Introduction to Data Management
CSE 344

Lecture 8:
SQL Wrap-up
Systems Architecture
Announcements

• Homework 3 will be out tonight
  – Due next Thursday
  – Play with SQL Azure!

• Webquiz 3 out
  – Due next Tuesday
Review: Indexes

\[ V(M, N); \]

Suppose we have queries like these:

\[
\begin{align*}
\text{SELECT} & \quad \text{*} \\
& \quad \text{FROM} \ V \\
& \quad \text{WHERE} \ M=？
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{*} \\
& \quad \text{FROM} \ V \\
& \quad \text{WHERE} \ N=？
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{*} \\
& \quad \text{FROM} \ V \\
& \quad \text{WHERE} \ M=？ \text{ and } N=？
\end{align*}
\]

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

Suppose V(M,N) contains 10,000 records:
(1,1), (1,2), (1,3), ..., (1,100), (2,1), ..., (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)
Suppose V(M,N) contains 10,000 records: (1,1), (1,2), (1,3), ..., (1,100), (2,1),..., (100, 100)

The index is useful here

List of pointers to records (3,1), (3,2), ..., (3,100)
Review: Indexes

Suppose $V(M,N)$ contains 10,000 records: 
(1,1), (1,2), (1,3), ..., (1,100), (2,1), ..., (100, 100)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

Index on $V(M)$

The index is useful here

Useless here

List of pointers to records (3,1), (3,2), ..., (3,100)
Review: Indexes

Suppose V(M,N) contains 10,000 records: (1,1), (1,2), (1,3), ..., (1,100), (2,1), ..., (100, 100)

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

The index is useful here
Useless here
Can we use it here?

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

Index on V(M)
Review: Indexes

Suppose \( V(M,N) \) contains 10,000 records: 
(1,1), (1,2), (1,3), ..., (1,100), (2,1), ..., (100, 100)

- Select * from \( V \) where \( M=3 \)
- Select * from \( V \) where \( N=5 \)
- Select * from \( V \) where \( M=3 \) and \( N=5 \)

Where does this index help?

Index on \( V(M) \)

Index on \( V(N) \)
Review: Indexes

Suppose \( V(M,N) \) contains 10,000 records:
(1,1), (1,2), (1,3), …, (1,100), (2,1),…, (100, 100)

- **Index on \( V(M) \)**
- **Index on \( V(N) \)**
- **Index on \( V(M,N) \)**

**SELECT * FROM \( V \) WHERE \( M=3 \)**

**SELECT * FROM \( V \) WHERE \( N=5 \)**

**SELECT * FROM \( V \) WHERE \( M=3 \) and \( N=5 \)**

And this?

Single pointer to record (3,5)
Review: Clustered vs Unclustered

Every table can have **only one** clustered and **many** unclustered indexes
Review: Index Classification

- **Clustered/unclustered**
  - Clustered = records close in index are close in data
    - Option 1: Data inside data file is sorted on disk
    - Option 2: Store data directly inside the index (no separate files)
  - Unclustered = records close in index may be far in data
Review: Indexes

Suppose M is the primary key in V(M, N):
Review: Indexes

Suppose M is the primary key in V(M, N):

How do the two indexes V(M) and V(M,N) compare?

Consider their utility for these predicates:
• M=5
• M=5 and N=7
Nested Queries

• Subqueries can occur in every clause:
  – SELECT
  – FROM
  – WHERE

• When we must use nested subqueries:
  – Non-monotone queries
  – Queries making complex use of aggregates
Practice these queries in SQL

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serves some beer they like.

Find drinkers that frequent some bar that serves only beers they like.

Find drinkers that frequent only bars that serves only beer they like.
Unnesting Aggregates

Find the number of companies in each city

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

Equivalent queries

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

Find the number of companies in each city

\[
\begin{align*}
\text{SELECT} & \quad \text{DISTINCT} \quad \text{X.city}, \ (\text{SELECT} \ \text{count}(*) \\
& \quad \text{FROM} \ \text{Company} \ Y \\
& \quad \text{WHERE} \ \text{X.city} = \ Y\text{.city}) \\
\text{FROM} & \quad \text{Company} \ X
\end{align*}
\]

Equivalent queries

\[
\begin{align*}
\text{SELECT} & \quad \text{city}, \ \text{count}(*) \\
& \quad \text{FROM} \ \text{Company} \\
& \quad \text{GROUP BY} \ \text{city}
\end{align*}
\]

Note: no need for \text{DISTINCT} (\text{DISTINCT} \ is \ the \ same \ as \ \text{GROUP BY})
Product (pname, price, cid)
Company(cid, cname, city)

Unnesting Aggregates

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
Find the number of products made in each city

\[
\text{SELECT DISTINCT } X.\text{city}, (\text{SELECT count(*) FROM Product Y, Company Z WHERE Z.cid=Y.cid AND Z.city = X.city}) \]

\[
\text{FROM Company X} \]

\[
\text{SELECT X.\text{city}, count(*) FROM Company X, Product Y WHERE X.cid=Y.cid GROUP BY X.\text{city}} \]

NOT equivalent! You should know why!
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 y.price > 1)
AS TotalSales
FROM Purchase x
WHERE x.price > 1
```

Why twice?
Author(login,name)
Wrote(login,url)

More Unnesting

Find authors who wrote $\geq$ 10 documents:
More Unnesting

Find authors who wrote $\geq 10$ 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
```

This is SQL by a novice
More Unnesting

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
```
More Unnesting

Find authors who wrote $\geq$ 10 documents:

Attempt 1: with nested queries

Attempt 2: using GROUP BY and HAVING

```
SELECT Author.name
FROM Author, Wrote
WHERE Author.login=Wrote.login
GROUP BY Author.name
HAVING count(wrote.url) $\geq$ 10
```
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)
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
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
SQLite:
• One data file
• One user
• One DBMS application

• But only a limited number of scenarios work with such model
SQLite:

- One data file
- One user
- One DBMS application

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

Client Applications
Client-Server

Connection (JDBC, ODBC)

Client Applications
Client-Server

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

Supports many apps and many users simultaneously

Server Machine

File 1

File 2

File 3

DB Server

Connection (JDBC, ODBC)

Client Applications

• 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)
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
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
3-Tiers DBMS Deployment

DB Server

File 1
File 2
File 3

Connection (e.g., JDBC)

App+Web Server

HTTP/SSL

Browser
3-Tiers DBMS Deployment

Connection (e.g., JDBC)

Browser

HTTP/SSL

Web-based applications
3-Tiers DBMS Deployment

File 1
File 2
File 3

DB Server

Connection
(e.g., JDBC)

App+Web Server

HTTP/SSL

CSE 344 - Winter 2015
3-Tiers DBMS Deployment

Why don’t we replicate the DB server too?
Why don’t we replicate the DB server too?

Replicate App server for scaleup

Connection (e.g., JDBC)

HTTP/SSL
DBMS Deployment: Cloud

Easy scaleup, scaledown

Users

HTTP/SSL

Developers

DB Server

Web & App Server
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