Introduction to Database Systems
CSE 414

Lecture 9: SQL Wrap-up and RDBMs Architecture

Announcements

- Webquiz due tonight, 11 pm
- Homework 3 is posted: due on Wednesday, 4/24
  - You need to attempt to log in to the Azure database now to discover any setup problems immediately. Use the discussion board (or mail to cse344-staff@cs if needed) to sort out logistics

Review: Indexes

V(M, N);

Suppose we have queries like these:

```
SELECT *
FROM V
WHERE M=?
```

```
SELECT *
FROM V
WHERE N=?
```

```
SELECT *
FROM V
WHERE M=? and N=?
```

Which of these indexes are helpful for each query?

1. Index on V(M)
2. Index on V(N)
3. Index on V(M,N)

Review: Indexes

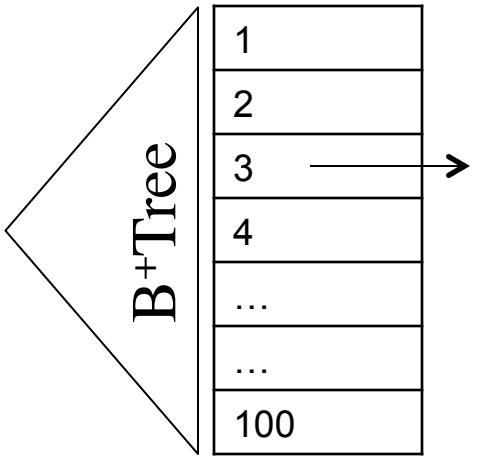
V(M, N);

Suppose V(M,N) contains 10,000 records:
(1,1), (1,2), …, (100, 100)

```
SELECT *
FROM V
WHERE M=3
```

```
SELECT *
FROM V
WHERE N=5
```

```
SELECT *
FROM V
WHERE M=3 and N=5
```



List of pointers to records (3,1), (3,2), …, (3,100)

Index on V(M)

Review: Indexes

V(M, N);

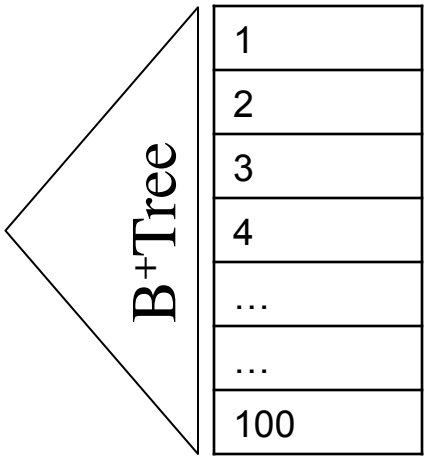
Suppose V(M,N) contains 10,000 records:
(1,1), (1,2), …, (100, 100)

```
SELECT *
FROM V
WHERE M=3
```

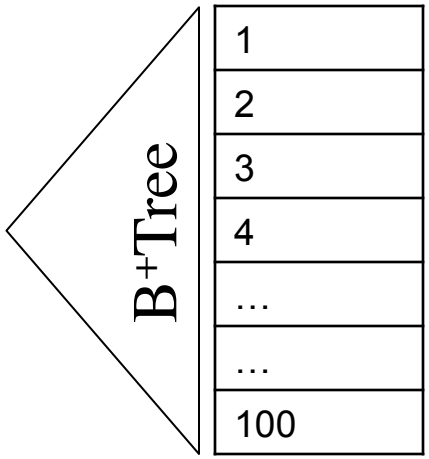
```
SELECT *
FROM V
WHERE N=5
```

```
SELECT *
FROM V
WHERE M=3 and N=5
```

How do we compute this query?



Index on V(M)



Index on V(N)

Review: Indexes

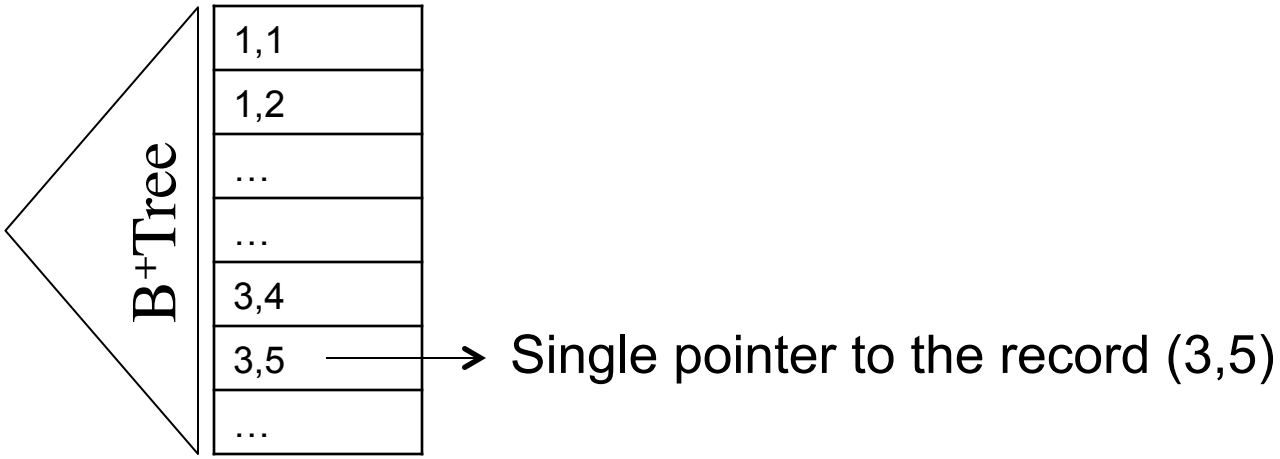
V(M, N);

Suppose V(M,N) contains 10,000 records:
(1,1), (1,2), …, (100, 100)

```
SELECT *
FROM V
WHERE M=3
```

```
SELECT *
FROM V
WHERE N=5
```

```
SELECT *
FROM V
WHERE M=3 and N=5
```



Index on V(M,N)

Review: Indexes

Discussion

- Why not create all three indexes V(M), V(N), V(M,N)?
- Suppose M is the primary key in V(<u>M</u>, N):
  V = {(1,1), (2,2), …, (10000, 10000)}
  How do the two indexes V(M) and V(M,N) compare? Consider their utility for evaluating the predicate M=5

Purchase(pid, product, quantity, price)

GROUP BY v.s. Nested Queries

```
SELECT      product, Sum(quantity) AS TotalSales
FROM        Purchase
WHERE       price > 1
GROUP BY    product
```

```
SELECT DISTINCT  x.product, (SELECT Sum(y.quantity)
                             FROM    Purchase y
                             WHERE x.product = y.product
                                   AND price > 1)
                            AS TotalSales
FROM        Purchase x
WHERE       price > 1
```

Why twice ?

Product (pname, price, cid)
Company(cid, cname, city)

Unnesting Aggregates

Find the number of companies in each city

```
SELECT DISTINCT city, (SELECT count(*)
                       FROM Company Y
                       WHERE X.city = Y.city)
FROM  Company X
```

```
SELECT city,  count(*)
FROM   Company
GROUP BY city
```

Equivalent queries

Note: no need for DISTINCT
(DISTINCT *is the same* as GROUP BY)

Product (pname, price, cid)
Company(cid, cname, city)

Unnesting Aggregates

What if there are no products for a city?

Find the number of products made in each city

```
SELECT DISTINCT X.city, (SELECT count(*)
                         FROM Product Y, Company Z
                         WHERE Z.cid=Y.cid
                         AND Z.city = X.city)
FROM  Company X
```

```
SELECT X.city, count(*)
FROM Company X, Product Y
WHERE X.cid=Y.cid
GROUP BY X.city
```

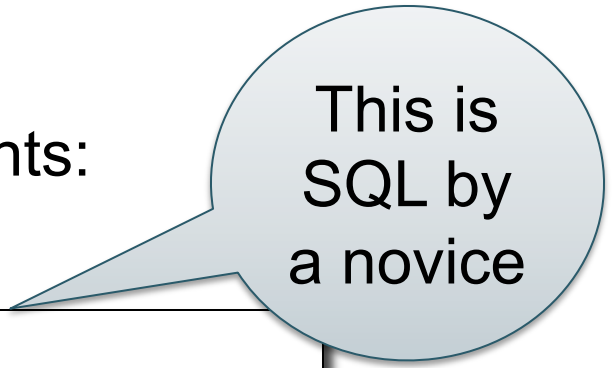
They are NOT equivalent ! (WHY?)

More Unnesting

Author(login,name)

Wrote(login,url)

- Find authors who wrote ≥ 10 documents:
- Attempt 1: with nested queries




```
SELECT DISTINCT Author.name
FROM        Author
WHERE       (SELECT count(Wrote.url)
             FROM Wrote
             WHERE Author.login=Wrote.login)
                > 10
```

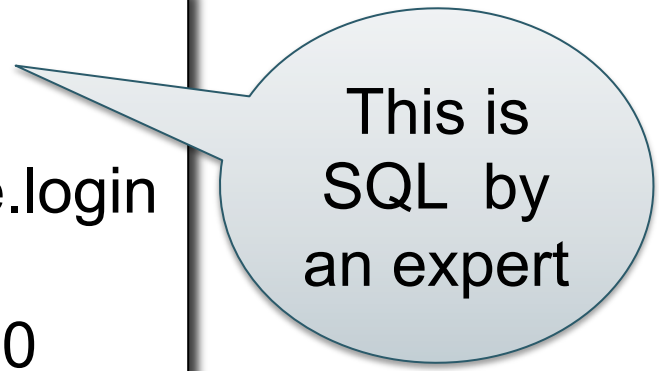
Author(login,name)

Wrote(login,url)

More Unnesting

- Find all authors who wrote at least 10 documents:
- Attempt 2: SQL style (with GROUP BY)

```
SELECT    Author.name
FROM      Author, Wrote
WHERE     Author.login=Wrote.login
GROUP BY Author.name
HAVING    count(wrote.url) > 10
```



Product (pname, price, cid)
Company(cid, cname, city)

Finding Witnesses

For each city, find the most expensive product made in that city

Product (pname, price, cid)
Company(cid, cname, city)

Finding Witnesses

For each city, find the most expensive product made in that city

Finding the maximum price is easy…

```
SELECT x.city, max(y.price)
FROM Company x, Product y
WHERE x.cid = y.cid
GROUP BY x.city;
```

But we need the *witnesses*, i.e. the products with max price

Product (pname, price, cid)
Company(cid, cname, city)

Finding Witnesses

To find the witnesses, compute the maximum price in a subquery

```
SELECT DISTINCT u.city, v.pname, v.price
FROM Company u, Product v,
     (SELECT x.city, max(y.price) as maxprice
      FROM Company x, Product y
      WHERE x.cid = y.cid
      GROUP BY x.city) w
WHERE u.cid = v.cid
       and u.city = w.city
       and v.price=w.maxprice;
```

Product (pname, price, cid)
Company(cid, cname, city)

Finding Witnesses

There is a more concise solution here:

```
SELECT  u.city, v.pname, v.price
FROM    Company u, Product v, Company x, Product y
WHERE u.cid = v.cid and u.city = x.city and x.cid = y.cid
GROUP BY u.city, v.pname, v.price
HAVING     v.price = max(y.price);
```

Product (pname, price, cid)
Company(cid, cname, city)

Finding Witnesses

And another one:

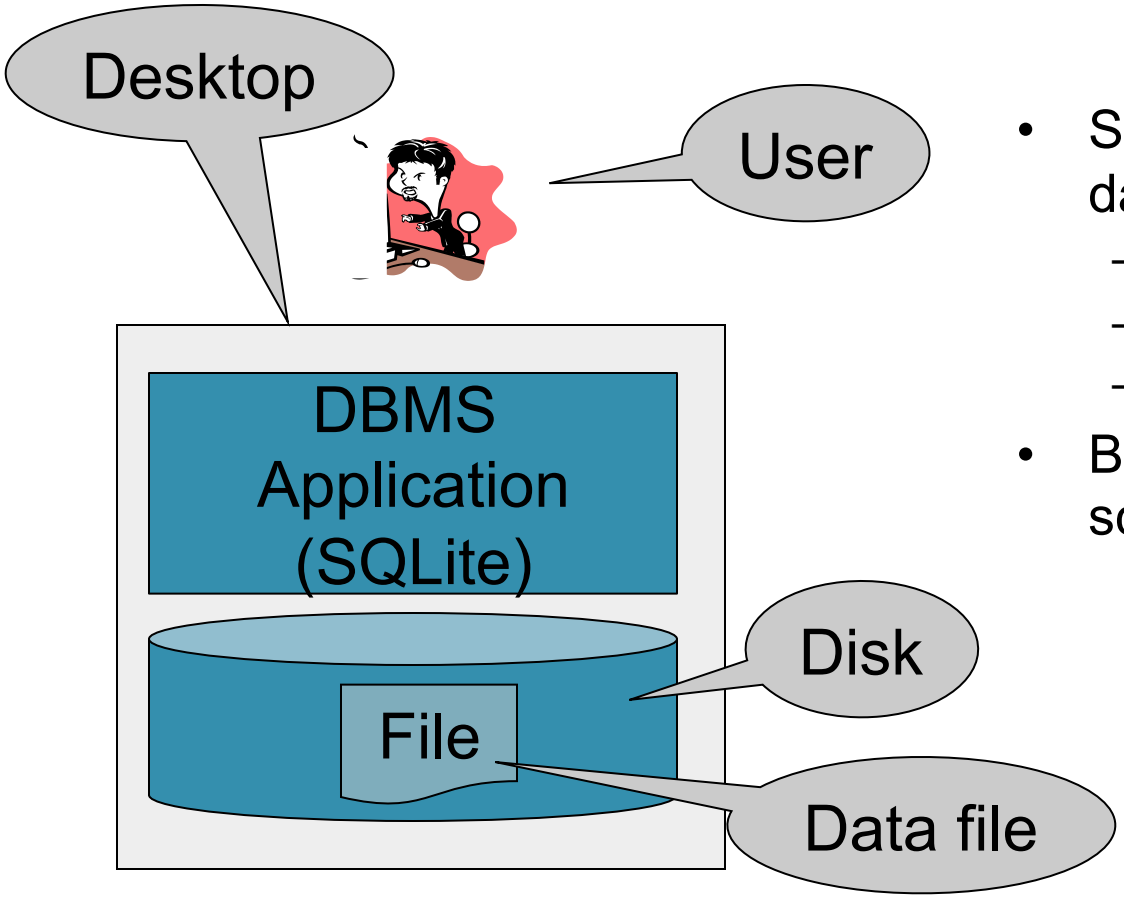
```
SELECT u.city, v.pname, v.price
FROM Company u, Product v
WHERE u.cid = v.cid
  and v.price >= ALL (SELECT y.price
                      FROM Company x, Product y
                      WHERE u.city=x.city
                            and x.cid=y.cid);
```

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)

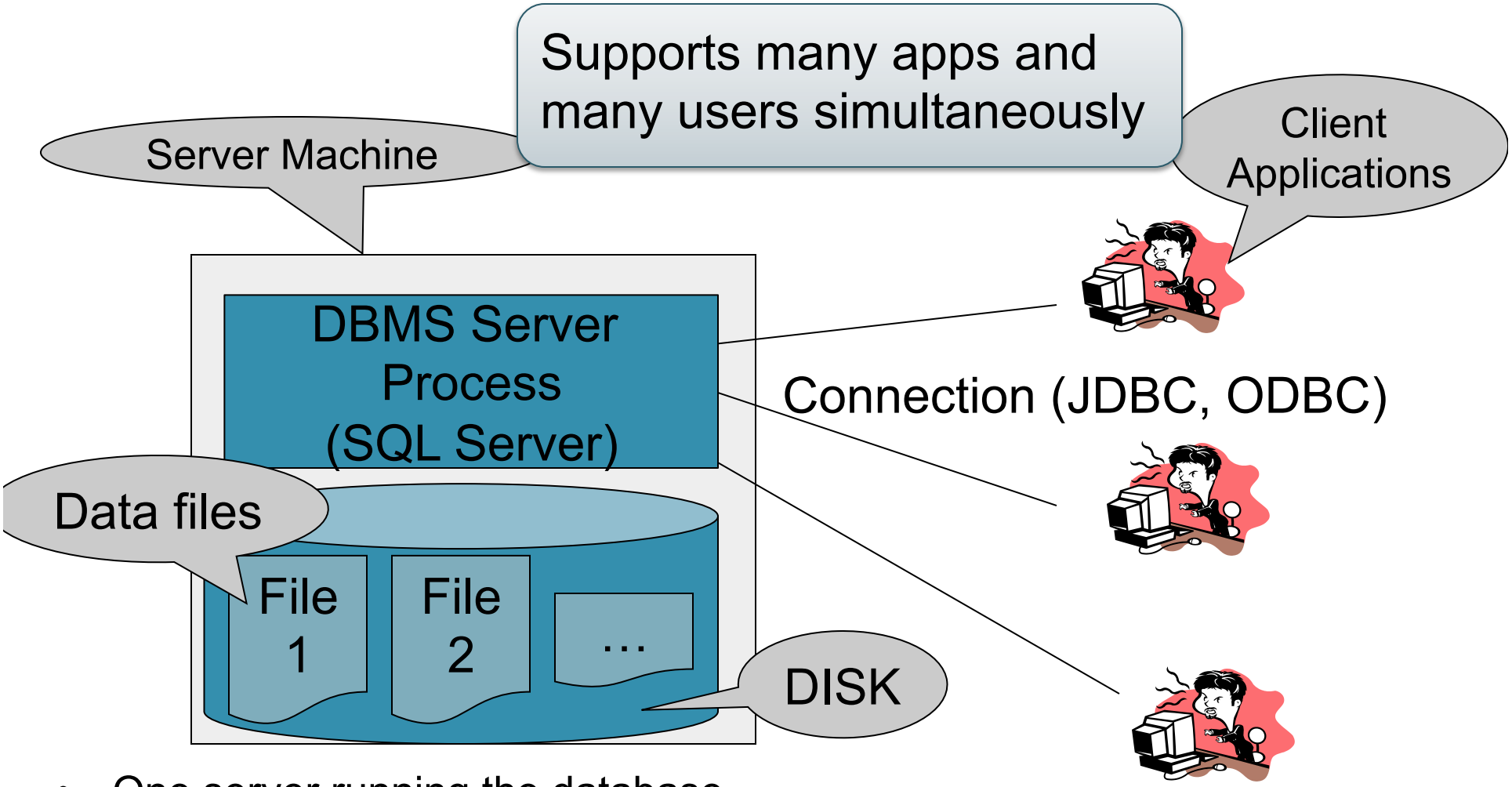
- Next step: More knowledge of how DBMSs work
  - Client-server architecture
  - Relational algebra and query execution

Data Management with SQLite



- So far, we have been managing data with SQLite as follows:
  - One data file
  - One user
  - One DBMS application
- But only a limited number of scenarios work with such model

Client-Server Architecture

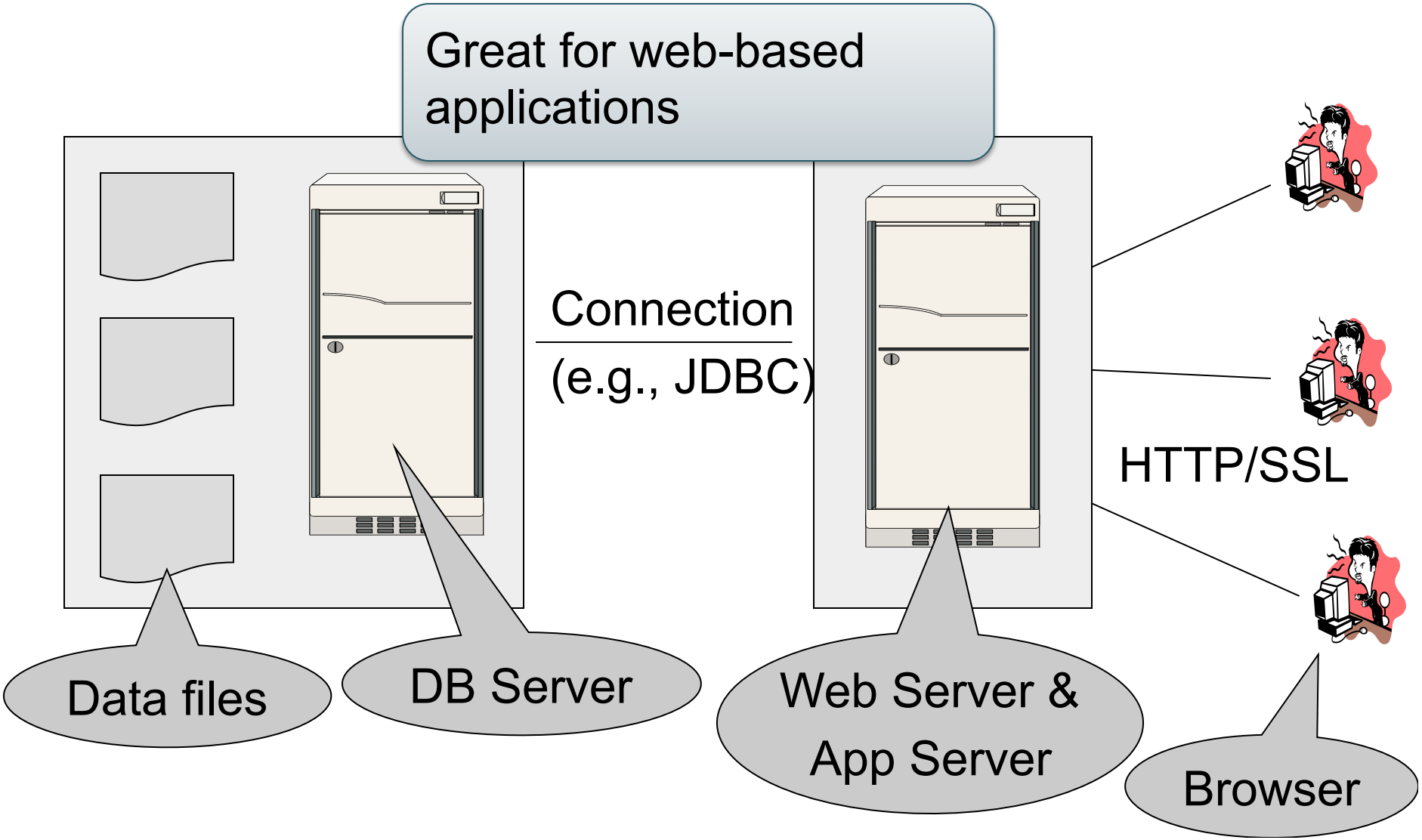


- One server running the database
- Many clients, connecting via the ODBC or JDBC (Java Database Connectivity) protocol

Client-Server Architecture

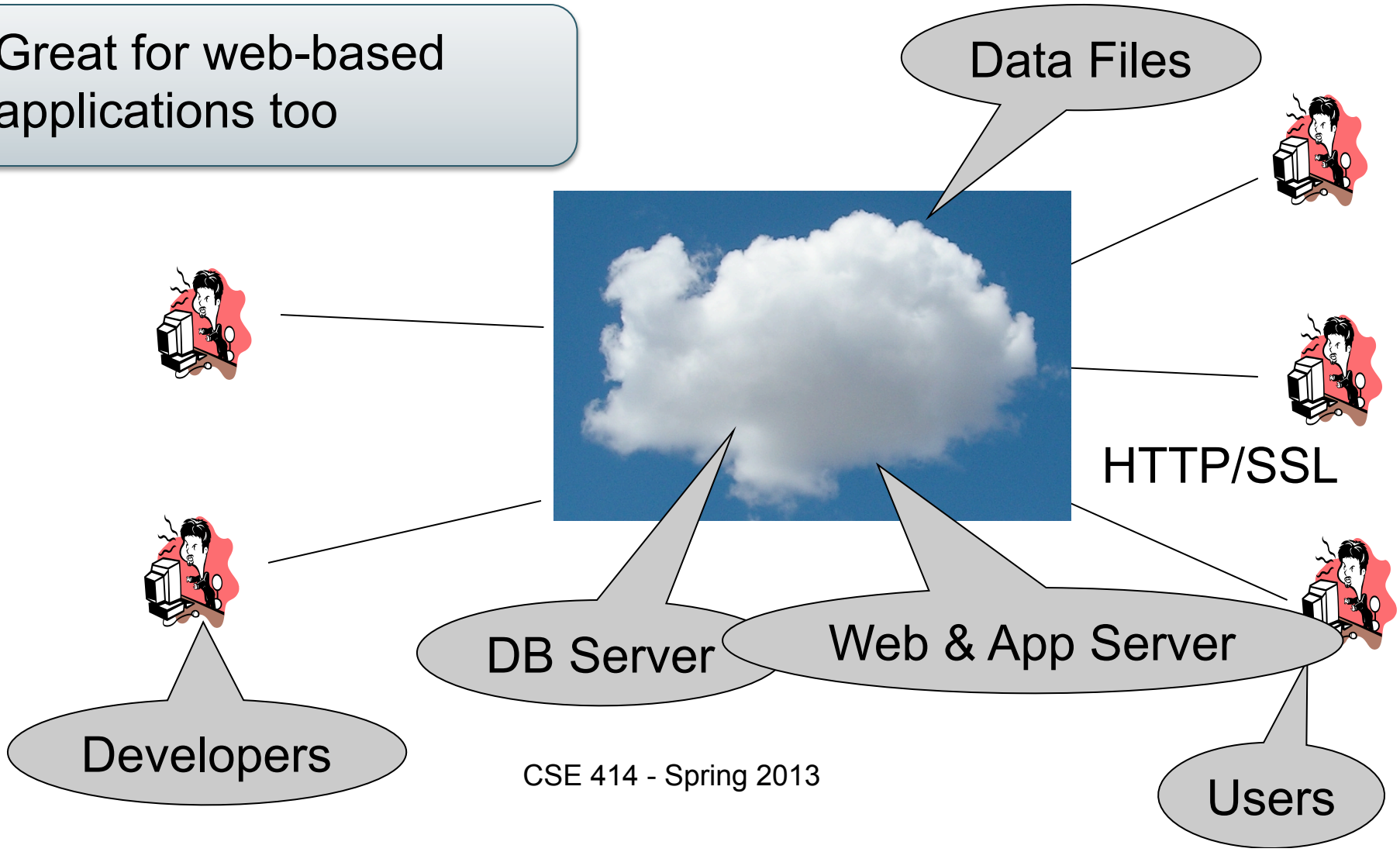
- One *server* that runs the DBMS (or RDBMS):
  - Your own desktop, or
  - Some beefy system, or
  - A cloud service (SQL Azure)
- Many *clients* run apps and connect to DBMS
  - Microsoft's Management Studio (for SQL Server), or
  - psql (for postgres)
  - Some Java program (HW5) or some C++ program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

DBMS Deployment: 3 Tiers



DBMS Deployment: Cloud

Great for web-based applications too



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