ECS 165B: Database System Implementation

UC Davis, Spring 2010

Acknowledgements: design of course project for this class borrowed from CS 346 @ Stanford's RedBase project, developed by Jennifer Widom, and used with permission. Slides based on earlier ones by Raghu Ramakrishnan, Johannes Gehrke, Jennifer Widom, Bertram Ludaescher, and Michael Gertz.

Welcome to ECS 165B!

Agenda for today's class:

- Logistics and course overview
- Introduction to the DavisDB project
- Technical material: pages, files, buffers, records (Chapter 9 of textbook)

Course Logistics

Instructor:

Prof. Todd J. Green (green@cs.ucdavis.edu)

Office hours: Tuesdays, 11:00-11:50am, 3055 Kemper Hall

Teaching assistant:

Mingmin Chen (michen@ucdavis.edu)

Office hours: Wednesdays, 11:00-11:50am, 053 Kemper Hall

Meeting times:

MWF 4:10-5:00pm (1 Wellman Hall)

Discussion section Fridays 11:00-11:50am (1 Wellman Hall)

More Logistics

Course webpage:

http://www.cs.ucdavis.edu/~green/courses/ecs165b

Class mailing list:

ecs165b-s10@ucdavis.edu

Anyone in the class can post! Don't be shy!

Textbook (optional):

Database Management Systems, 3rd Edition, Ramakrishnan and Gehrke, McGraw Hill, 2003

What's This Course About?

ECS 165A (last quarter):

how to **use** a DBMS

ECS 165B (this quarter):

how to **build** a DBMS

What's This Course About?

Primary focus (new this year!): quarter-long implementation project

- You will build major components of a (simplified) relational database system, **DavisDB**, in C++
- In teams of 2, delivered in 5 stages

Secondary focus: a sampler of further topics in databases

- XML and semistructured data, data warehousing, …
- A taste of database theory

Meta-focus: large-scale software engineering (debugging, revision control systems, best coding practices, …)

How Will This Course Be Graded?

Basic formula: project 80% (in 5 parts), closed-book quizzes 20% (2 of them)

No midterm, no final…

…but this will be a difficult, time-consuming class!

Code graded for correctness, efficiency, and style

Extra credit for winners of **DavisDB I/O efficiency contest**, as well as the **DavisDB code beauty contest**

Should I Take This Class?

Pre-requisites:

- DBMS fundamentals (ECS 165A)
- C/C++ programming and data structures (ECS 60)
- Ability to work independently
- Time, ingenuity, and a sense of humor ☺ (This class is a beta-version!)

What you'll get out of the class:

- Deeper and broader knowledge of DBMS
- Software engineering experience that will pay off once you enter the real world
- Images of the CSIF lab's soul-crushing mountain scenery posters forever burned into your retinas

Forming Teams

The project will be done in teams of 2.*

Choose your partner carefully! Your grades for the project will be **identical.** It's up to you to figure out how to share the work and get along. No marriage counseling provided.**

Send an email to the TA by **Wednesday** with your preference for a project partner (or "no preference" if you have none).

*If you prefer to work alone, you may do so, but you will still be responsible for the same work as the teams, and no special allowance will be made in grading.

**Divorces may be granted on a case-by-case basis.

Some Project Logistics

Team members will coordinate their efforts, and submit their code, via **subversion** (a standard revision control system)

A short (1-2 page), high-level **writeup** will be part of the submitted work

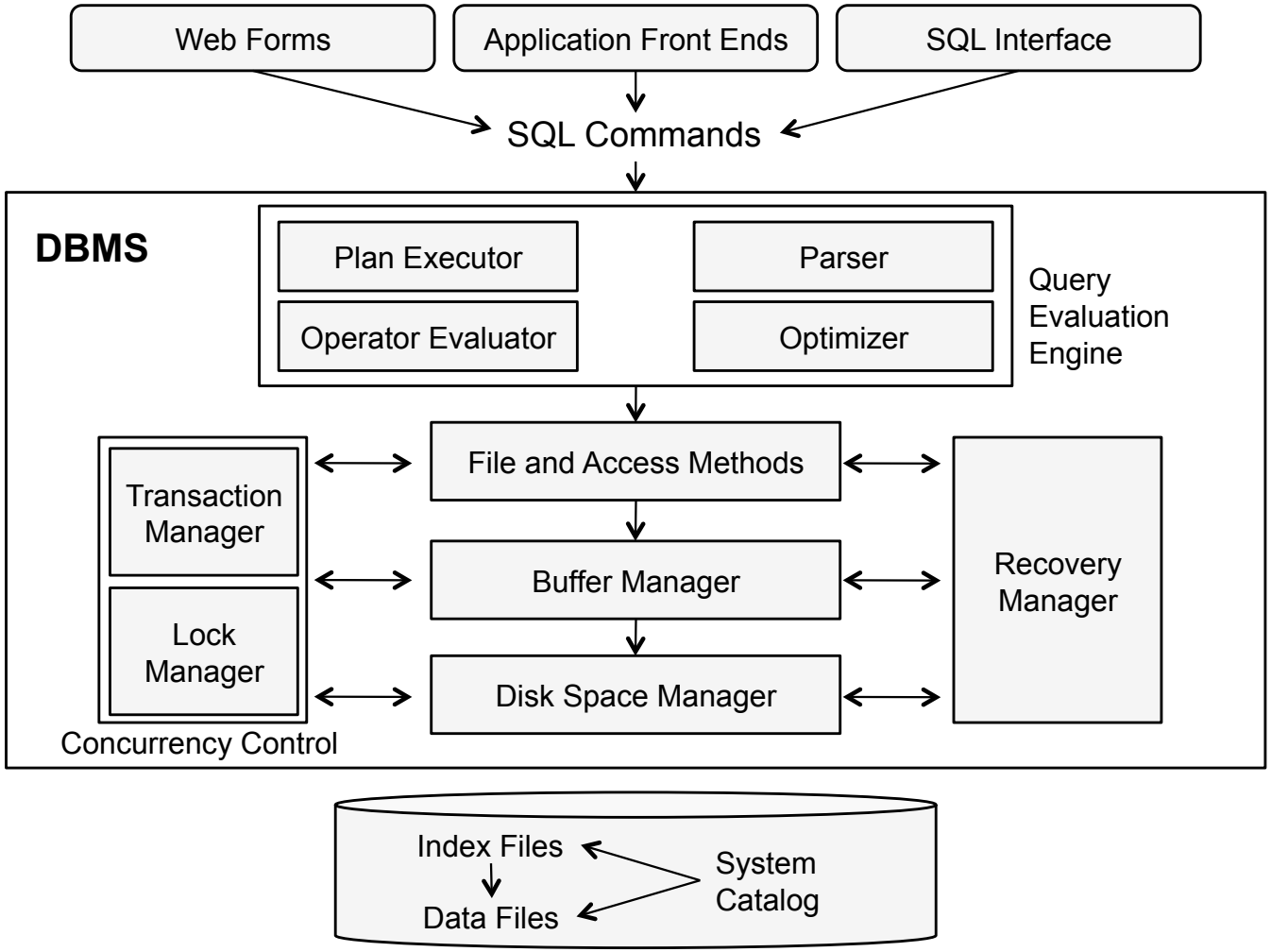
Standard platform: the CSIF Linux machines

Automated testing for correctness (~80% of score), manual grading of writeup, design, and code style (~20% of score)

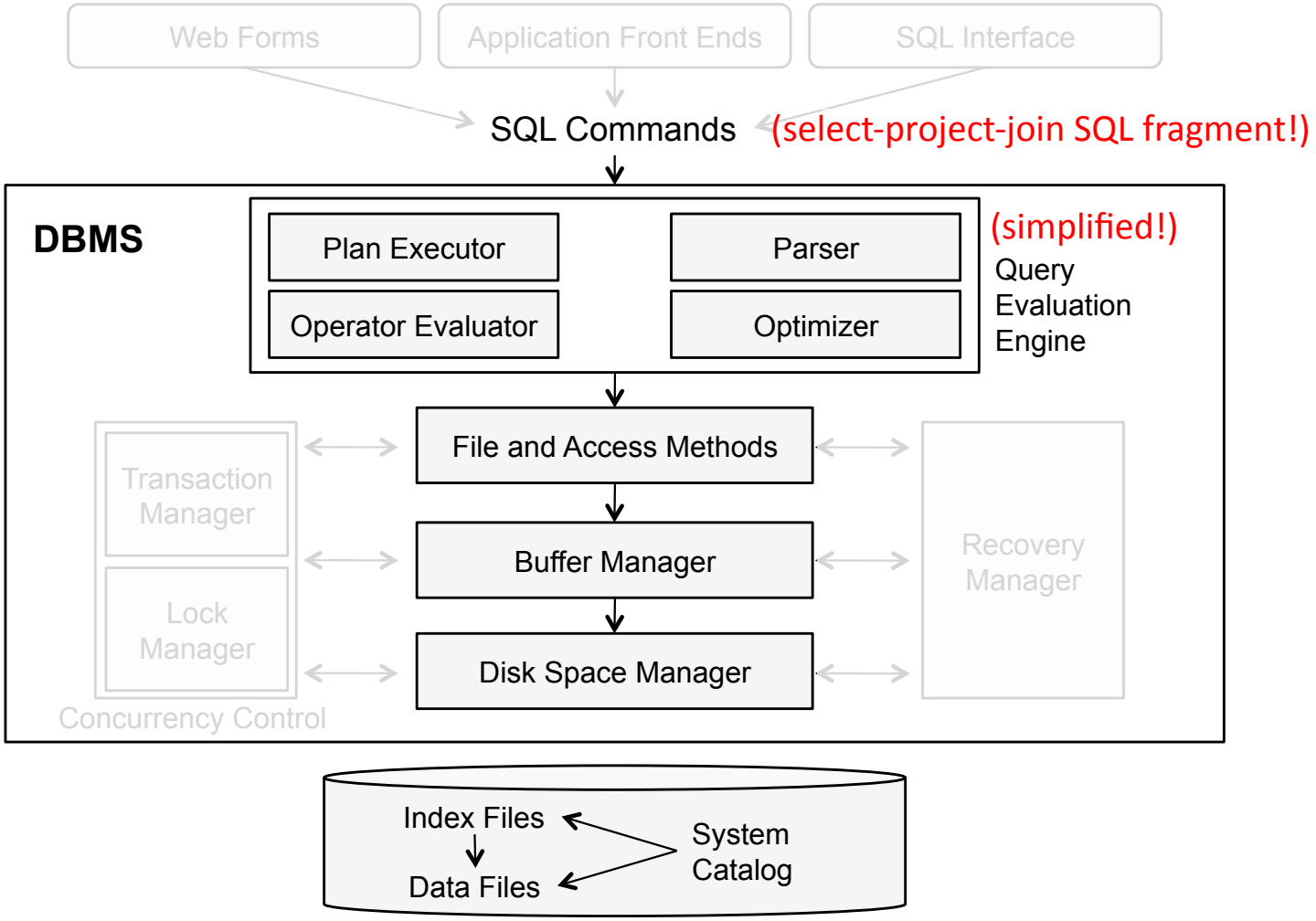
We'll emphasize fundamental skills, such as the proper use of a **debugger**. (printf won't cut it in this class, just as it doesn't in the real world.)

More on the logistics next time…

Review: Basic DBMS Architecture

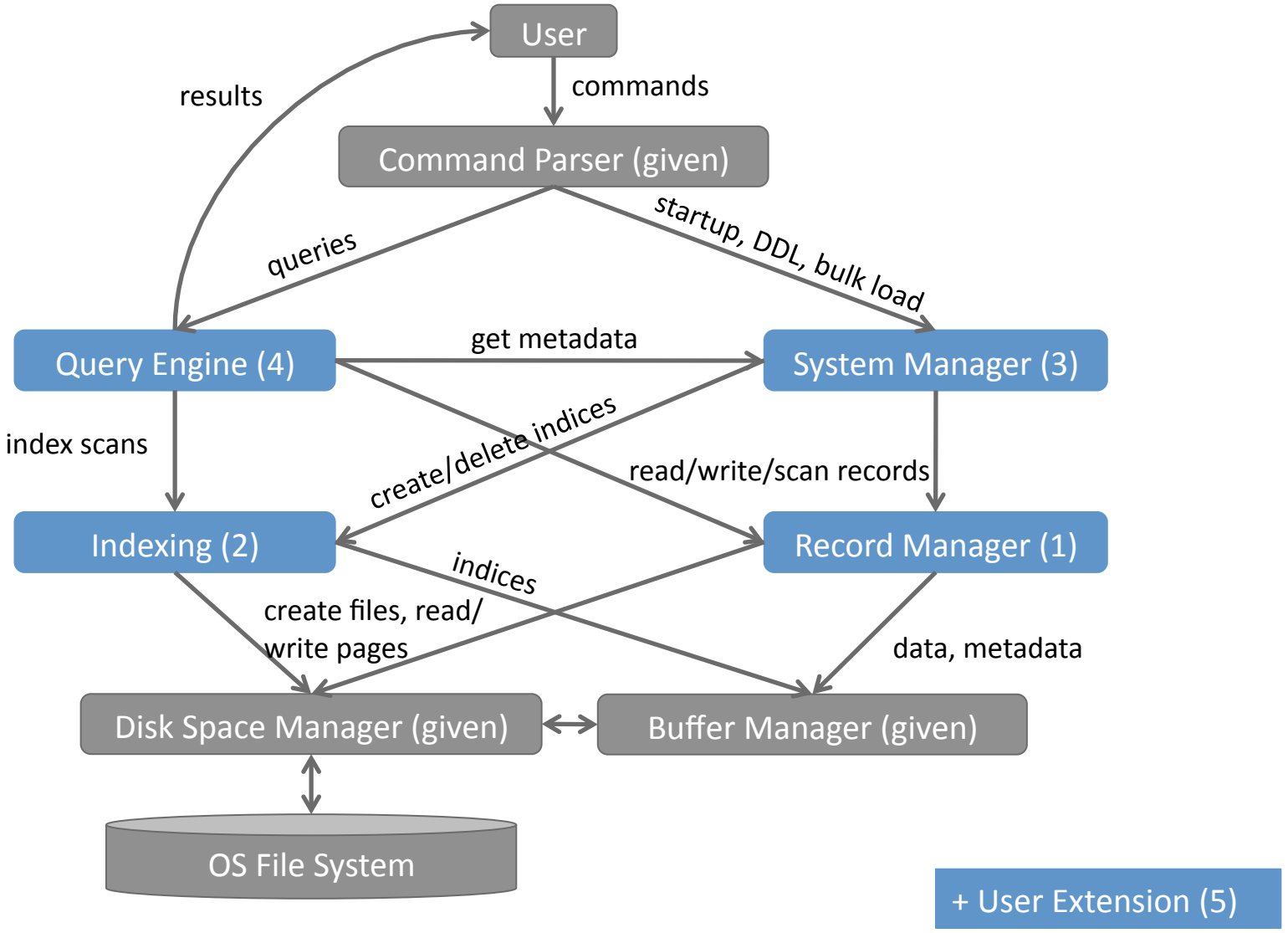


DavisDB Architecture (What's Left Out)



+ user-defined extension

Major Components of DavisDB



Important Dates

Project due dates, subject to change:

- Part 1 (record manager): 4/11
- Part 2 (indexing): 4/25
- Part 3 (system manager): 5/2
- Part 4 (query engine): 5/23
- Part 5 (user extension): proposal due 5/16, code due 6/6

Quizzes:

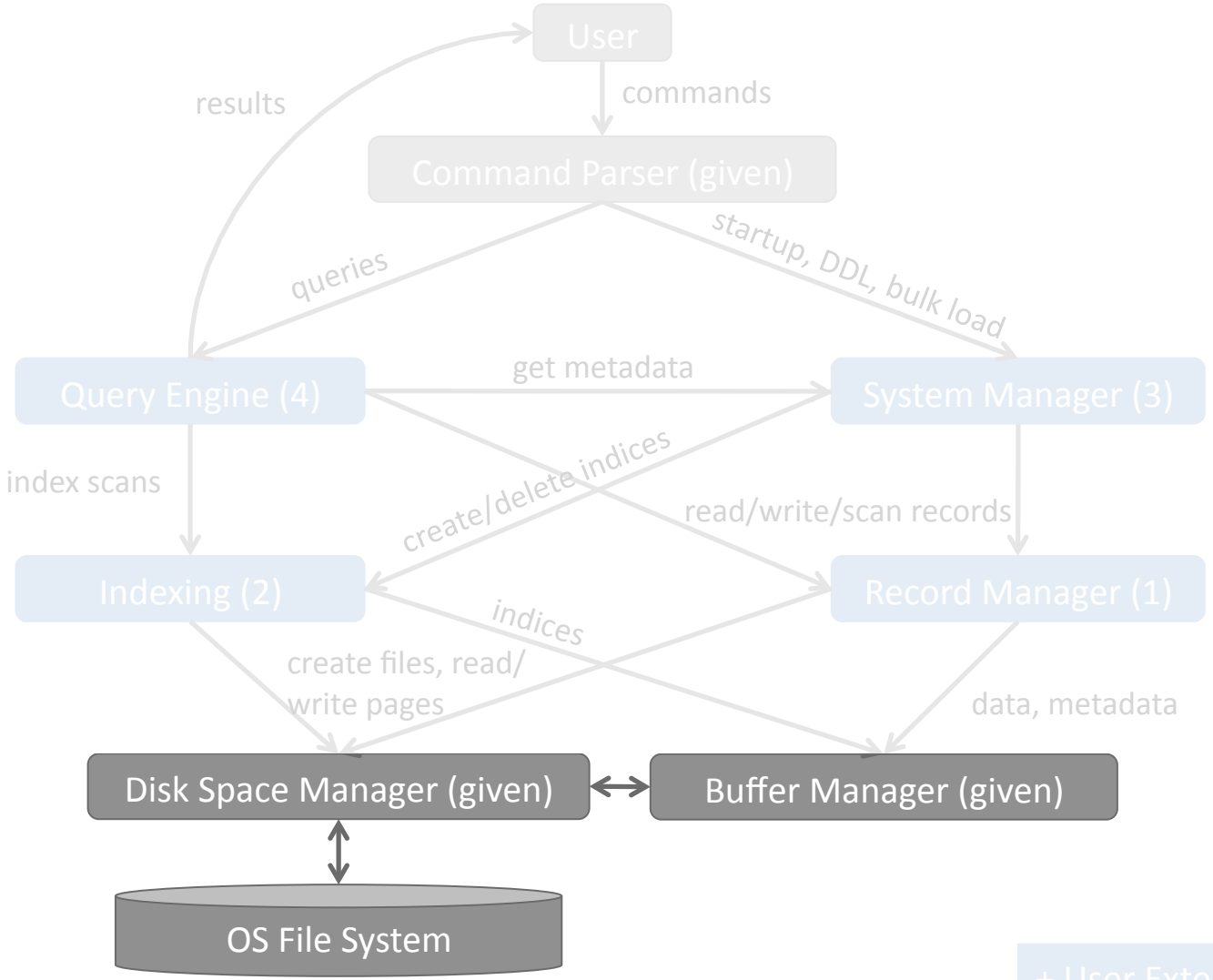
- Quiz #1: 5/5
- Quiz #2: 6/2

Also:

- Mid-quarter course review: 5/7

File and Buffer Management Review

File and Buffer Management Review



+ User Extension (5)

Disks and Files

- (Traditional) DBMS stores information on hard disks
- This has major implications for DBMS design!
  - READ: transfer data from disk to memory (RAM)
  - WRITE: transfer data from RAM to disk
  - Both are high-cost operations, relative to in-memory operations, so must be planned carefully!
  - DavisDB I/O efficiency contest: minimize total READS and WRITES

Why Not Store Everything in Main Memory?

- Traditional arguments:
  - *It costs too much*. In 1995, $1000 would buy you either 128MB of RAM or 7.5GB of disk.
  - *Main memory is volatile*. We want data to be saved between runs. (Obviously!)
- Traditional storage hierarchy:
  - Main memory (RAM) for currently-used data
  - Disk for the main database (secondary storage)
  - Tapes for archiving older versions of the data (tertiary storage)
- DavisDB follows traditional model (minus the tapes ☺
- Discussion: do the traditional arguments still hold water?

Disks and Paged Files

- Secondary storage device of choice
- Main advantage over tapes: *random access* versus *sequential*
- Data on hard disks is stored and retrieved in units called *disk blocks* or (as we'll term them in DavisDB) *pages*
- Unlike RAM, time to retrieve a disk page varies depending upon location on disk…
- …therefore, relative placement of pages on disk has major impact on DBMS performance!
  - For simplicity, we'll overlook this in DavisDB
- File is organized as a sequence of pages

Buffer Management

- Main memory is limited
- Pages of disk files move in/out of in-memory *buffer pool*
- DavisDB # pages in buffer pool: 40
- Total buffer size (@4K pages): 160K -- tiny!