WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

# 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 name

**Product**

| **name** | **category** | **price** | **manufacturer** |
|---|---|---|---|
| iPad | tablet | $399.00 | Apple |
| Surface | tablet | $299.00 | Microsoft |
| … | … | … | … |

# 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 name

**Product**

| **name** | **category** | **price** | **manufacturer** |
| --- | --- | --- | --- |
| iPad | tablet | $399.00 | Apple |
| Surface | tablet | $299.00 | Microsoft |
| … | … | … | … |

record/tuple/row

# 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 name

**Product**

attribute/column

| name | category | price | manufacturer |
| --- | --- | --- | --- |
| iPad | tablet | $399.00 | Apple |
| Surface | tablet | $299.00 | Microsoft |
| … | … | … | … |

record/tuple/row

# SQL Queries

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

# A Database Template

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

Vanilla query executor

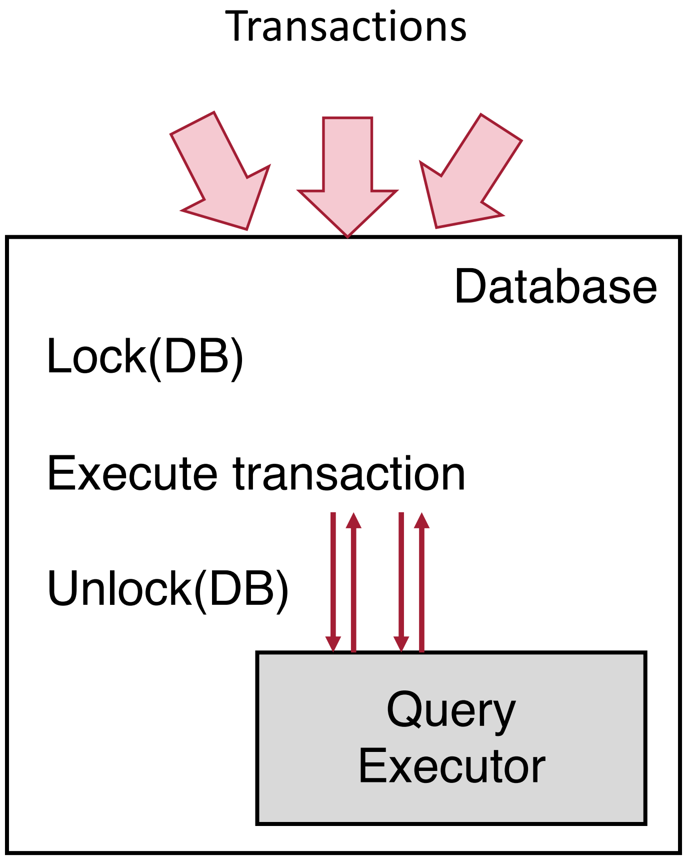
```
answer = {}
for t₁ in R₁ do
  for t₂ in R₂ do
    …
    for tₙ in Rₙ do
      if conditions
        then answer = answer ∪ {(a₁,…,aₖ)}
return answer
```

# A Database Template

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

Vanilla query executor

```
answer = {}
for t₁ in R₁ do
  for t₂ in R₂ do
    …
    for tₙ in Rₙ do
      if conditions
        then answer = answer ∪ {(a₁,…,aₖ)}
return answer
```




A DBMS can be heavily optimized beneath this simple interface

# Optimizing the Template Implementation

SELECT $a_1, a_2, \ldots, a_k$

FROM $R_1, R_2, \ldots, R_n$

WHERE conditions

Vanilla query executor

```
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
```

Cross products are expensive, can replace with **joins**

# Optimizing the Template Implementation

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

Vanilla query executor

```
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
```

Cross products are expensive, can replace with **joins**

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

# Optimizing the Template Implementation

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

Vanilla query executor

```
answer = {}
for t₁ in R₁ do
  for t₂ in R₂ do
    …
      for tₙ in Rₙ do
        if conditions
          then answer = answer ∪ {(a₁,…,aₖ)}
return answer
```

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

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

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

Vanilla query executor

```
answer = {}
for t₁ in R₁ do
  for t₂ in R₂ do
    …
    for tₙ in Rₙ do
      if conditions
        then answer = answer ∪ {(a₁,…,aₖ)}
return answer
```

Cross products are expensive, can replace with **joins**

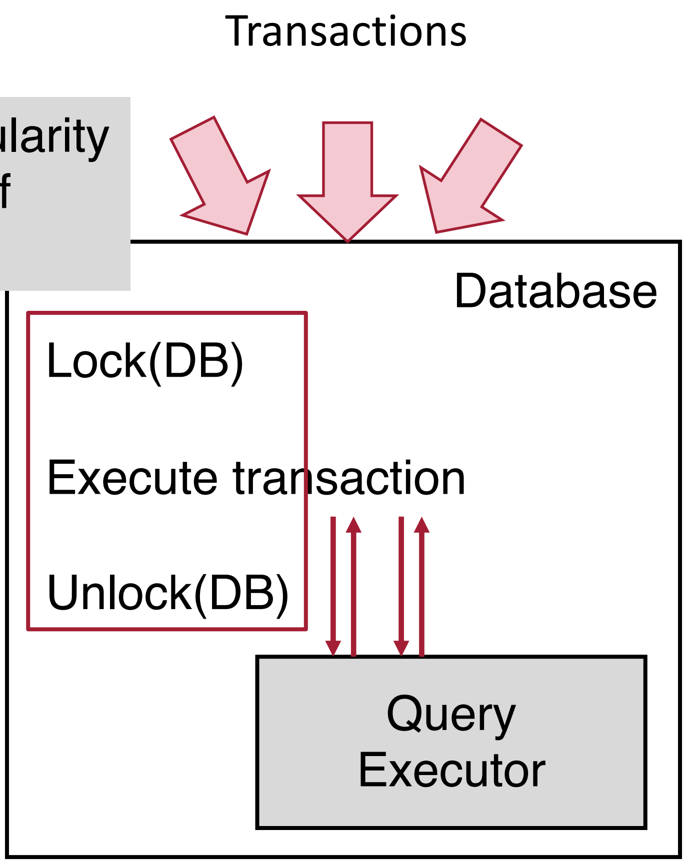
Avoid scanning the entire table by accessing subsets of records through an **index**

Query plan can be **optimized** to minimize the execution overhead

# Optimizing the Template Implementation

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

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

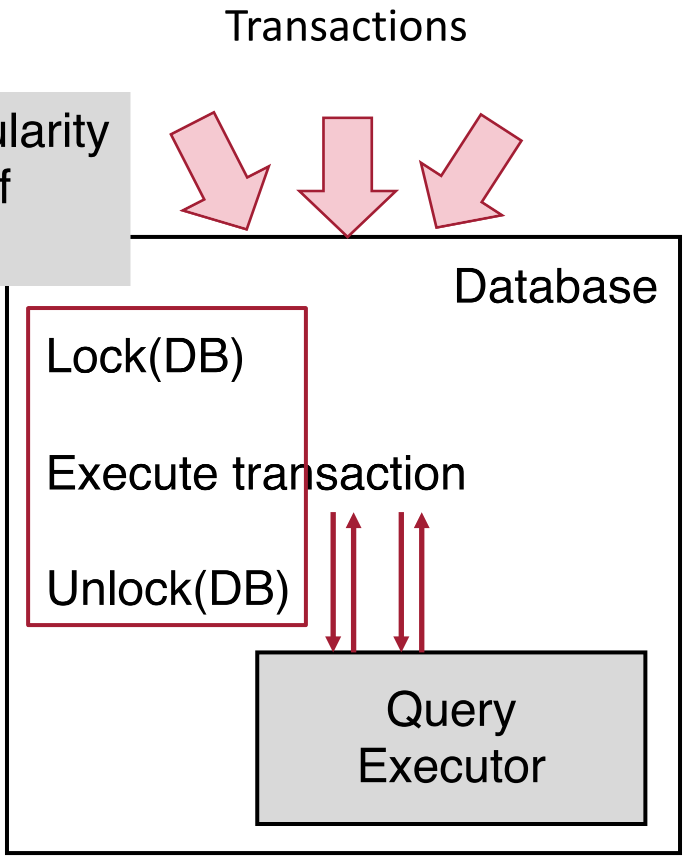


# Optimizing the Template Implementation

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

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

Ensure that parallel execution results are **equivalent to serial** execution



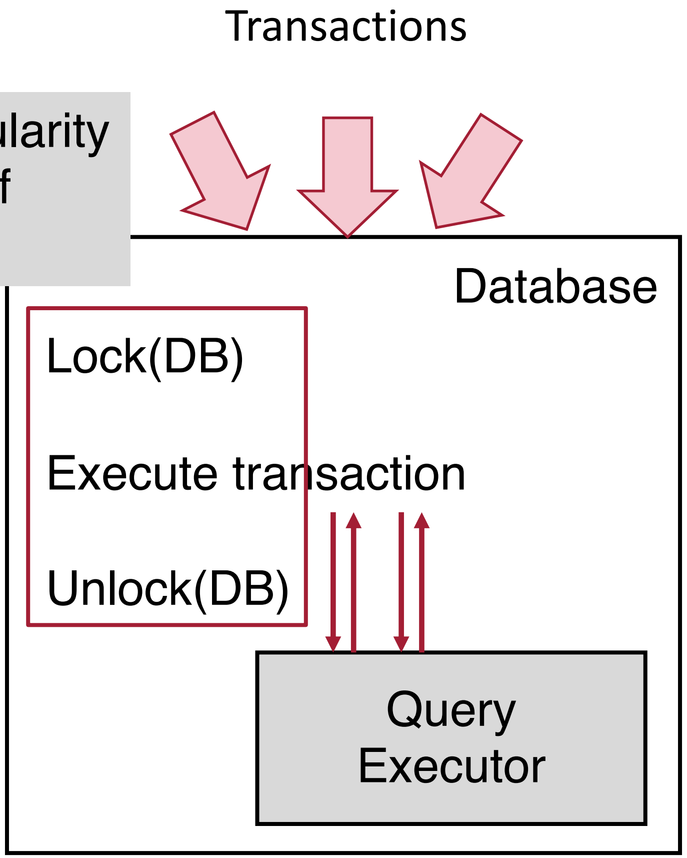
# Optimizing the Template Implementation

SELECT $a_1, a_2, \ldots, a_k$
FROM $R_1, R_2, \ldots, R_n$
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

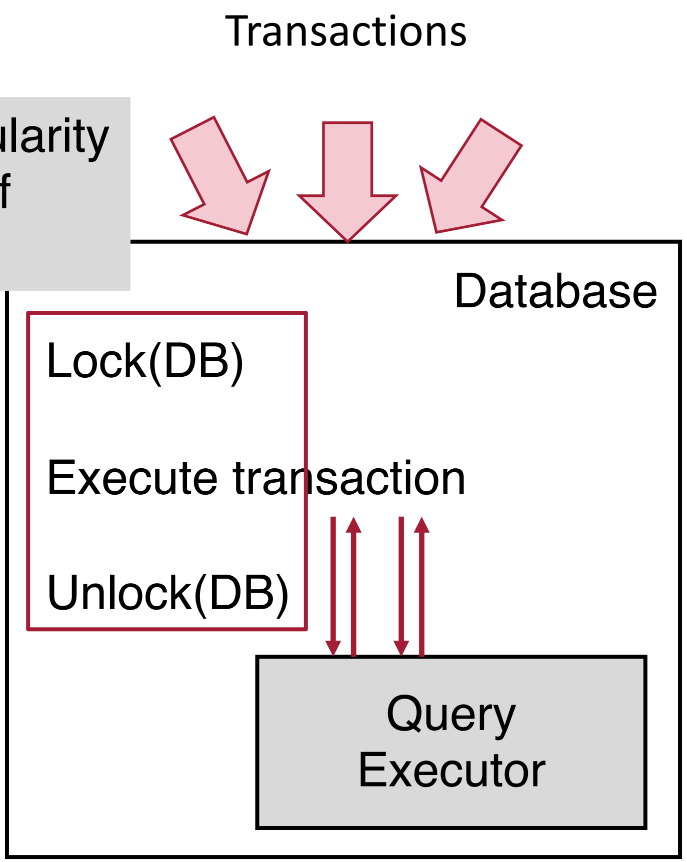
SELECT $a_1, a_2, \ldots, a_k$
FROM $R_1, R_2, \ldots, R_n$
WHERE conditions

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**

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



# 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

Course Website: http://pages.cs.wisc.edu/~yxy/cs764-f21/

Canvas page: https://canvas.wisc.edu/courses/259034

Piazza: 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 (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 (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
- Example projects in 2019 http://pages.cs.wisc.edu/~yxy/cs764-f21/misc/dawn19.pdf

# 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
- Example projects in 2019 http://pages.cs.wisc.edu/~yxy/cs764-f21/misc/dawn19.pdf

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

https://www.cloudlab.us/signup.php?pid=NextGenDB (project name: NextGenDB)

## 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

# Before next lecture

Read the following paper and submit review

- Leonard D. Shapiro, Join Processing in Database Systems with Large Main Memories. ACM Trans. Database Syst. 1986.

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

Enroll on Piazza

- piazza.com/wisc/fall2021/cs764/home