WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 764: Topics in Database Management Systems

Lecture 1: Introduction

Xiangyao Yu

09/07/2022

Who am I?

Name: Xiangyao Yu

Assistant professor in Computer Sciences, Database Group

Postdoc and PhD at MIT on databases and computer architecture

Research interests:

- Cloud-native databases
- New hardware for database
- Transactions and HTAP

Basic Information

Course website: https://pages.cs.wisc.edu/~yxy/cs764-f22/index.html

Instructor: Xiangyao Yu

Office hours:

- Monday 2:30pm–3:30pm CS 4361
- Or schedule by email

TA: Keren Chen

Office hours: schedule by email

Piazza for discussions and questions

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 relational 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 relational 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 relational 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

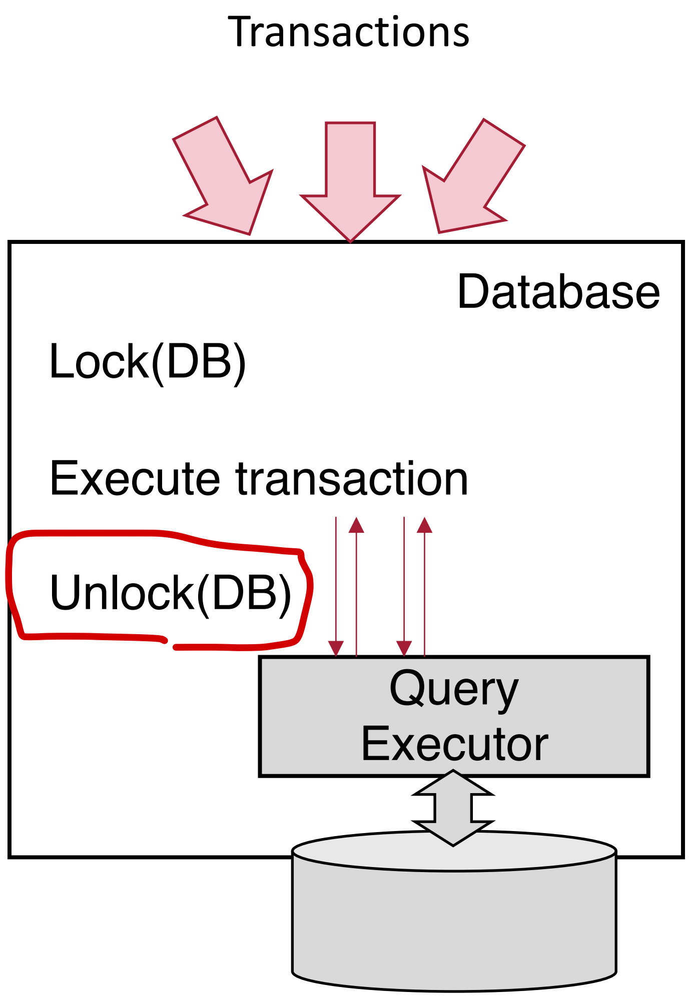
```
answer = {}
for t1 in R1 do
  for t2 in R2 do
    …
    for tn in Rn do
      if conditions
        then answer = answer U {(a1,…,ak)}
return answer
```

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 U {(a₁,…,aₖ)}
return answer
```

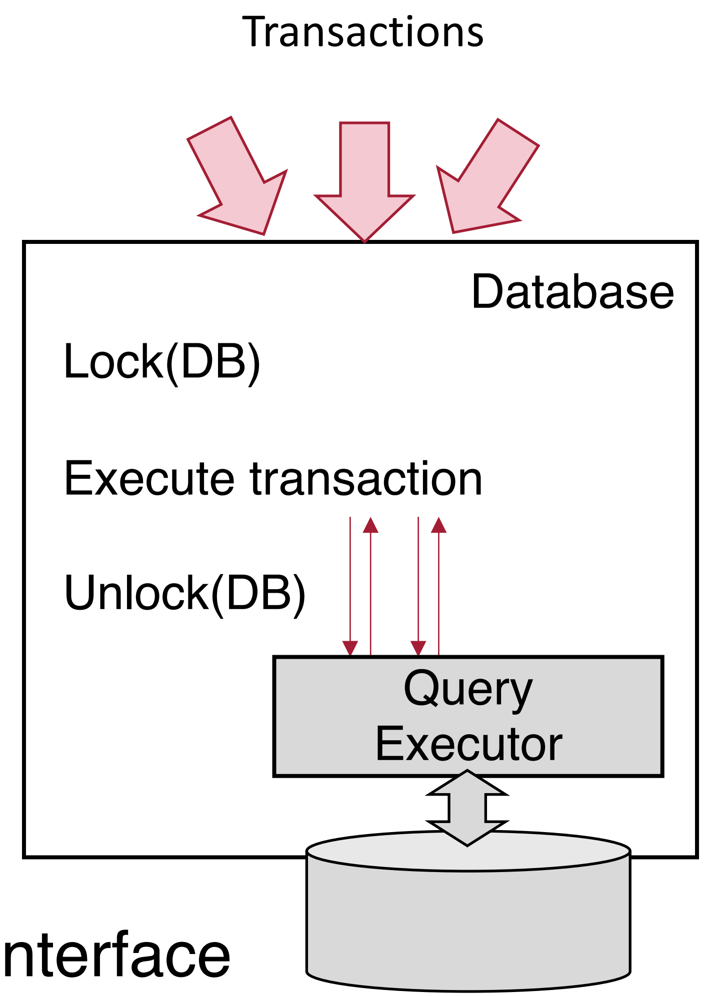


A Database Template

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 U {(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 ∪ {(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
```

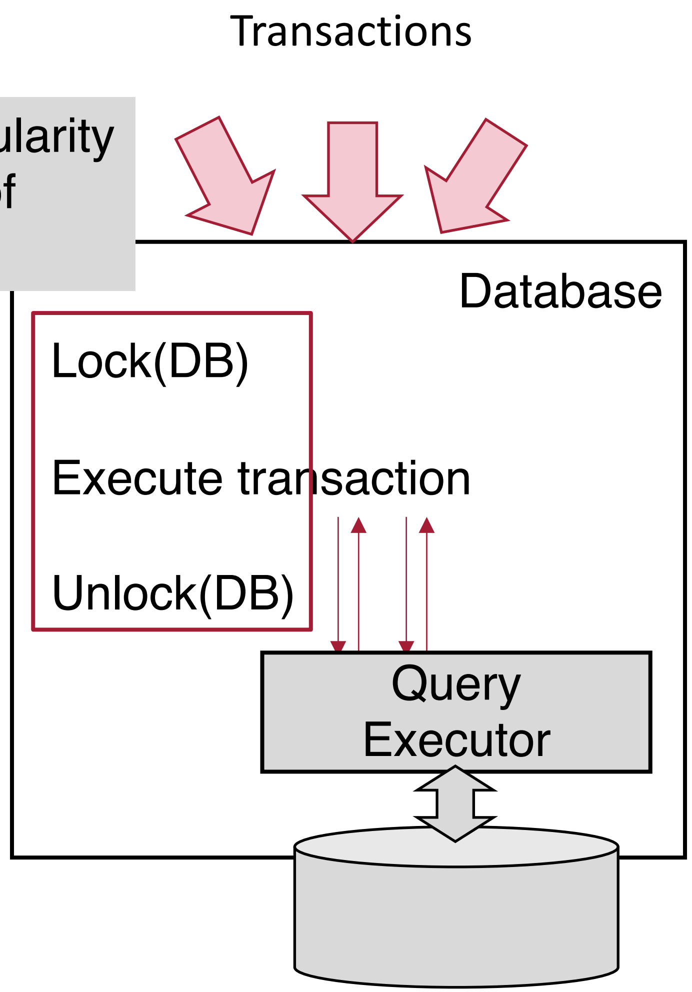
Cross products are expensive, can replace with **joins**

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

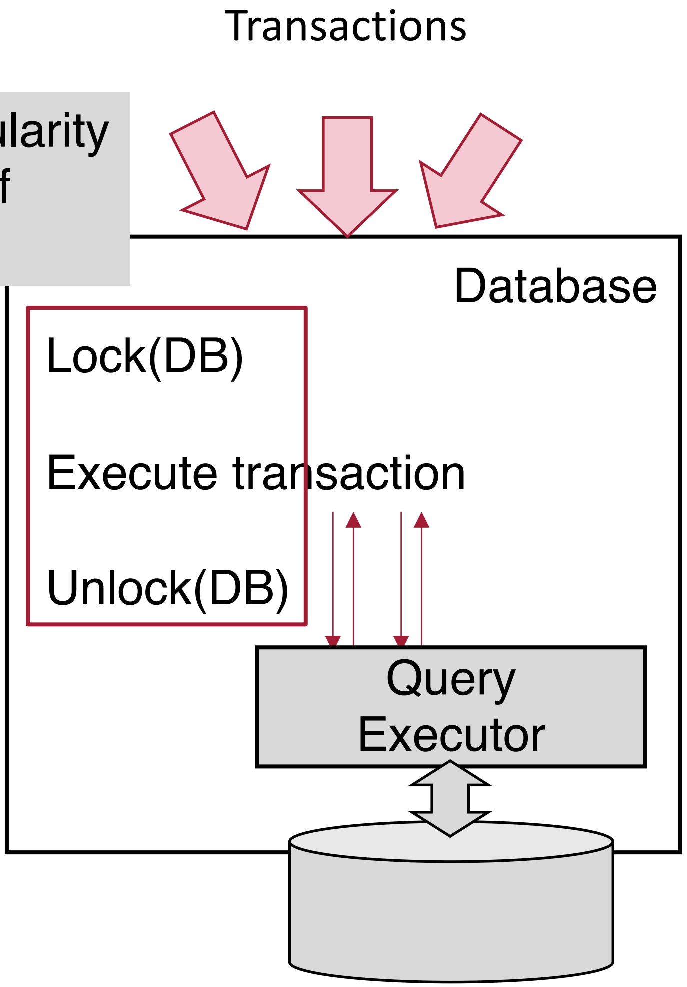
Perform **locking** at fine-granularity to enable parallel execution of transactions



Optimizing the Template Implementation

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

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

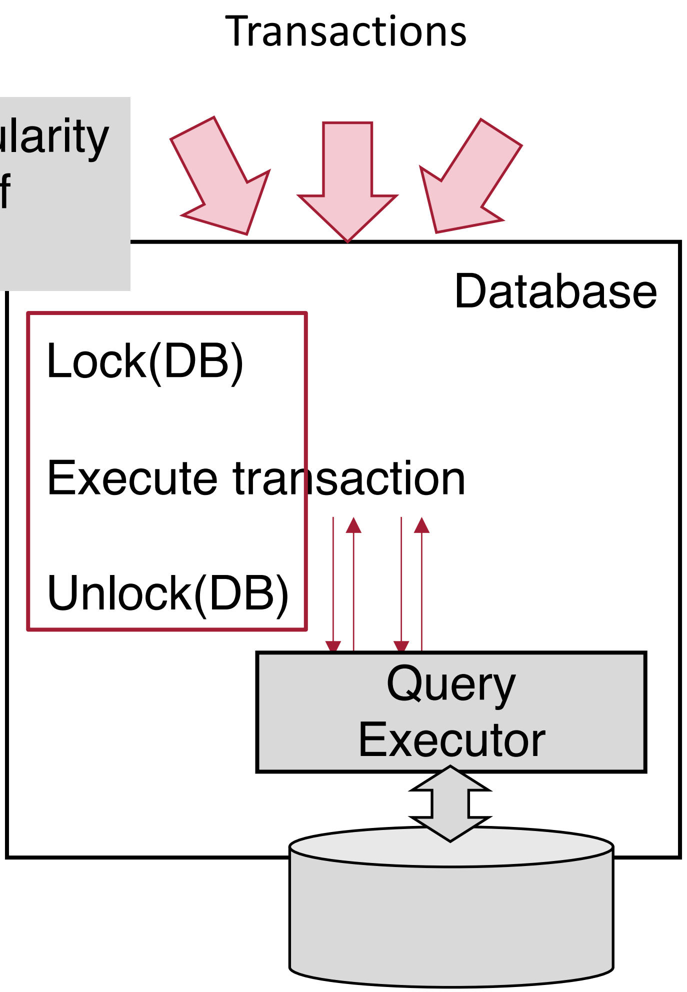


Optimizing the Template Implementation

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**



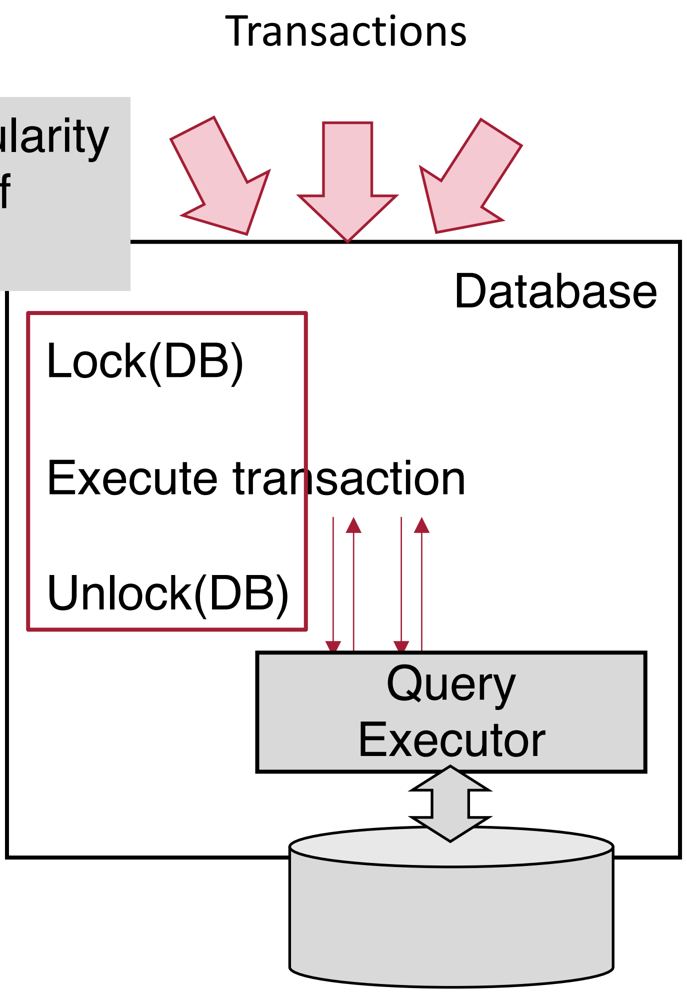
Optimizing the Template Implementation

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 2–8)

- Join
- Buffer management
- Query optimization
- Column store
- Parallel database

Advanced transaction processing (Lectures 9–20)

- Two-phase locking
- Isolation
- Optimistic concurrency control
- B-tree and radix-tree
- Fault tolerance

Advanced topics in databases (Lectures 21–27)

- Cloud-native databases
- Database with new hardware technologies

Guest lectures from Oracle and TiDB

Topics in CS 764

Query processing and buffer management (Lectures 2–8)

- Join
- Buffer management
- Query optimization
- Column store
- Parallel database

Advanced transaction processing (Lectures 9–20)

- Two-phase locking
- Isolation
- Optimistic concurrency control
- B-tree and radix-tree
- Fault tolerance

Advanced topics in databases (Lectures 21–27)

- Cloud-native databases
- Database with new hardware technologies

Guest lectures from Oracle and TiDB

Topics in CS 764

Query processing and buffer management (Lectures 2–8)

- Join
- Buffer management
- Query optimization
- Column store
- Parallel database

Advanced transaction processing (Lectures 9–20)

- Two-phase locking
- Isolation
- Optimistic concurrency control
- B-tree and radix-tree
- Fault tolerance

Advanced topics in databases (Lectures 21–27)

- Cloud-native databases
- Database with new hardware technologies

Guest lectures from Oracle and TiDB

Topics in CS 764

Query processing and buffer management (Lectures 2–8)
- Join
- Buffer management
- Query optimization
- Column store
- Parallel database

Advanced transaction processing (Lectures 9–20)
- Two-phase locking
- Isolation
- Optimistic concurrency control
- B-tree and radix-tree
- Fault tolerance

Advanced topics in databases (Lectures 21–27)
- Cloud-native databases
- Database with new hardware technologies

Guest lectures from Oracle and TiDB

Course Logistics

Course Information

Course Website: http://pages.cs.wisc.edu/~yxy/cs764-f22/

Canvas page: https://canvas.wisc.edu/courses/309574

Piazza: piazza.com/wisc/fall2022/cs764/

Zoom: You can take the lectures online

- https://uwmadison.zoom.us/j/93625999493?pwd=Si9PWEdpWnhNaTdxMG1zVXFFcXk0QT09
- passcode: 764

Prerequisite: CS 564

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-f22.hotcrp.com (must submit 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**
  - 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-f22.hotcrp.com (must submit 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**
  - 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

Exam (35%)

In-class exam

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

One lecture to review exams in previous years

Sample exam questions are available on course website

Course Project (50%)

In **groups of 2–4** students

Project ideas will be provided but you are encouraged to propose your own ideas

- Project ideas for Fall 2020 and 2021 are available on the course website
- Three example projects are available on the course website (two papers based on course projects accepted to SIGMOD 2022 and SIGMOD 2023)

Course Project (50%)

In **groups of 2–4** students

Project ideas will be provided but you are encouraged to propose your own ideas

- Project ideas for Fall 2020 and 2021 are available on the course website
- Three example projects are available on the course website (two papers based on course projects accepted to SIGMOD 2022 and SIGMOD 2023)

Important dates

- Create teams and submit proposal: Oct. 24
- Project meetings with instructor: Nov. 21
- Presentation: Dec. 12 & 14
- Paper submission: Dec. 19

Computation Resources

CloudLab

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

Chameleon

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

Lectures in Hybrid Mode

Each lecture will be streamed online on zoom

- https://uwmadison.zoom.us/j/93625999493?pwd=Si9PWEdpWnhNaTdxMG1zVXFFcXk0QT09 (passcode: 764)

Lectures will be recorded and the video recording will be available on canvas (canvas -> zoom)

Waitlist

Class size limited to ~60

If you are enrolled but don’t want to take the class, please drop ASAP

If you are on the waitlist, we will admit students first-come-first-serve

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-f22.hotcrp.com after Friday

Enroll on Piazza

- piazza.com/wisc/fall2022/cs764/