

Introduction to Database Systems

CS4320

Instructor: Christoph Koch
koch@cs.cornell.edu

CS4320/1: Introduction to Database Systems

Underlying theme: How do I build a data management system?

- CS4320 will deal with the underlying *concepts*
 - No programming assignments
- CS4322 will be the *practicum*
 - Build components of a small search engine (C++ programming)

CS4320/1: Introduction to Database Systems

Information is one of the most valuable resources in this information age.

- How do we effectively and efficiently manage this information?
- Relational database management systems
 - Dominant data management paradigm today
- Search engines
 - Ubiquitous today
- 100+ billion dollar a year industry
 - You will see this in the job market!

RDBMS Market

Worldwide 2005 Vendor Revenue Estimates from RDBMS Software, Based on Total Software Revenue (Millions of Dollars)

Company	2005	2005 Market Share (%)	2004	2004 Market Share (%)	2004-2005 Growth (%)
Oracle	6,721.1	48.6	6,234.1	48.9	7.8
IBM	3,040.7	22.0	2,860.4	22.4	6.3
Microsoft	2,073.2	15.0	1,777.9	13.9	16.6
Teradata	440.7	3.2	412.1	3.2	6.9
Sybase	407.0	2.9	382.8	3.0	6.3
Other Vendors	1,134.7	8.2	1,090.4	8.5	4.1
Total	13,817.4	100.0	12,757.8	100.0	8.3

Source: Gartner Dataquest (May 2006)



World's largest software companies

From Wikipedia, the free encyclopedia

The [Forbes Global 2000](#) includes the following list of the **world's largest software companies**.

Relative rank	Global rank	Name	Country	Sales (\$bil)	Profits (\$bil)	Assets (\$bil)	Market Value (\$bil)
1	54	Microsoft	United States	41.36	13.06	67.26	279.02
2	240	Oracle	United States	12.89	2.88	19.35	64.01
3	248	First Data	United States	10.49	1.59	34.25	34.43
4	381	SAP	Germany	10.06	1.77	10.43	63.10
5	418	Accenture	Bermuda	17.57	0.96	8.12	27.77
6	439	Google	United States	6.14	1.47	10.27	107.17
7	473	Yahoo!	United States	5.26	1.90	10.87	45.48
8	499	Computer Sciences Corporation	United States	14.61	0.85	12.62	10.12
9	715	Electronic Data Systems	United States	19.76	0.15	17.09	13.84
10	776	SoftBank	Japan	7.81	-0.56	15.53	32.78
11	948	Symantec	United States	3.62	0.16	17.68	17.60
12	959	CA	United States	3.76	0.22	10.06	15.73
13	993	Fiserv	United States	4.06	0.52	6.04	7.70
14	1010	Affiliated Computer Services	United States	4.94	0.42	5.31	7.86
15	1077	Adobe Systems	United States	1.97	0.60	2.44	23.12
16	1086	Capgemini Group	France	8.22	0.17	7.15	6.50

navigation

- [Main page](#)
- [Contents](#)
- [Featured content](#)
- [Current events](#)
- [Random article](#)

interaction

- [About Wikipedia](#)
- [Community portal](#)
- [Recent changes](#)
- [Contact us](#)
- [Make a donation](#)
- [Help](#)

search

toolbox

- [What links here](#)
- [Related changes](#)
- [Upload file](#)
- [Special pages](#)
- [Printable version](#)
- [Permanent link](#)
- [Cite this article](#)

„Note: The above list does not include companies like IBM whose software/services part is bigger than Microsoft. In the Forbes2000 report IBM and HP were listed as Technology Hardware companies.“

From the IBM 2006 Annual Report

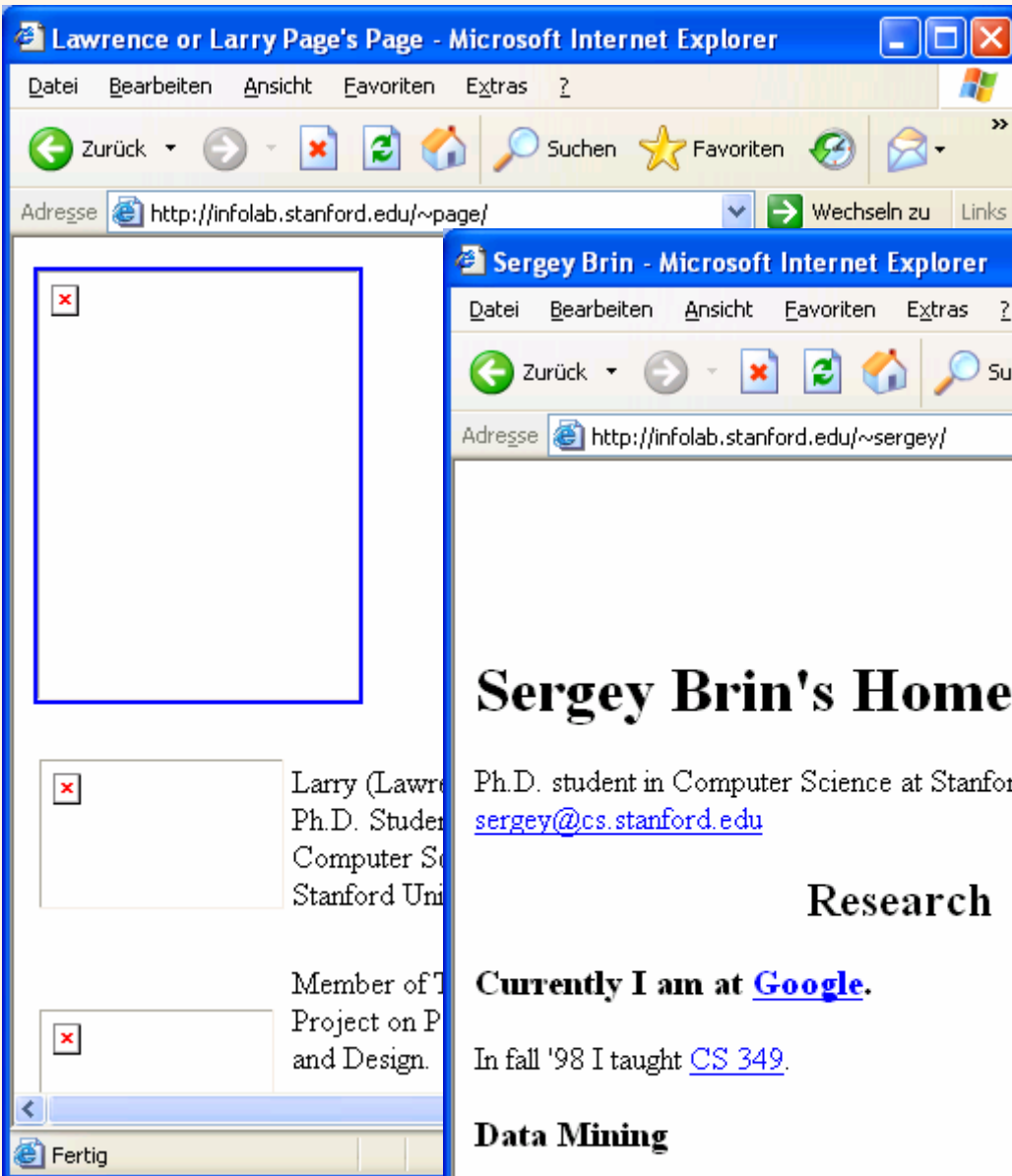
YEAR IN REVIEW

RESULTS OF CONTINUING OPERATIONS

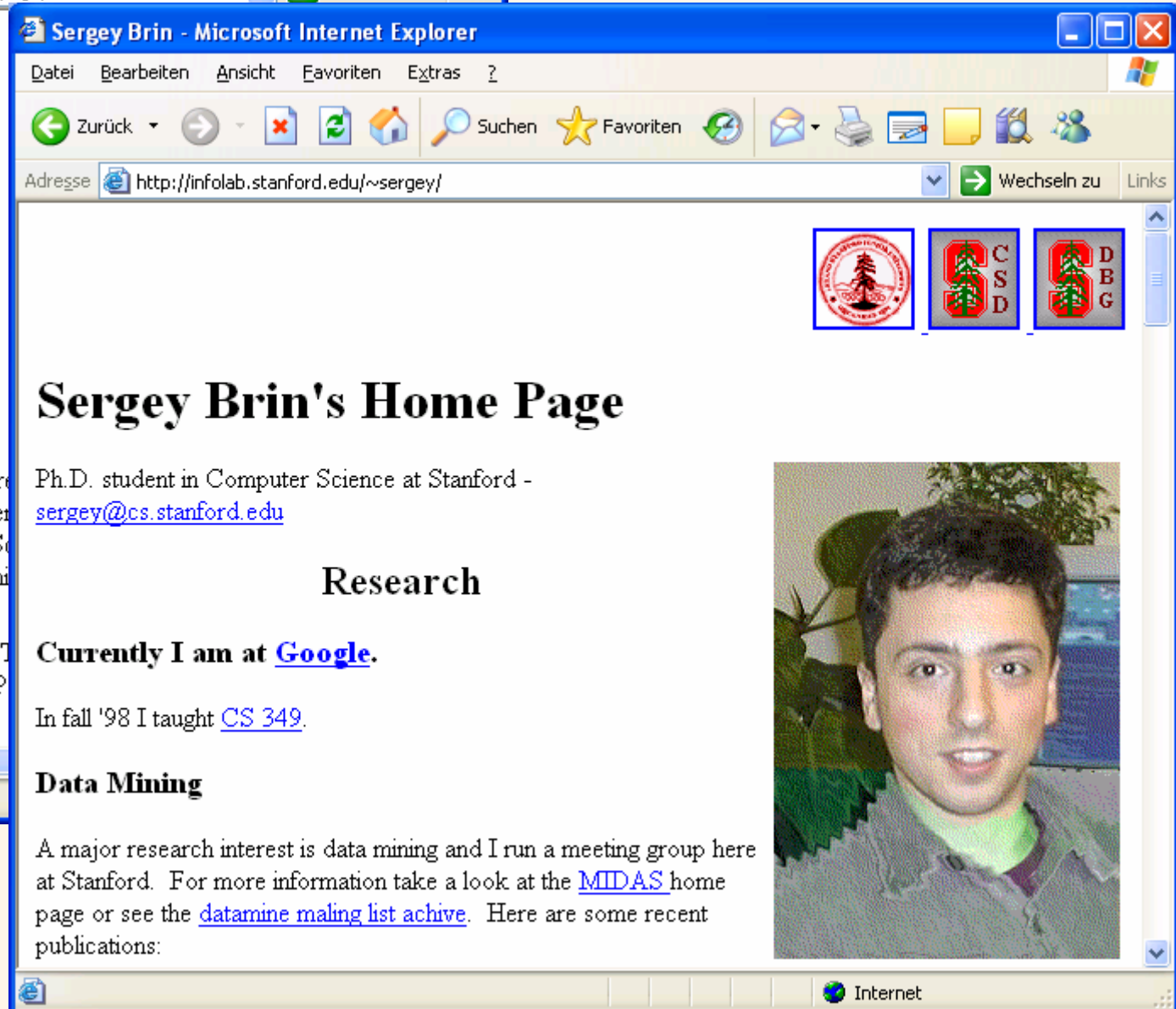
Revenue

(Dollars in millions)

	2006
FOR THE YEAR ENDED DECEMBER 31:	
Statement of Earnings Revenue Presentation:	
Global Services	\$48,247
Hardware	22,499
Software	18,204
Global Financing	2,379
Other	94
Total	\$91,424



Founders of Google
(DB PhD students)



CS4320 Prerequisites

Courses

- CS212 (Computers and Programming)
- CS312 (Structure and Interpretation of Computer Programs)

People

- Instructor:
 - Christoph Koch
- TAs:
 - Shuang Zhao
 - Guozhang Wang

Access to Instructor and TAs

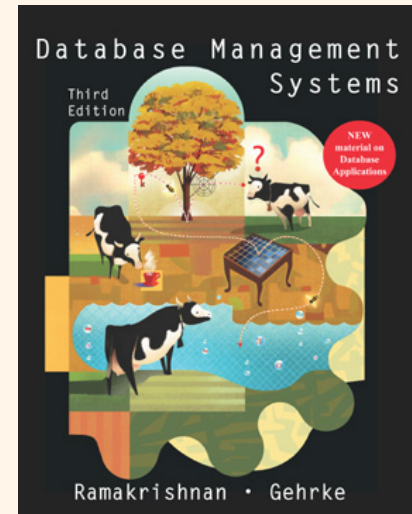
- Office hours
 - Posted on course web site
<http://www.cs.cornell.edu/courses/cs4320/>
- TA mailing list
 - cs4320ta-l@cs.cornell.edu
 - cs4321ta-l@cs.cornell.edu
 - Do not directly email TAs

Course Structure

- Two components
 - Assignments (50%)
 - Five assignments
 - Each assignment worth 10% of total grade.
 - Two examinations (50%)
- No programming assignments in CS4320
 - CS4321 will have all programming assignments

Textbook

- Textbook: “Database Management Systems” (3rd Edition)
 - By R. Ramakrishnan and J. Gehrke
 - Required textbook
- Syllabus
 - Defined by class lectures
 - Not defined by textbook



Assignment Policies

- Assignments have to be done individually
 - No collaboration with others
- Academic integrity violations taken VERY seriously
 - Read Cornell and CS academic integrity policies
 - Available off course web page
 - Need to sign and hand in form
- Course management system used to post assignment grades

Assignment Policies (ctd.)

- Late submissions
 - Grace period of 48 hours during which you can still achieve 90% of the full score.
 - After that: 0% of grade for even later submissions
 - No exceptions (assignments handed out well in advance of deadline)
- Regrade requests
 - Within 7 days after assignments are graded
 - Hard deadline

Exams

- Mid-term exam (20%)
 - 23 October 2008, 7:30-10:00pm
 - Closed book exam
- Final exam (30%)
 - Date TBA
 - Closed book exam
 - Cumulative with emphasis on second half
- Do not schedule other exams or interviews on these days

Relationship to CS4321

- CS4320 is about *concepts* underlying databases
 - No programming assignments
- CS4321 is the *practicum* associated with CS4320
 - Will actually build the core of a “realistic” database management system.
 - C++ programming
- Complementary
 - Suggest that you take both
 - **Can** take CS4320 without taking CS4321
 - **Cannot** take CS4321 without taking CS4320

Is CS4320/4321 a lot of work?

- It depends!
 - Much of the material in CS4320 is probably new to you
 - CS4321 has substantial programming assignments
- Then why on earth should I take this course?
 - Intellectual argument
 - Big conceptual ideas
 - Meeting of theory and practice
 - Utilitarian argument
 - Many, many real applications (data management, data-driven websites, search engines,...)
 - Job market!

Reminder

- Complete academic integrity form
(download from course homepage)
- Hand in this week in class!

What Is a DBMS?

- A very large, integrated collection of data.
- Models real-world *enterprise*.
 - Entities (e.g., students, courses)
 - Relationships (e.g., Madonna is taking CS564)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Why Use a DBMS?



- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Why Study Databases??



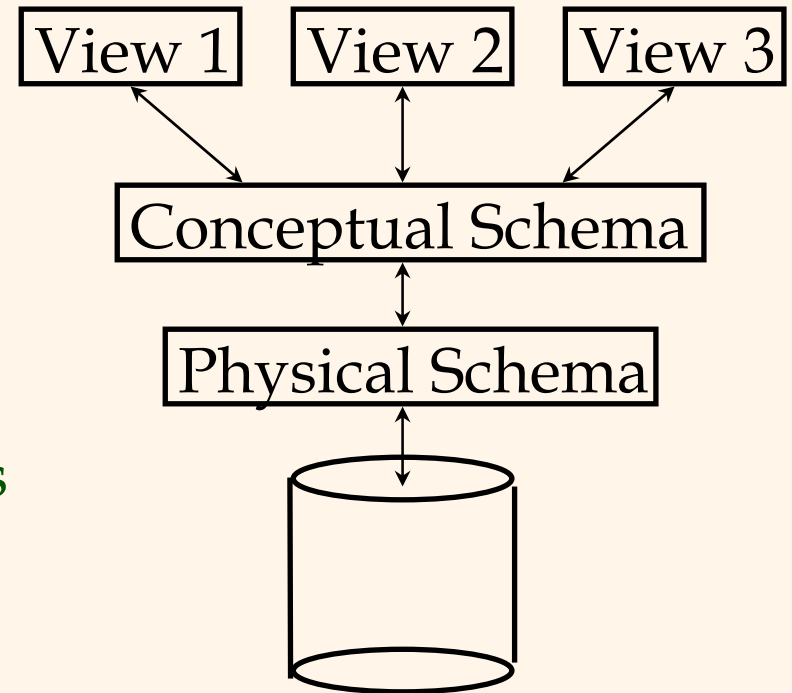
- Shift from computation to information
 - at the “low end”: scramble to webspace (a mess!)
 - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
 - Digital libraries, interactive video, Human Genome project, EOS project
 - ... need for DBMS exploding
- DBMS encompasses most of CS
 - OS, languages, theory, AI, multimedia, logic

Data Models

- A data model is a collection of concepts for describing data.
- A schema is a description of a particular collection of data, using the a given data model.
- The relational model of data is the most widely used model today.
 - Main concept: relation, basically a table with rows and columns.
 - Every relation has a schema, which describes the columns, or fields.

Levels of Abstraction

- Many views, single conceptual (logical) schema and physical schema.
 - Views describe how users see the data.
 - Conceptual schema defines logical structure
 - Physical schema describes the files and indexes used.



* Schemas are defined using DDL; data is modified/queried using DML.

Example: University Database

- Conceptual schema:
 - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
 - *Courses(cid: string, cname:string, credits:integer)*
 - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
 - Relations stored as unordered files.
 - Index on first column of Students.
- External Schema (View):
 - *Course_info(cid:string,enrollment:integer)*

*Data Independence **

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

** One of the most important benefits of using a DBMS!*

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
 - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

- Key concept is transaction, which is an *atomic* sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
 - Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints.
 - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
 - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the *user's* responsibility!

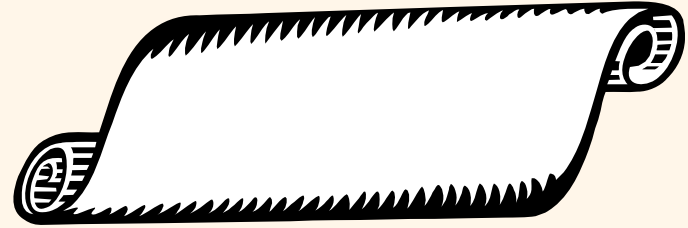
Scheduling Concurrent Transactions

- DBMS ensures that execution of $\{T_1, \dots, T_n\}$ is equivalent to some serial execution $T_1' \dots T_n'$.
 - Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)
 - **Idea:** If an action of T_i (say, writing X) affects T_j (which perhaps reads X), one of them, say T_i , will obtain the lock on X first and T_j is forced to wait until T_i completes; this effectively orders the transactions.
 - What if T_j already has a lock on Y and T_i later requests a lock on Y ? (Deadlock!) T_i or T_j is aborted and restarted!

Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.
- **Idea:** Keep a log (history) of all actions carried out by the DBMS while executing a set of Xacts:
 - **Before** a change is made to the database, the corresponding log entry is forced to a safe location. (WAL protocol; OS support for this is often inadequate.)
 - After a crash, the effects of partially executed transactions are undone using the log. (Thanks to WAL, if log entry wasn't saved before the crash, corresponding change was not applied to database!)

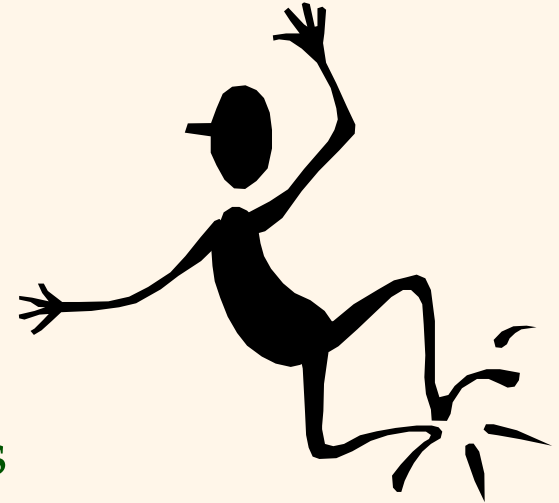
The Log



- The following actions are recorded in the log:
 - *Ti writes an object*: The old value and the new value.
 - Log record must go to disk before the changed page!
 - *Ti commits/aborts*: A log record indicating this action.
- Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- Log is often *duplexed* and *archived* on “stable” storage.
- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
 - E.g., smart webmasters
- *Database administrator (DBA)*
 - Designs logical / physical schemas
 - Handles security and authorization
 - Data availability, crash recovery
 - Database tuning as needs evolve

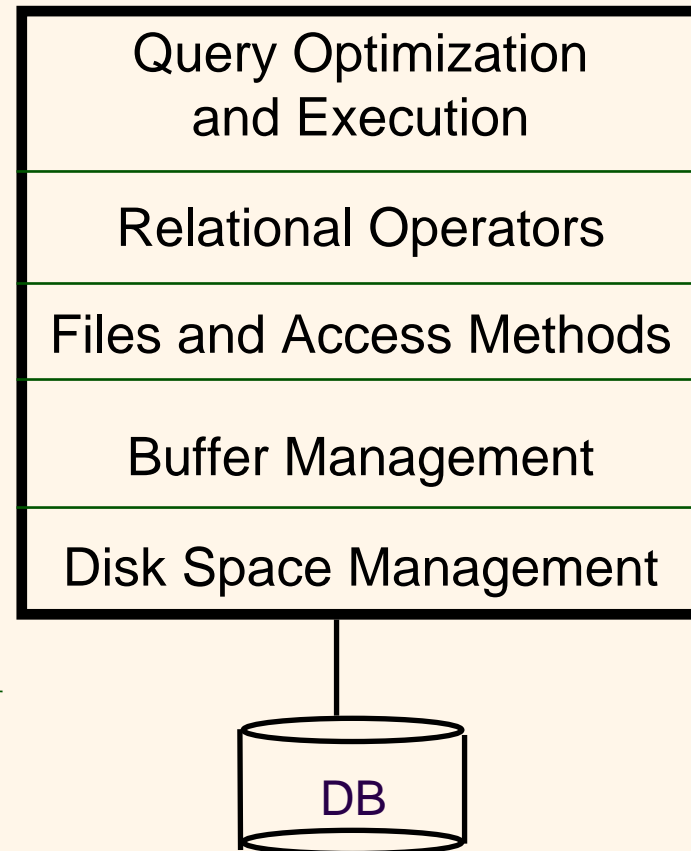


Must understand how a DBMS works!

Structure of a DBMS

These layers must consider concurrency control and recovery

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.



Summary

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs hold responsible jobs and are **well-paid!** 😊
- DBMS R&D is one of the broadest, most exciting areas in CS.

