

CSE462/562: Database Systems (Fall 24)

Lecture 1: Introduction & Course Logistics;

POSIX I/O Interface

1/29/2024

Nsc 205, T&H 5:00 pm – 6:20 pm. In-person attendance required.

Find more on course website & Piazza:

https://cse.buffalo.edu/~zzhao35/teaching/cse562_fall24

<https://piazza.com/buffalo/fall2024/cse462562/home>

Today's agenda

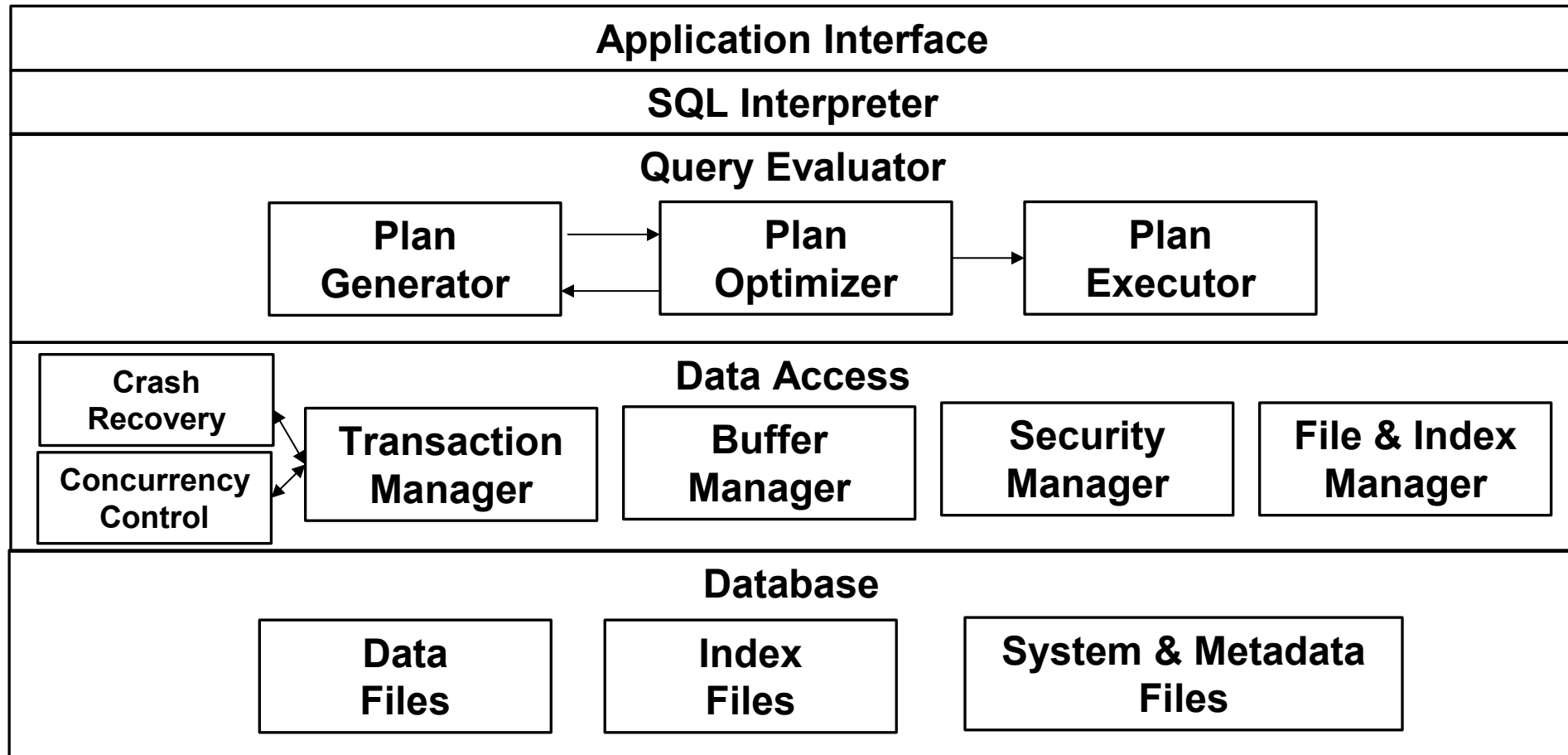
- Introduction
 - What is a Database?
 - What is a Database Management System?
 - What is this course about and why should I care?
- Course logistics
- Getting started with project: POSIX I/O Interface

What is a Database?

- Database is
 - a collection of interrelated data
 - often organized in a certain structure for convenient and efficient access
- Databases are found almost everywhere, sometimes unnoticed
 - Business: sales, accounting, human resource, IT support, ...
 - Financial industry: banking, credit card, investment platform
 - University: student records, course registration, LMS (e.g., UB Learns), ...
 - Some less obvious examples of databases
 - Software package and configuration DB (e.g., windows registry)
 - Your photo library (e.g., Google Photos)
 - Your personal finance records
 - ...

What's a DataBase Management System?

- DataBase Management System (DBMS) is a software system for convenient and efficient data access over databases.



Why using a DataBase Management System?

- Let's review an example of how to manage a database.

How to manage a database?

- Suppose I'd like to track my daily spending
- What I can do:
 - Step 1: collect all the receipts



- Step 2: do some analysis
 - How much did my spend on grocery and fast food in February?
 - How much could I have saved if I cook by myself in February?
 - What about January/last quarter/last year/past five years?



How to manage a database?

- Suppose I'd like to track my daily spending

- What I can do:

- Step 1: collect all the receipts
- Step 2: write them down on a notebook

Date	Amount	Description
2/1	\$20.21	Grocery
2/2	\$10.54	Fast food
2/3	\$39.22	Cell phone bill
...		
2/27	\$33.00	Clothes

- Step 3: do some analysis

- How much did my spend on grocery and fast food in February?
- How much could I have saved if I cook by myself in February?
- What about January/last quarter/last year/past five years?



How to manage a database?

- Suppose I'd like to track my daily spending

- What I can do:

- Step 1: collect all the receipts
- Step 2: ~~write them down on a notebook~~
store them in a text file

Date	Amount	Description
2/1	\$20.21	Grocery
2/2	\$10.54	Fast food
2/3	\$39.22	Cell phone bill
...		
2/27	\$33.00	Clothes

- Step 3: do some analysis

- How much did my spend on groceries
- How much could I have saved if I cooked
- What about January/last quarter/last year

```
f = open('myspend_feb_22.txt', 'r')
grocery = 0
fast_food = 0
for line in f:
    date, amount, desc = line.split(' ')
    if desc == 'Fast food':
        fast_food += eval(amount)
    elif desc == 'Grocery':
        grocery += eval(amount)
.....
```


How to manage a database?

- Suppose I'd like to track my daily spending
- What I can do:
 - Step 1: collect all the receipts
 - Step 2: ~~write them down on a notebook~~
~~store them in a text file~~
use a spreadsheet
- Step 3: do some analysis
 - How much did my spend on grocery and fast food
 - How much could I have saved if I cook by myself
 - What about January/last quarter/last year/past year

Date	Amount	Description
2/1	\$20.21	Grocery
2/2	\$10.54	Fast food
2/3	\$39.22	Cell phone bill
...		
2/27	\$33.00	Clothes

	A	B	C	D	E
1	Date	Amount	Description		
2	1-Feb	20.21	Grocery		
3	2-Feb	10.54	Fast food		
4	3-Feb	39.22	Cell phone		
5					
6					
7		Grocery	=SUMIFS(B2:B4,C2:C4,"Grocery")		

How to manage a database?

- Suppose I'd like to track my daily spending

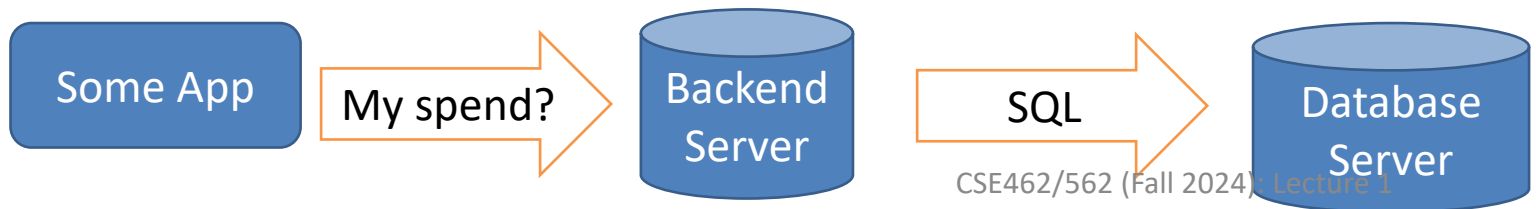
- What I can do:

- Step 1: collect all the receipts
- Step 2: ~~write them down on a notebook~~
~~store them in a text file~~
~~use a spreadsheet~~
use/build a personal finance app

- Step 3: do some analysis

- How much did my spend on grocery and fast food in February?
- How much could I have saved if I cook by myself in February?
- What about January/last quarter/last year/past five years?

Date	Amount	Description
2/1	\$20.21	Grocery
2/2	\$10.54	Fast food
2/3	\$39.22	Cell phone bill
...		
2/27	\$33.00	Clothes



```
SELECT category, SUM(amount)
FROM spend
WHERE userid = 123456
GROUP BY category;
```

Why using a DataBase Management System?

- DataBase Management System (DBMS) is a software system for convenient and efficient data access over databases,

which provides:

- Data abstraction
 - Flexible data manipulation and query interfaces
 - Scalable data storage
 - Efficient query and transaction processing
- Integrity checks
- Concurrency control and atomicity
- Fault tolerance
- Security and privacy
- ...

What does this course cover?

- The design and implementation of DataBase Management System (DBMS)
 - **Relational DBMS (RDBMS)** as a case study
 - Stores tables that consist of rows and columns
 - Declarative query language (SQL) in the simple yet powerful relational model
 - Focus on principles and techniques generally applicable in Data Management
- Note, this course is not about *(but we assume you have learned these somewhere else):*
 - Database design
 - The relational model and the SQL language (we'll briefly review them)
 - Programming/data structure/algorithm analysis/math...

Why should I care about DBMS internals?

- > 90 billion dollar worth industry
 - Many more are directly or indirectly using DBMS products
- Many vendors and products:
 - Relational: MySQL, Oracle DB, Microsoft SQL Server, IBM Db2, PostgreSQL, SQLite...
 - Graph DB and Graph data processing: Neo4j, Virtuoso, GraphLab, Spark GraphX, ...
 - Stream Processing: Apache Flink, Spark Streaming, Apache Storm, ...
 - Semi-structured DB: MongoDB, CouchBase, DocumentDB, ...
 - Distributed database: Google Spanner, Microsoft CosmosDB, ...
 - ...
- Used by many other research and application areas:
 - Artificial Intelligence/data mining/search engine/social media/fintech/...

Why should I care about DBMS internals?

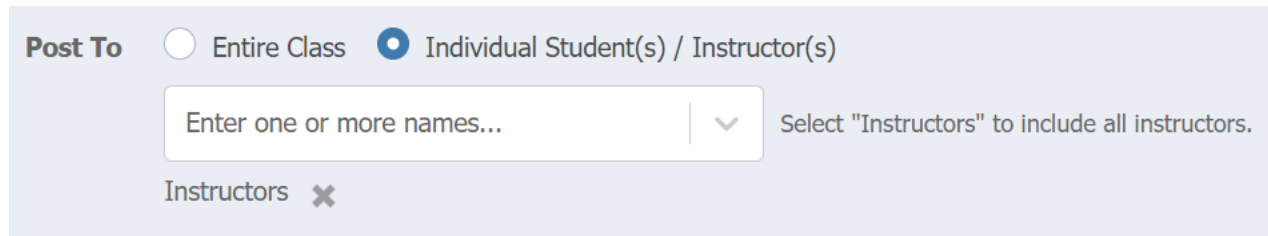
- Huge demand in industry for those who can
 - query/manipulate data in database efficiently
 - fine-tune the imperfect DBMS/big data processing systems
 - work seamlessly with the data infrastructure team
- An actively researched area that
 - has strong real-life impacts and connection to the industry
 - has many related open engineering and research positions
- The goal of this course:
 - understanding the common problems and solutions in data management
 - gaining hands-on experience with building a complex software system
 - to be helpful in your future industrial/academic career

Logistics

- Nsc 205, T&H 5:00 pm – 6:20 pm.
 - In-person attendance required.
- Office hours:
 - TA (Congying Wang)
 - Tuesdays 11:00 AM - 1:00 PM, Thursdays 10:00 AM - 12:00 PM @ Davis 338Y
 - Instructor (Zhuoyue Zhao)
 - Wednesdays 1:00 PM - 5:00 PM @ Davis 338I
 - Rules:
 - First-come first-serve
 - Please come with concrete questions about course materials/projects/assignments/exams
 - Not intended for code debugging, troubleshooting of your dev environment, etc.
- Find more on course website:
https://cse.buffalo.edu/~zzhao35/teaching/cse562_fall24/

Logistics

- We mainly use Piazza for communication:
 - <https://piazza.com/buffalo/fall2024/cse462562/home>
 - Please post messages on Piazza instead of sending emails
- When you have any private question/request for the instructor or TA:
 - please select “Instructors” in Post To



The screenshot shows the 'Post To' section of a Piazza post form. It features two radio buttons: 'Entire Class' (unselected) and 'Individual Student(s) / Instructor(s)' (selected). Below the radio buttons is a text input field with the placeholder text 'Enter one or more names...' and a dropdown arrow. To the right of the input field is the text 'Select "Instructors" to include all instructors.' Below the input field, the word 'Instructors' is displayed with a small 'x' icon to its right, indicating it has been selected.

Logistics

- Important Dates:
 - Mid-term exam: 10/17/2024, Nsc 205, 5:05 pm – 6:20 pm (75 minutes)
 - Final exam: 12/12/2024, Nsc 205, 7:30 pm – 9:00 pm (90 minutes)
- Exam conflict policy:
 - If you have [final exam conflicts](#) as defined by the Office of the Registrar
 - **please notify the instructor on Piazza by 9/9/2024**
 - (we might not have enough seats if you do not notify us by that date)
 - you may still opt for the original final exam at any time with one-week prior notice

Grading

- Grading
 - Mid-term exam: 15%
 - Final exam: 20%
 - Homework Assignments: 20%
 - Projects: 45%

- Grading policy:
 - No curving.

[0, 10)	[10, 20)	[20, 30)	[30, 40)	[40, 50)	[50, 60)	[60, 70)	[70, 80)	[80, 90)	[90, +∞)
F	D	C-	C	C+	B-	B	B+	A-	A

Exams and Assignments

- 4 written assignments
 - 5% each
 - Similar problems that will appear in exams
 - Must be written electronically, e.g., in LaTeX (encouraged) or word
 - Do not submit scans of handwriting except graphs and plots
- Exams
 - **Open-book exams**
 - Only **paper-copy** of the course slides, the written assignments and solutions, the optional textbook, and your lecture notes are allowed
 - No electronic devices except a calculator

Course project

- Build a mini RDBMS through 5 projects (C++ 17)
- Teams allowed with up to 2 students
 - teamwork allowed only **within teams**
 - see academic integrity policy for details
- **Using generative AI is disallowed**
 - No ChatGPT
 - No Github Copilot (if you use an IDE, please make sure to disable it)
- Code must be kept in **private** Github repository, even after this semester

Academic Integrity Policy

- Academic integrity is critical to the learning process. It is your responsibility to understand and follow all the departmental and university academic integrity policies.
- **Zero tolerance** towards academic integrity violations, which includes but are not limited to
 - Sharing/copying code in projects or
 - Plagiarizing write-ups
 - Cheating in exam
 - Making project code publicly available or available to any current or future students
 - Submitting code repository that does not belong to you
 - ***Use of generative AI in this class for any coursework***
- Any AI violation will result in **an F grade** and will be reported to the Office of Academic Integrity
 - unless it's an honest mistake that does not give anyone any undue advantage
 - (e.g., you accidentally set your Github repo to public but changed it back before anyone accesses it)

More on Academic Integrity Policy

- Think of the course projects as take-home exams:
 - you must complete them by yourself (or with your teammate for coding only)
 - please do not discuss any project specifics outside your team
- Examples of AI violation related to course project:
 - Viewing/committing/submitting code written by anyone who is not your teammate
 - verbatim or with modification
 - ***including those generated or adapted from outputs from generative AI software (e.g., ChatGPT)***
 - Viewing/copying/rephrasing answers found online or from a past or current student
- What is allowed and encouraged (on Piazza/in lecture/offline, publicly or privately)
 - Ask questions about lectures/projects/homework assignments
 - Preparation for mid-term and final exams
 - Seek clarification about projects/homework assignments
 - If you're unsure, please do ask.

Course project

- Instructions for projects:
 - Project pages contain very detailed instructions.
 - If something requires clarification, it's most likely covered there.
 - Still have questions on project or found bugs?
 - Feel free to post it on Piazza (though we may point you back to the instructions).
 - Your team will get 1 extra credit towards your final grade for every validated bug or question that cannot be answered by the project instruction.
- Where to find project pages:
https://cse.buffalo.edu/~zzhao35/teaching/cse562_fall24/

CSE 462/562: Database Systems (Fall 2024)

Course home **Projects** ▾ Piazza UB Learns

Late policy

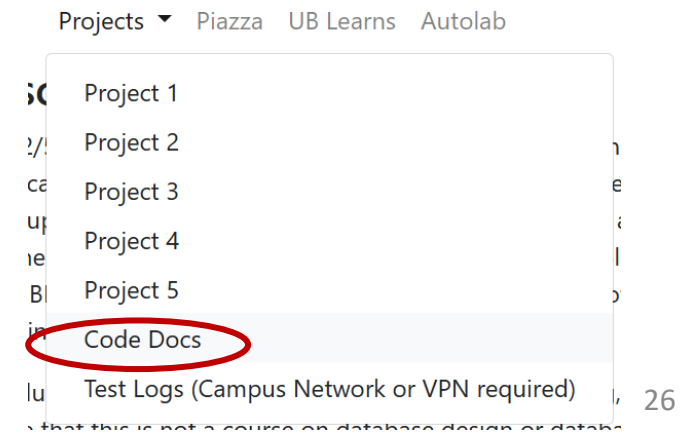
- Late policy:
 - Each student will have **three (3)** grace days throughout the semester.
 - For **each project/assignment**, you may use **up to one (1)** grace day with no penalty
 - Examples:
 - You submit project 1 - 3 within a day after the posted deadlines
 - No penalty to the grades.
 - You submit project 1, HW1, project 2, project 3 within a day after the posted deadlines
 - No penalty to the grades of project 1, HW1, project 2.
 - No points will be received for project 3.
 - You submit HW1 after one day after the posted deadline
 - No points will be received for HW1 (but it will be graded to provide you feedbacks)

Setting up your dev container

- Please follow instructions for project 1 - lab 0

Project 0

- Project and team sign-up
 - Please find a teammate, and follow the repository and sign-up instructions
 - **Due 9/2, 11:59:59 PM EDT, no late submissions allowed in project 0**



Project 1

- Build a simple C++ class that encapsulates `POSIX I/O interfaces`
 - Goal: get familiar with reading documentations
 - Use ``man <function_name>`` command to find syscall docs
 - Find code docs of Taco-DB from the Project drop-down menu
 - **Due 9/9, 11:59:59 PM EDT, see late policy**
 - **Submission will be open no later than 9/3.**

File System Interface

- POSIX I/O interface
 - A standard synchronous I/O interface
 - Agnostic to the underlying storage device/file system

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.

`open(2)`: open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

opens the file at path
/data/a.dat

1. read-only access
2. create the file if it does not exist

The permission bits if the file is created.
0644 = rw allowed for user (file owner);
read only for group & others.

Case 1: `fd >= 0` on success.

Case 2: `fd == -1` if an error occurred -- check `errno` for reasons; also see `strerror(3)`

File System Interface

- POSIX I/O interface
 - A standard synchronous I/O interface
 - Agnostic to the underlying storage device/file system

`open(2)`: open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

`pread(2)`, `pwrite(2)`: read from or write to a file descriptor at a given offset

```
char buf[4096];
ssize_t sz = pread(fd, buf, 4096, 1048576);
if (sz == 4096) /* success */; else /* error */;
```

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.

reading 4096 bytes at file offset 1048576 = 4096 * 256 (i.e., reading page 255 from a file assuming 4KB pages)

File System Interface

- POSIX I/O interface
 - A standard synchronous I/O interface
 - Agnostic to the underlying storage device/file system

`open(2)`: open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

`pread(2)`, `pwrite(2)`: read from or write to a file descriptor at a given offset

`posix_fallocate(3)`, `fallocate(2)`

`fsync(2)`, `fdatasync(2)`,

`close(2)`

Check man pages for details (e.g., [Linux man pages online \(man7.org\)](https://www.man7.org/linux/), or [Linux man pages \(die.net\)](https://www.die.net/linux/))

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.