Introduction to Data Management CSE 344

Lectures 10: System's Architecture and Relational Algebra

Announcements

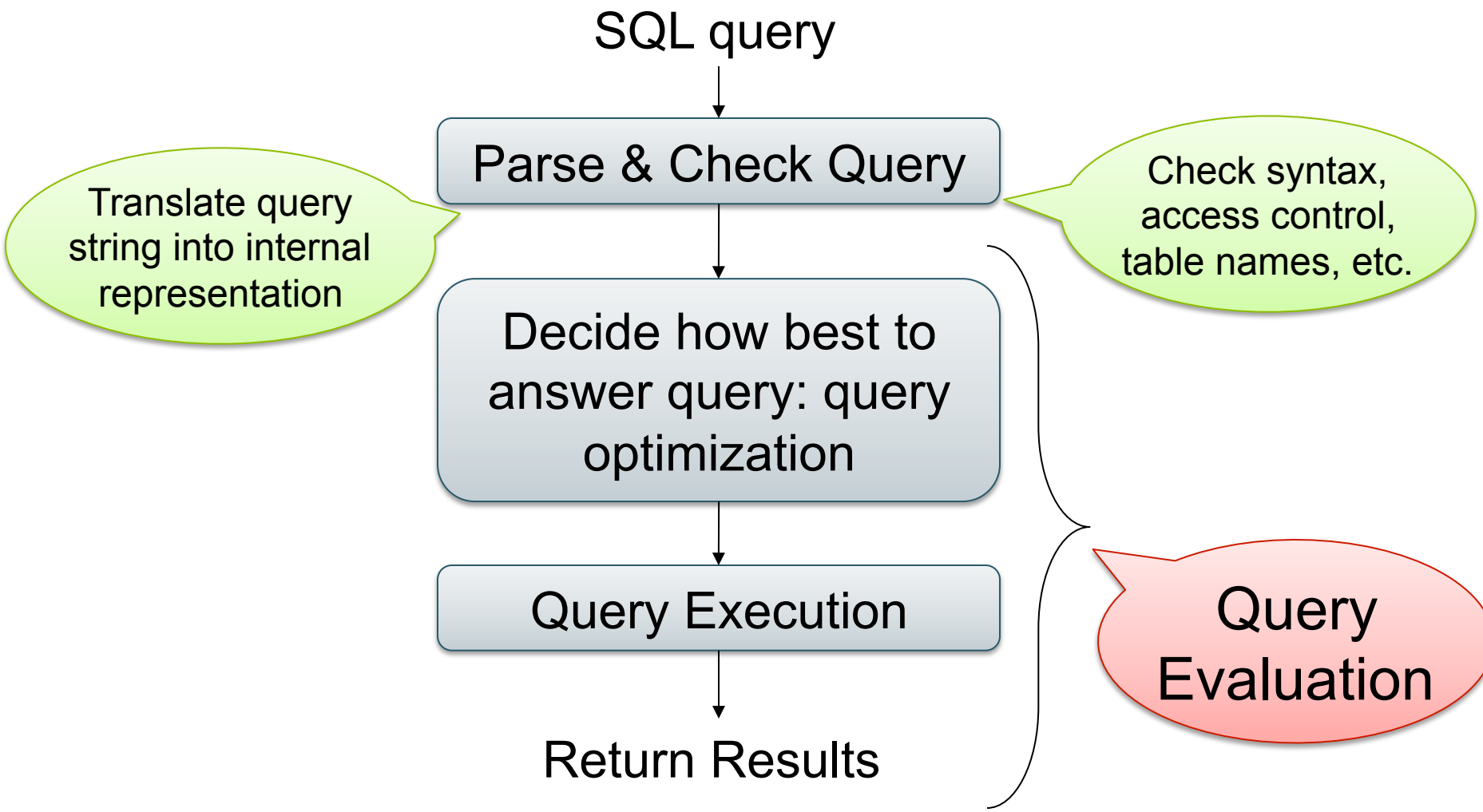
- Webquiz 3 due tonight!
- Today's lecture: 2.4 and 5.1

Where We Are

- Motivation for using a DBMS for managing data
- SQL, SQL, SQL
  - Declaring the schema for our data (CREATE TABLE)
  - Inserting data one row at a time or in bulk (INSERT/.import)
  - Modifying the schema and updating the data (ALTER/UPDATE)
  - Querying the data (SELECT)
  - Tuning queries (CREATE INDEX)

- Next step: More knowledge of how DBMSs work
  - Client-server architecture
  - Relational algebra and query execution

Query Evaluation Steps



The WHAT and the HOW

- SQL = WHAT we want to get form the data
- Relational Algebra = HOW to get the data we want
- The passage from WHAT to HOW is called query optimization

Overview: SQL = WHAT

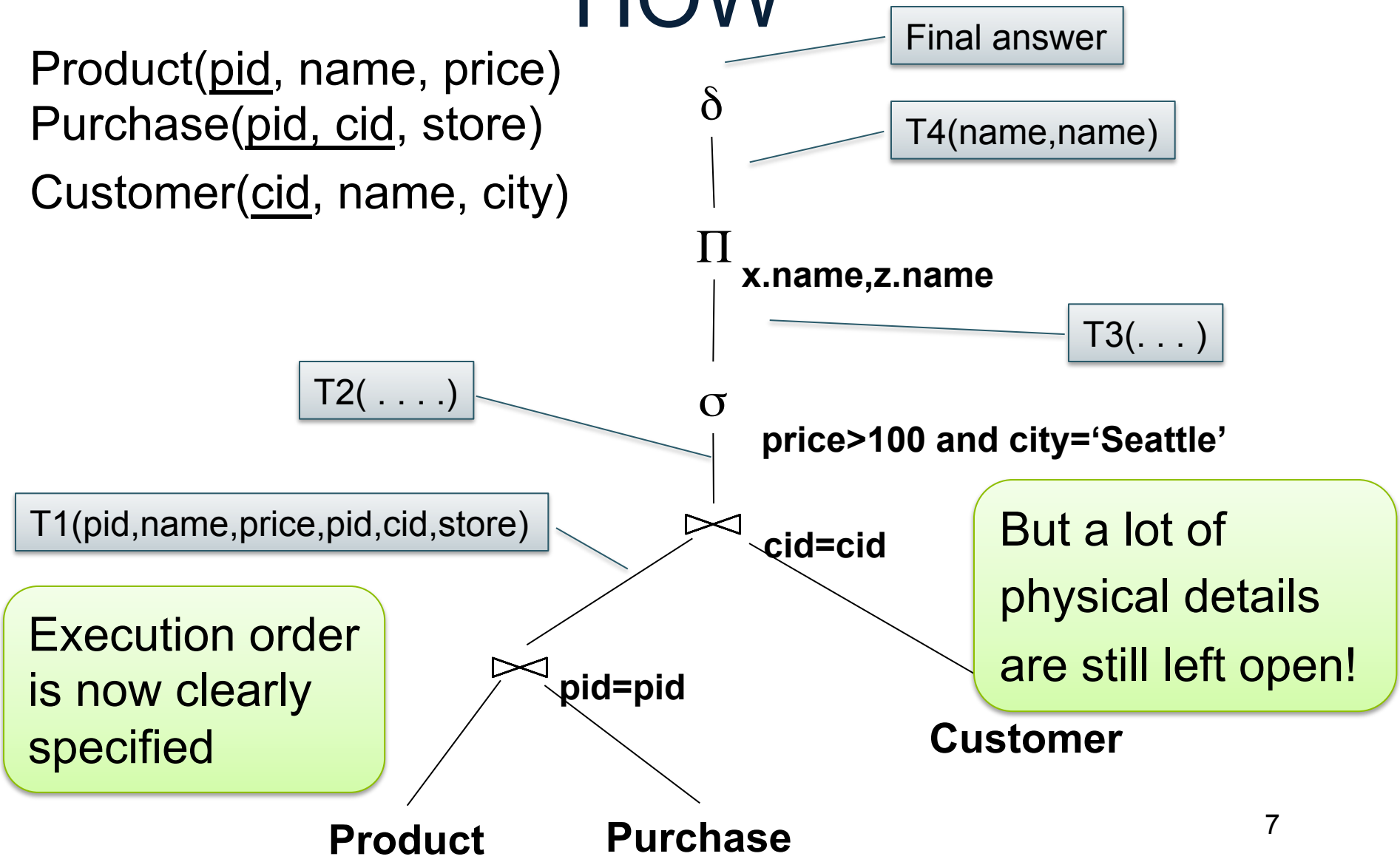
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = y.cid and
      x.price > 100 and z.city = 'Seattle'
```

It's clear WHAT we want, unclear HOW to get it

Overview: Relational Algebra = HOW

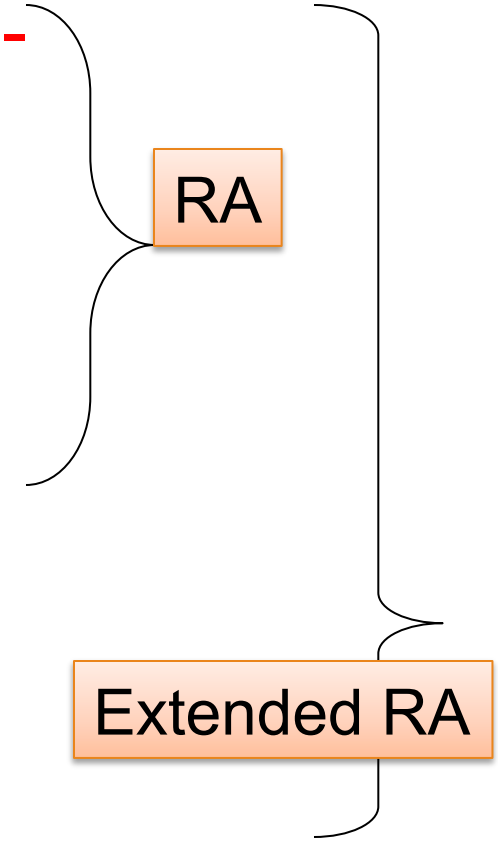
Product(<u>pid</u>, name, price)
Purchase(<u>pid, cid</u>, store)
Customer(<u>cid</u>, name, city)



Relational Algebra

Relational Algebra Operators

- Union ∪, intersection ∩, difference -
- Selection σ
- Projection Π
- Cartesian product ×, join ⋈
- Rename ρ
- Duplicate elimination δ
- Grouping and aggregation γ
- Sorting τ



From Logical Plans to Physical Plans

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Example

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and  y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

Give a relational algebra expression for this query

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Relational Algebra

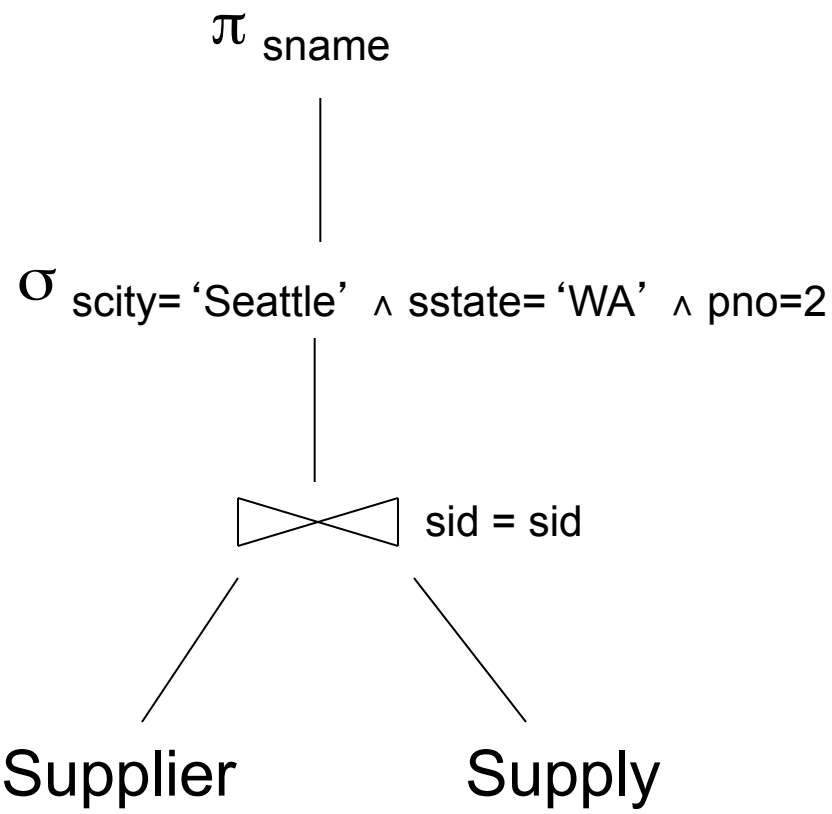
$$\pi_{\text{sname}}(\sigma_{\text{scity= 'Seattle'} \wedge \text{sstate= 'WA'} \wedge \text{pno=2}}(\text{Supplier} \bowtie_{\text{sid = sid}} \text{Supply}))$$

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Relational Algebra

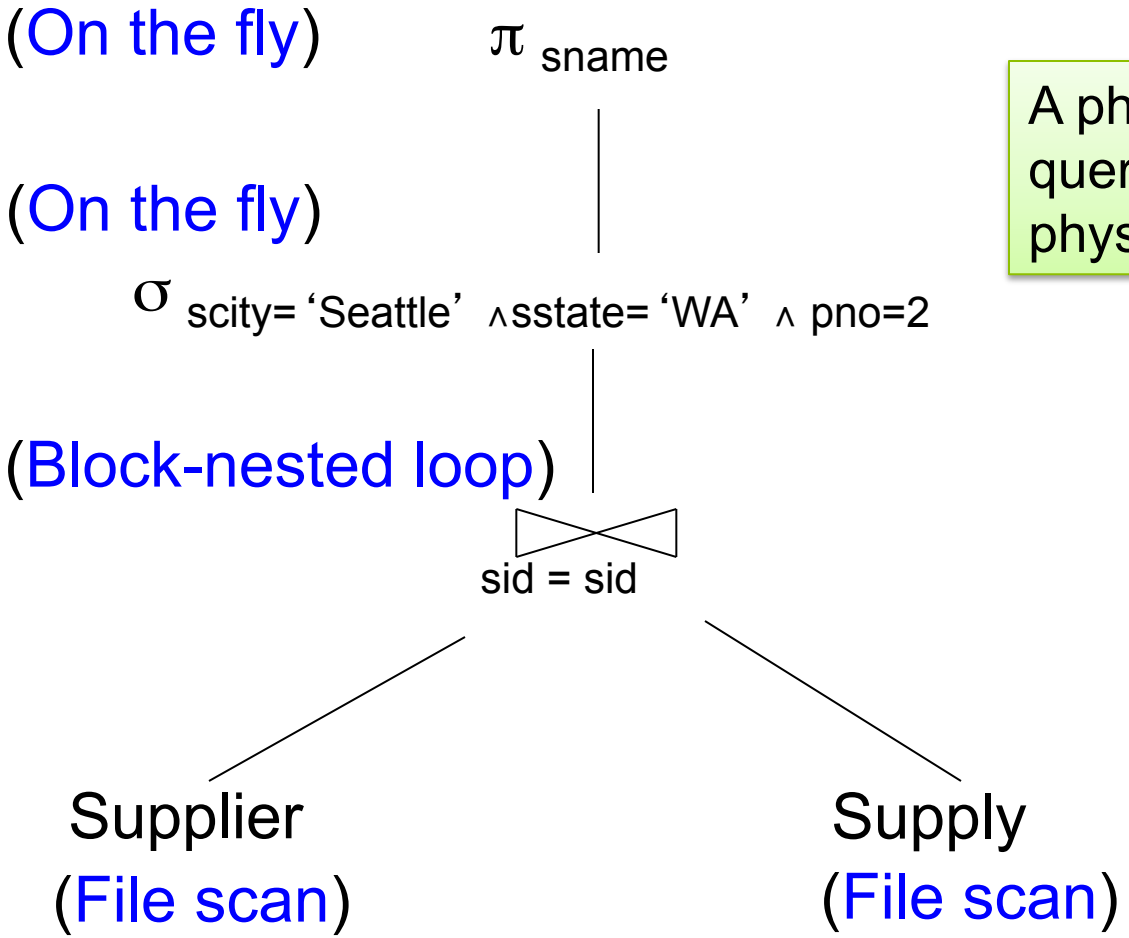
Relational algebra expression is also called the "logical query plan"



Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Physical Query Plan 1

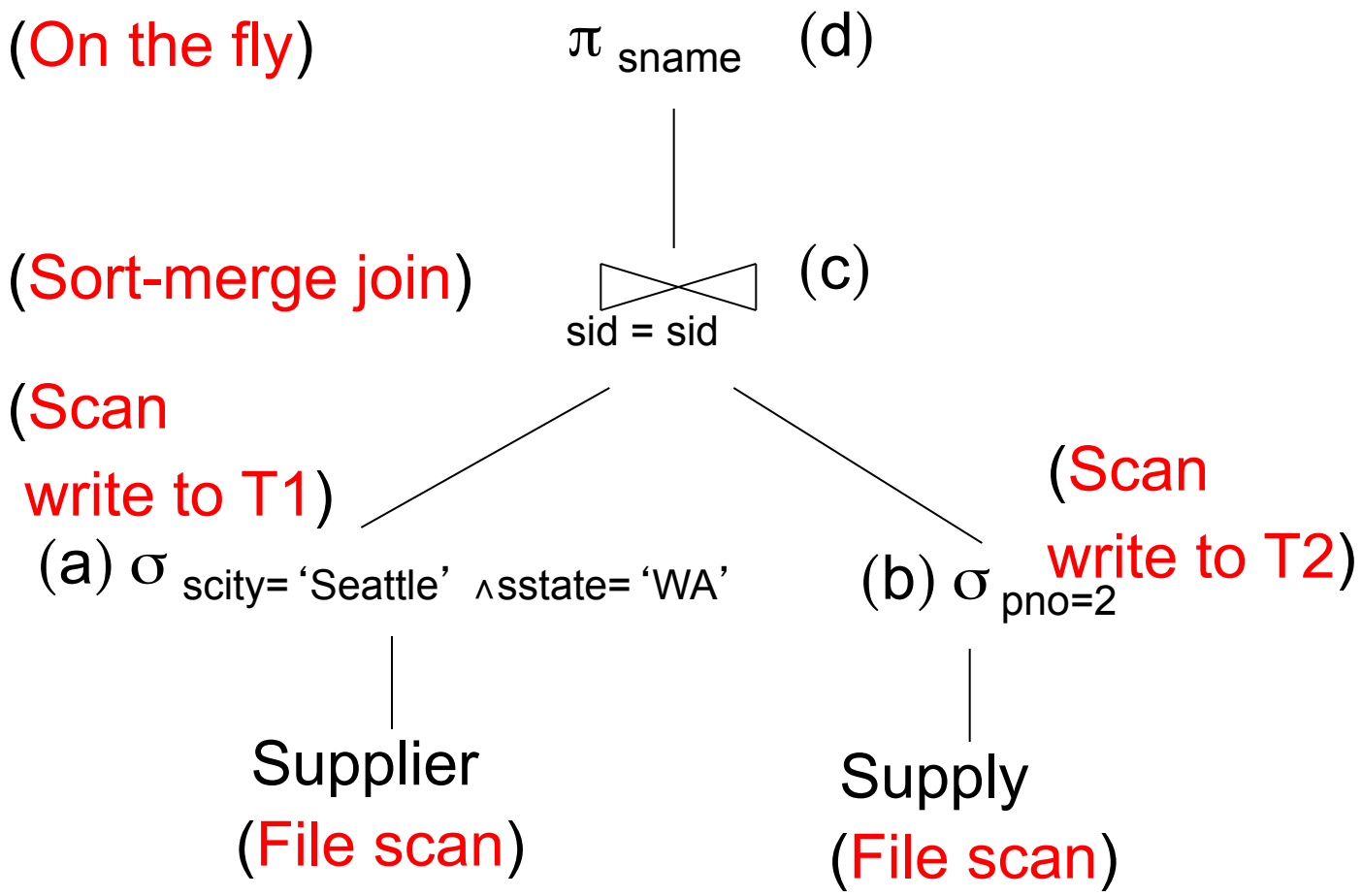


A physical query plan is a logical query plan annotated with physical implementation details

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

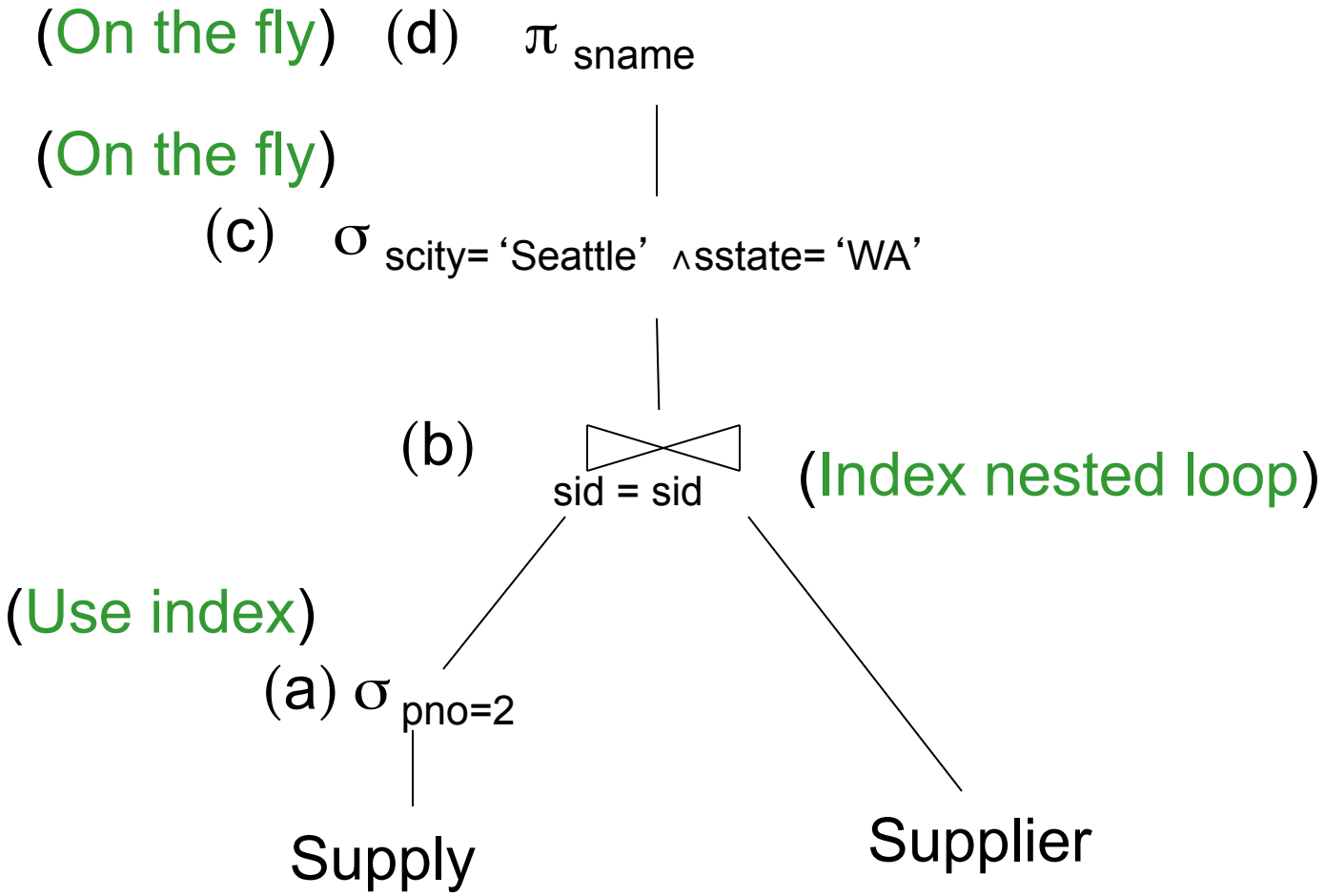
Physical Query Plan 2



Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Physical Query Plan 3



Physical Data Independence

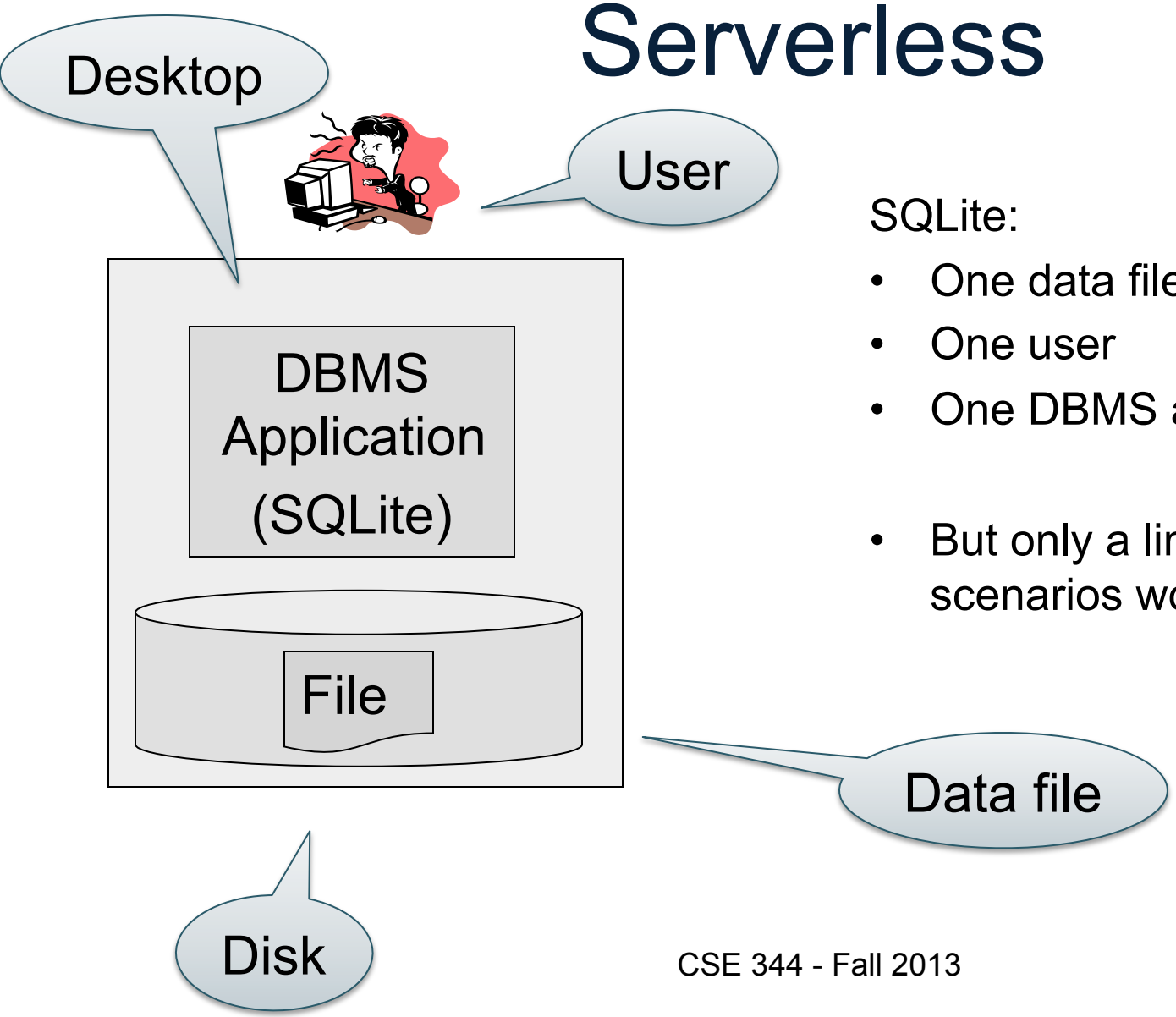
- Means that applications are insulated from changes in physical storage details
  - E.g., can add/remove indexes without changing apps
  - Can do other physical tunings for performance

- SQL and relational algebra facilitate physical data independence because both languages are "set-at-a-time": Relations as input and output

Architectures

1. Serverless
2. Two tier: client/server
3. Three tier: client/app-server/db-server

Serverless



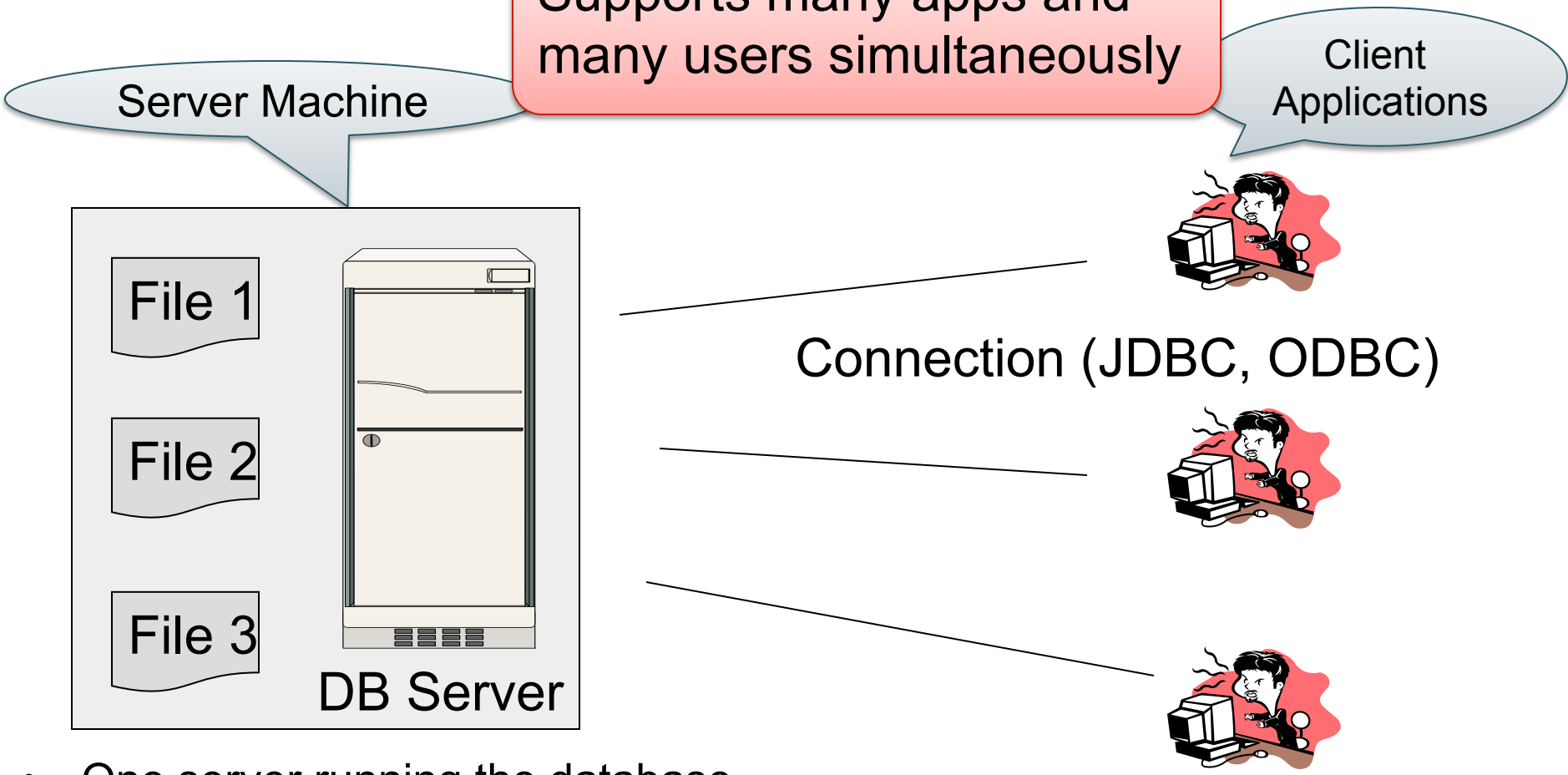
SQLite:

- One data file
- One user
- One DBMS application

- But only a limited number of scenarios work with such model

Client-Server

Supports many apps and many users simultaneously

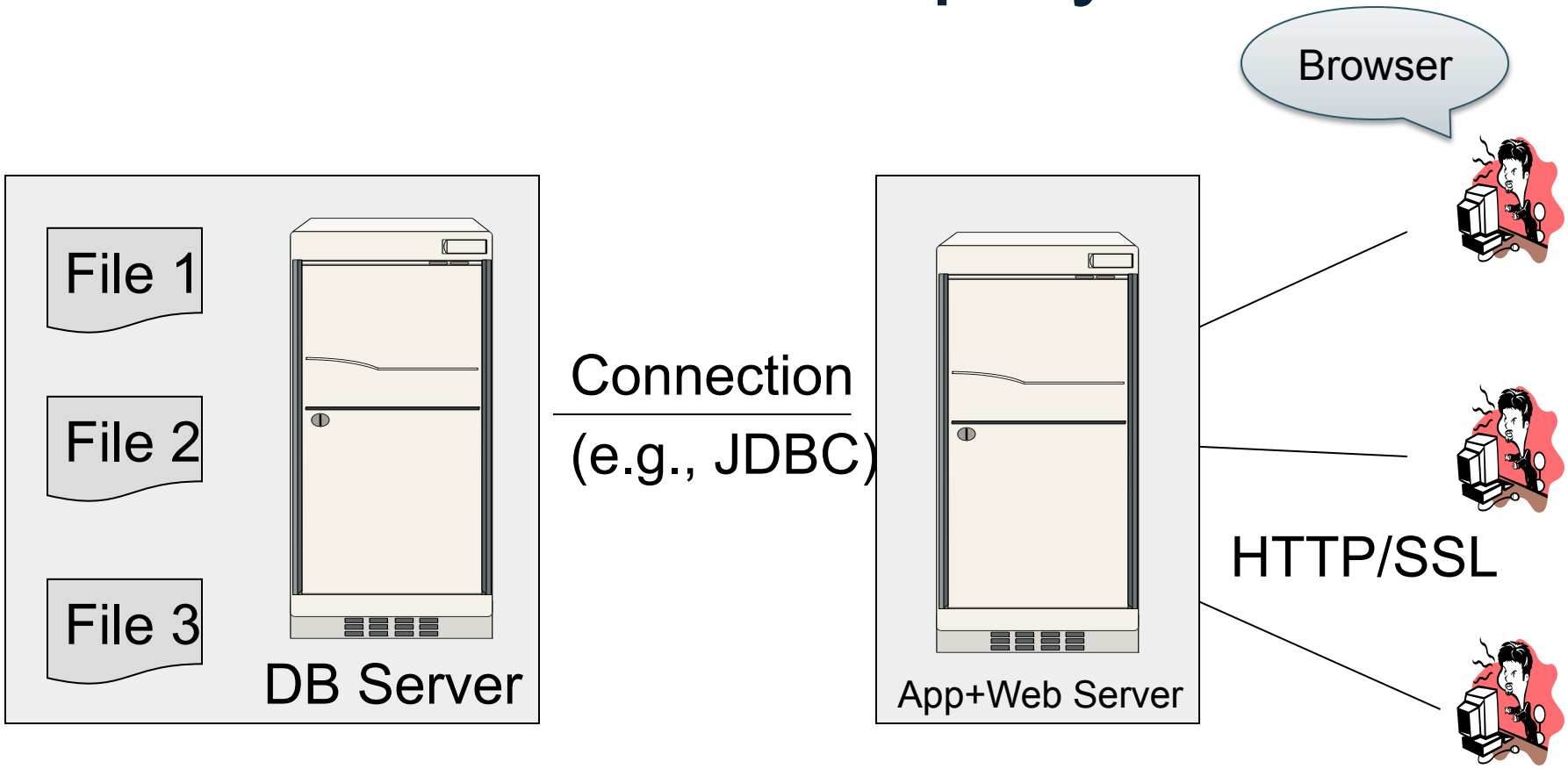


- One server running the database
- Many clients, connecting via the ODBC or JDBC (Java Database Connectivity) protocol

Client-Server

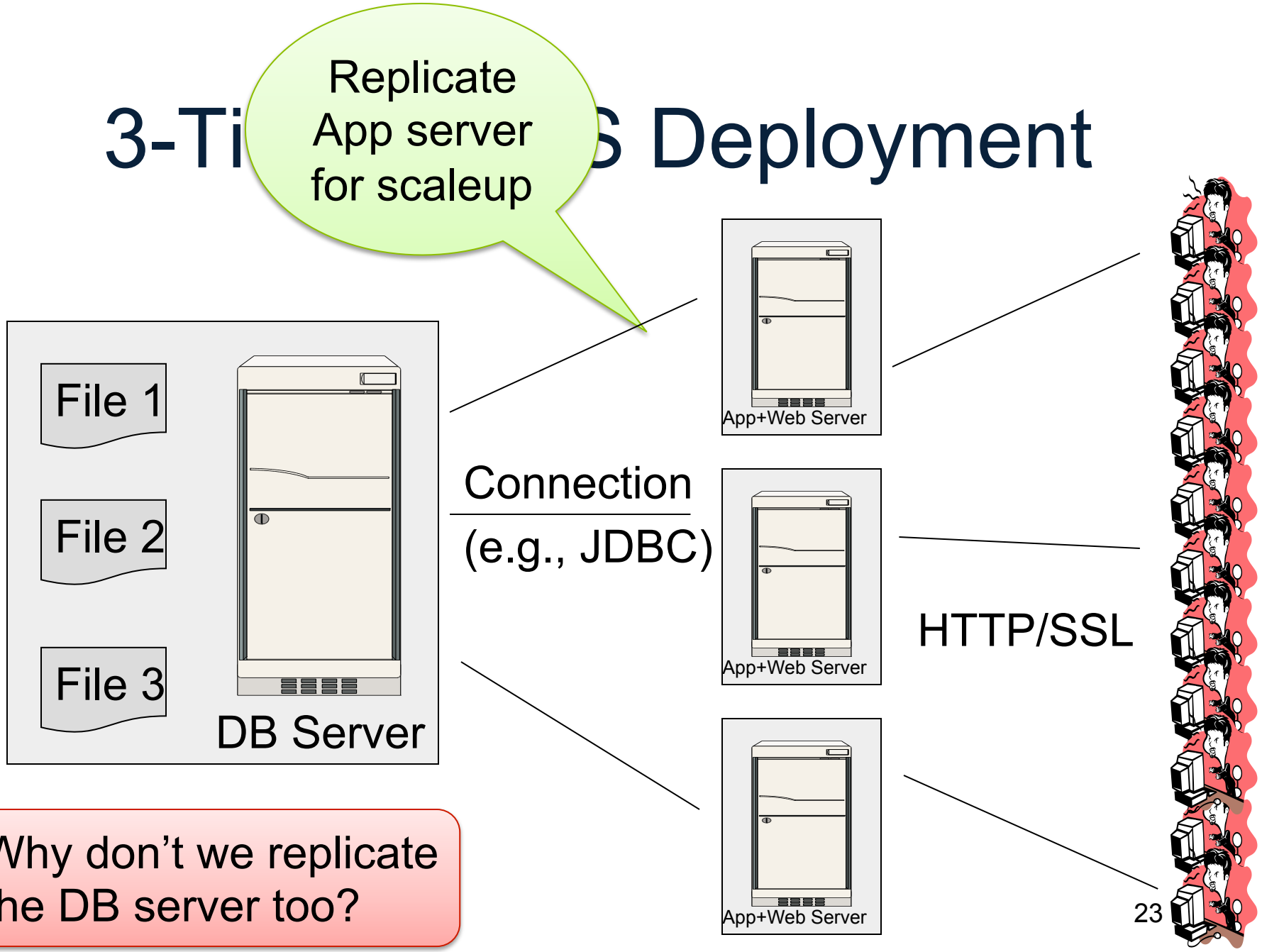
- One *server* that runs the DBMS (or RDBMS):
  - Your own desktop, or
  - Some beefy system, or
  - A cloud service (SQL Azure)
- Many *clients* run apps and connect to DBMS
  - Microsoft's Management Studio (for SQL Server), or
  - psql (for postgres)
  - Some Java program (HW5) or some C++ program
- Clients "talk" to server using JDBC/ODBC protocol

3-Tiers DBMS Deployment



Web-based applications

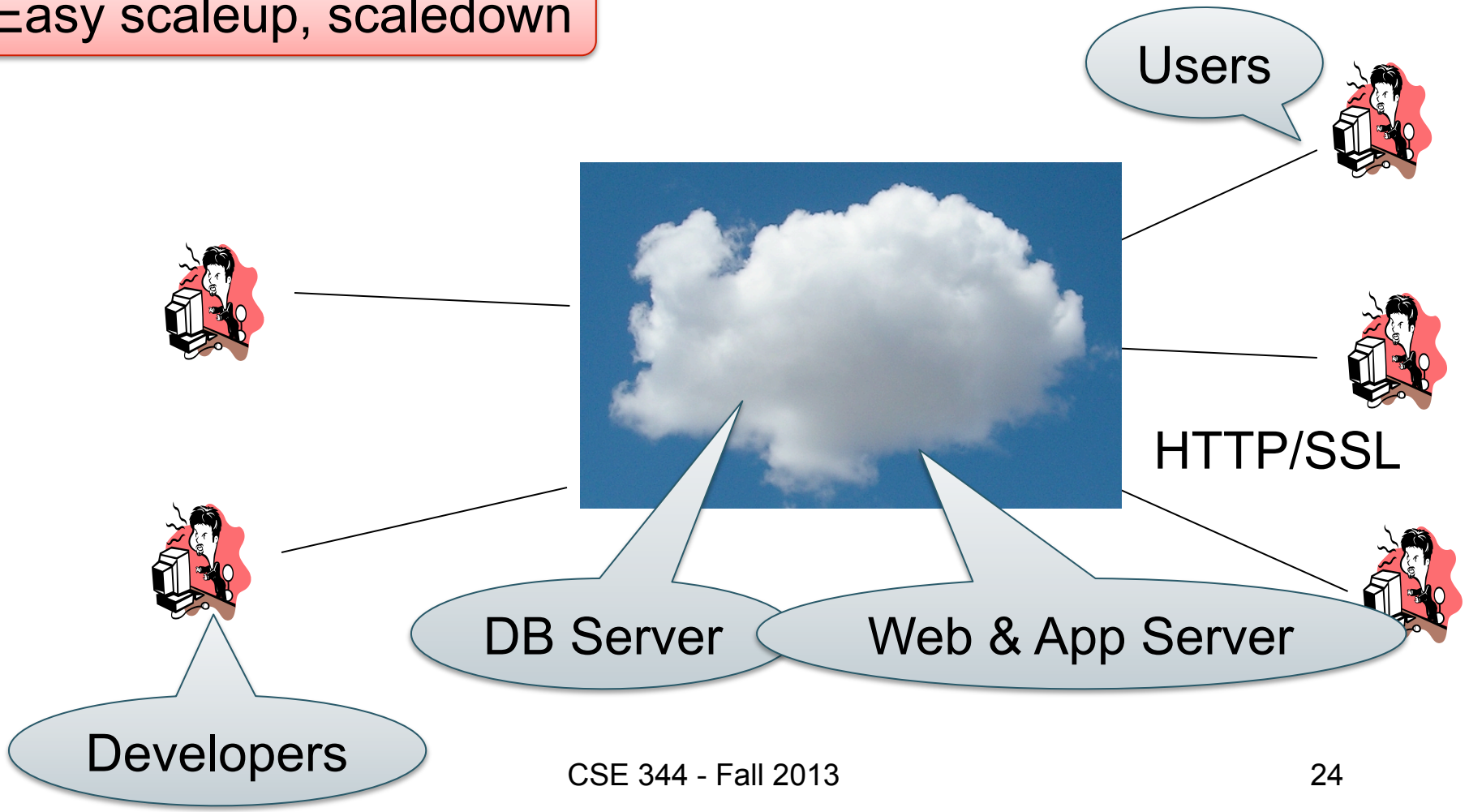
3-Ti S Deployment



Why don't we replicate the DB server too?

DBMS Deployment: Cloud

Easy scaleup, scaledown



Using a DBMS Server

1. Client application establishes connection to server
2. Client must authenticate self
3. Client submits SQL commands to server
4. Server executes commands and returns results