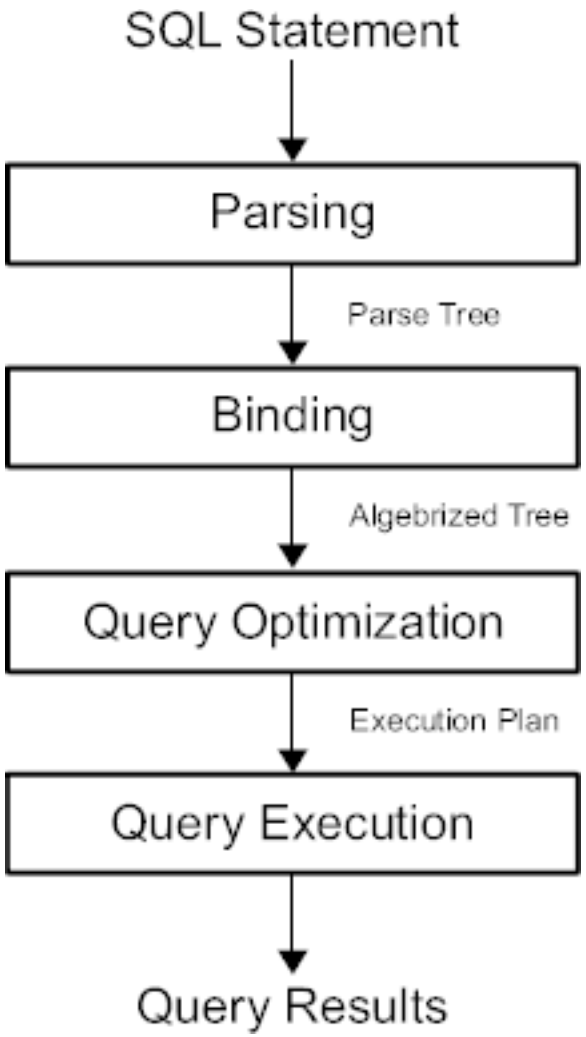
- Connection Objects
  - cursor() - A control structure enabling traversal over the records in a database (similar to a file handle)
  - commit() - commit pending transactions to the database
  - rollback() - roll back to the start of current pending transaction
  - close() - Close a database connection
- Cursor Methods
  - execute(), executemany()
  - fetchone(), fetchmany(), fetchall()
  - nextset()
  - callproc()
  - close()

SQL API

- Connection Objects
  - cursor() - A control structure enabling traversal over the records in a database (similar to a file handle)
  - commit() - commit pending transactions to the database
  - rollback() - roll back to the start of current pending transaction
  - close() - Close a database connection
- Cursor Methods
  - execute(), executemany()
  - fetchone(), fetchmany(), fetchall()
  - nextset()
  - callproc()
  - close()



Practical Tips on Query Optimization



Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *

```
SELECT *
FROM Teacher
```

```
SELECT full_name, age
FROM Teacher
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE

```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid;
```

```
SELECT t.full_name, tt.cid
FROM Teacher AS t
      INNER JOIN Teaches AS tt
      ON t.tid = tt.tid;
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT

```
SELECT primary_title, type
FROM titles
```

```
SELECT primary_title, type
FROM titles
LIMIT 10
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT
- USE wildcards wisely

```
SELECT primary_title, type
FROM titles
WHERE primary_title like “%Top%”
```

```
SELECT primary_title, type
FROM titles
WHERE primary_title like “Top%”
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT
- USE wildcards wisely
- Keep your queries simple

```
SELECT primary_title, type
FROM titles
WHERE type = " movie"
      OR type = "tvSeries"
      OR type = "videoGame"
```

```
SELECT primary_title, type
FROM titles
WHERE type IN (" movie", "tvSeries", "videoGame")
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT
- USE wildcards wisely
- Keep your queries simple

```
SELECT primary_title, type
FROM titles
WHERE premiered > 1980
    AND premiered < 2005
```

```
SELECT primary_title, type
FROM titles
WHERE premiered BETWEEN 1980 and 2005
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT
- USE wildcards wisely
- Keep your queries simple

```
SELECT primary_title, type
FROM titles
WHERE premiered +10 > 1980
```

```
SELECT primary_title, type
FROM titles
WHERE premiered > 1970
```

Practical Tips on Query Optimization

- Only retreive what you need - SELECT fields that are requested instead of SELECT *
- INNER JOIN vs. WHERE
  - Some DBMS systems may perform cartesian product when executing WHERE
- When exploring tables of huge amount of enteries, use LIMIT
- USE wildcards wisely
- Keep your queries simple
- Time complexity of a query plan
  - Constant time O(1)
  - Linear time O(n)
  - Logarithmic time O(nlog(n))
  - Quadratic time $O(n^2)$

```
SELECT TOP 1 titles.* FROM titles
```

```
SELECT title_id FROM titles
```

```
SELECT title_id FROM titles WHERE title_id = N
```

```
SELECT * FROM titles, ratings WHERE titles.title_id = ratings.title_id
```