CS 764: Topics in Database Management Systems
Lecture 1: Introduction

Xiangyao Yu
9/8/2021
Who am I?

Name: Xiangyao Yu

Assistant professor in computer sciences, database group

Research interests:

• Transactions and HTAP
• New hardware for databases
• Cloud-native databases
Today’s Agenda

Database 101

Course logistics
Database 101

**Database**: A collection of data, typically describing the activities of one or more related organizations. For example:

- Entities: students, instructors, courses
- Relationships: students enroll in courses, instructors teach courses
Database 101

**Database**: A collection of data, typically describing the activities of one or more related organizations. For example:
- Entities: students, instructors, courses
- Relationships: students enroll in courses, instructors teach courses

**Database management system (DBMS)**: Software designed to assist in *maintaining* and *utilizing large collection* of data.
Relational Model

A database is a collection of one or more relations, where each relation is a table with rows and columns.

An example relation:

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad</td>
<td>tablet</td>
<td>$399.00</td>
<td>Apple</td>
</tr>
<tr>
<td>Surface</td>
<td>tablet</td>
<td>$299.00</td>
<td>Microsoft</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Relational Model

A database is a **collection of one or more relations**, where each relation is a **table with rows and columns**.

An example relation:

<table>
<thead>
<tr>
<th>Product</th>
<th>name</th>
<th>category</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad</td>
<td>tablet</td>
<td></td>
<td>$399.00</td>
<td>Apple</td>
</tr>
<tr>
<td>Surface</td>
<td>tablet</td>
<td></td>
<td>$299.00</td>
<td>Microsoft</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

record/tuple/row
A database is a **collection of one or more relations**, where each relation is a **table with rows and columns**.

An example relation:

<table>
<thead>
<tr>
<th>Product</th>
<th>name</th>
<th>category</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iPad</td>
<td>tablet</td>
<td>$399.00</td>
<td>Apple</td>
</tr>
<tr>
<td>Surface</td>
<td>Surface</td>
<td>tablet</td>
<td>$299.00</td>
<td>Microsoft</td>
</tr>
<tr>
<td></td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
SQL Queries

SELECT a_1, a_2, ..., a_k
FROM R_1, R_2, ..., R_n
WHERE conditions
A Database Template

SELECT a₁, a₂, ..., aₖ
FROM R₁, R₂, ..., Rₙ
WHERE conditions

answer = {}

for t₁ in R₁ do
  for t₂ in R₂ do
    ...
    for tₙ in Rₙ do
      if conditions
        then answer = answer U {(a₁, ..., aₖ)}

return answer

Vanilla query executor
A Database Template

```
SELECT  a_1, a_2, ..., a_k
FROM    R_1, R_2, ..., R_n
WHERE   conditions
```

```
answer = {}
for t_1 in R_1 do
    for t_2 in R_2 do
        ...
        for t_n in R_n do
            if conditions
                then answer = answer ∪ {(a_1,...,a_k)}
return answer
```

Vanilla query executor

Database

Transactions

Lock(DB)

Execute transaction

Unlock(DB)

A DBMS can be heavily optimized beneath this simple interface
SELECT a_1, a_2, ..., a_k
FROM R_1, R_2, ..., R_n
WHERE conditions

```
answer = {}
for t_1 in R_1 do
    for t_2 in R_2 do
        ...
        for t_n in R_n do
            if conditions
                then answer = answer \cup \{(a_1, ..., a_k)\}
return answer
```

Vanilla query executor

Cross products are expensive, can replace with joins
Optimizing the Template Implementation

\[
\text{SELECT } a_1, a_2, \ldots, a_k \\
\text{FROM } R_1, R_2, \ldots, R_n \\
\text{WHERE } \text{conditions}
\]

\[
\text{answer} = \{} \\
\text{for } t_1 \text{ in } R_1 \text{ do} \\
\quad \text{for } t_2 \text{ in } R_2 \text{ do} \\
\quad \quad \ldots \\
\quad \quad \text{for } t_n \text{ in } R_n \text{ do} \\
\quad \quad \quad \text{if } \text{conditions} \\
\quad \quad \quad \quad \text{then } \text{answer} = \text{answer} \cup \{(a_1, \ldots, a_k)\} \\
\text{return } \text{answer}
\]

Vanilla query executor

Cross products are expensive, can replace with \textbf{joins}

Avoid scanning the entire table by accessing subsets of records through an \textbf{index}
Optimizing the Template Implementation

SELECT \( a_1, a_2, \ldots, a_k \)
FROM \( R_1, R_2, \ldots, R_n \)
WHERE conditions

\[
\text{answer} = \{ \}
\]

\[
\text{for } t_1 \text{ in } R_1 \text{ do }
    \text{for } t_2 \text{ in } R_2 \text{ do }
    \quad \ldots
    \quad \text{for } t_n \text{ in } R_n \text{ do }
    \quad \text{if } \text{conditions} \text{ then }
        \text{answer} = \text{answer} \cup \{(a_1, \ldots, a_k)\}
\]

\text{return answer}

Vanilla query executor

Cross products are expensive, can replace with \textbf{joins}

Avoid scanning the entire table by accessing subsets of records through an \textbf{index}

Query plan can be \textbf{optimized} to minimize the execution overhead
Optimizing the Template Implementation

```
SELECT a_1, a_2, ..., a_k
FROM R_1, R_2, ..., R_n
WHERE conditions
```

Data can be stored in disks for persistency and low cost and buffered in DRAM

```
answer = {}
for t_1 in R_1 do
    for t_2 in R_2 do
        ...
        for t_n in R_n do
            if conditions
                then answer = answer U {(a_1, ..., a_k)}
return answer
```

Vanilla query executor

Cross products are expensive, can replace with joins

Avoid scanning the entire table by accessing subsets of records through an index

Query plan can be optimized to minimize the execution overhead
Optimizing the Template Implementation

\[
\text{SELECT } a_1, a_2, \ldots, a_k \\
\text{FROM } R_1, R_2, \ldots, R_n \\
\text{WHERE } \text{conditions}
\]

Perform **locking** at fine-granularity to enable parallel execution of transactions
Optimizing the Template Implementation

\[
\text{SELECT } a_1, a_2, \ldots, a_k \\
\text{FROM } R_1, R_2, \ldots, R_n \\
\text{WHERE conditions}
\]

Perform locking at fine-granularity to enable parallel execution of transactions

Ensure that parallel execution results are equivalent to serial execution
Optimizing the Template Implementation

```
SELECT a_1, a_2, ..., a_k
FROM R_1, R_2, ..., R_n
WHERE conditions
```

- Perform **locking** at fine-granularity to enable parallel execution of transactions.
- Ensure that parallel execution results are **equivalent to serial** execution.
- Ensure the database can tolerate failures by providing **durability and high availability**.
Optimizing the Template Implementation

\[
\text{SELECT } a_1, a_2, \ldots, a_k \\
\text{FROM } R_1, R_2, \ldots, R_n \\
\text{WHERE } \text{conditions}
\]

Perform locking at fine-granularity to enable parallel execution of transactions.

Ensure that parallel execution results are equivalent to serial execution.

Ensure the database can tolerate failures by providing durability and high availability.

Can scale up to multicore processors and scale out to distributed systems.

Transactions

Database

Lock(DB)

Execute transaction

Unlock(DB)

Query Executor
Topics in CS 764

• Query processing and buffer management (Lectures 2–7)
  – Join (hash join, radix join)
  – Buffer management (disk-based, NVM-based)
  – Query optimization

• Advanced transaction processing (Lectures 8–22)
  – Two-phase locking
  – Isolation
  – Optimistic concurrency control
  – B-tree and radix-tree
  – Fault tolerance

• Cloud-native databases (Lectures 23–27)
  – Amazon Aurora, Snowflake
  – PushdownDB

• Guest lectures from AWS and Oracle
Course Logistics
Course Information

Canvas page: [https://canvas.wisc.edu/courses/259034](https://canvas.wisc.edu/courses/259034)
Piazza: [piazza.com/wisc/fall2021/cs764/home](https://piazza.com/wisc/fall2021/cs764/home)

Prerequisite: CS 564

Office Hour: Monday 2:30—3:30pm on zoom (link available on canvas)

Reference textbooks:
- Red book
- Cow book
Grading

Paper review: 15%

Exam: 35%

Project proposal: 10%

Project presentation: 10%

Project final report: 30%
Paper Review (15%)

**Paper reading**: one classic/modern paper per lecture
- username: cs764  password: dbguru
Paper Review (15%)

**Paper reading**: one classic/modern paper per lecture
- username: cs764  password: dbguru

**Upload review**: [https://wisc-cs764-f21.hotcrp.com](https://wisc-cs764-f21.hotcrp.com) (must be submitted before the lecture starts in order to be graded)
- Overall merit
- Paper summary
  - What main research problem/challenge did the paper address?
  - What is the key contribution of the paper?
- Comments and questions
  - Particular aspects you like or dislike about the paper
  - Questions about that paper that you wish to be discussed in the lecture
Paper Review (15%)

**Paper reading**: one classic/modern paper per lecture
- username: cs764   password: dbguru

**Upload review**: [https://wisc-cs764-f21.hotcrp.com](https://wisc-cs764-f21.hotcrp.com) (must be submitted before the lecture starts in order to be graded)
- Overall merit
- Paper summary
  - What main research problem/challenge did the paper address?
  - What is the key contribution of the paper?
- Comments and questions
  - Particular aspects you like or dislike about the paper
  - Questions about that paper that you wish to be discussed in the lecture

**Grading**: You can skip up to 2 reviews without losing points; otherwise 1% of total grade (up to 15%) is deducted for each missing review
Course Project (50%)

In groups of 2–4 students

A list of example project ideas will be provided but you are encouraged to propose your own ideas

– A list of project ideas for Fall 2020 is available on the course website
Course Project (50%)

In **groups of 2–4 students**

A list of example project ideas will be provided but you are encouraged to propose your own ideas

- A list of project ideas for Fall 2020 is available on the course website

Important dates

- Create teams and submit proposal: Oct. 25
- Project meetings with instructor: Nov. 24
- Presentation: Dec. 13 & 15
- Paper submission: Dec. 18
Exam (35%)

Take-home exam
- Open-book, open-notes
- You can use any material provided in this course or on the Internet

Sample exam questions are available on course website

Important dates
- **Nov. 10** Exam review
- **Nov. 15** Mid-term exam
Computation Resources

CloudLab

Chameleon
  https://www.chameleoncloud.org (project name: ngdb)
Lecture Mode

If you choose in-person mode
  – Strongly suggest wearing a face mask

Each lecture will be streamed online using the same zoom link

Each lecture will be recorded and the video recording will be available on canvas
Read the following paper and submit review

Email the instructor if you have problems registering for https://wisc-cs764-f21.hotcrp.com

Enroll on Piazza
– piazza.com/wisc/fall2021/cs764/home