CSE 4020/5260
Database Systems

Instructor: Fitzroy Nembhard, Ph.D.

Week 1

Introduction:
Intro to Database Systems

FLORIDA TECH
FLORIDA'S **STEM** UNIVERSITY®

Distribution

- All slides included in this class are for the exclusive use of students and instructors associated with Database Systems (CSE 4020/5260) at the Florida Institute of Technology
- Redistribution of the slides is not permitted without the written consent of the author.

Introduction to Database Systems

- Databases
- Database Management Systems (DBMS)
- Levels of Abstraction
- Data Models
- Database Languages
- Types of Users
- DBMS Function and Structure

In other words, a somewhat random list of words and concepts that are necessary to move on…

Read Chapter 1, including the historical notes on pages 25 - 28.

Concept #1: Databases & Database Management Systems

What is a Database?

- According to the book:
  - Database is a collection of data
  - DBMS is a collection of interrelated data and a set of programs to access the data
  - A DBMS contains information about a particular enterprise
  - DBMS provides an environment that is both *convenient* and *efficient* to use.
- Another definition (know these):
  - A *database* is a collection of organized, interrelated data, typically relating to a particular enterprise
  - A *Database Management System* (DBMS) is a set of programs for managing and accessing databases

Some Popular Database Management Systems

- Commercial “off-the-shelf” (COTS):
  - Oracle
  - IBM DB2 (IBM)
  - SQL Server (Microsoft)
  - Sybase (now SAP)
  - Informix (IBM)
  - Access (Microsoft)
  - Caché (by Intersystems – object and relational)
- Open Source:
  - MySQL or MariaDB
  - PostgreSQL

Note: The theory in this course is <u>not</u> on any particular DBMS!

Some Database Applications

- Anywhere there is data, there could be a database:
  - Banking - accounts, loans, customers
  - Airlines - reservations, schedules
  - Universities - registration, grades
  - Sales - customers, products, purchases
  - Manufacturing - production, inventory, orders, supply chain
  - Human resources - employee records, salaries, tax deductions
- Course context is an “enterprise” that has requirements for:
  - Storage and management of 100s of gigabytes or terabytes of data
  - Support for 100s or more concurrent users and transactions
  - Traditional supporting platform, e.g., Dell PowerEdge R720xd, 68 processors, 16GB RAM each, 50TB of disk space

Purpose of Database System

- Prior to the availability of COTS DBMSs, database applications were built on top of file systems – coded from the ground up.

- Drawbacks of this approach:
  - Difficult to reprogram sophisticated processing, i.e., concurrency control, backup and recovery, security
  - Re-inventing the wheel can be expensive and error-prone.
  - "We need a truck, lets design and build our own truck."***
- According to the book, this leads to:
  - Data redundancy and inconsistency
  - Multiple files and formats
  - A new program to carry out each new task
  - Integrity constraints (e.g., account balance > 0) become embedded throughout program code, etc.

- Database systems offer proven solutions for the above problems.

Purpose of Database Systems (Cont.)

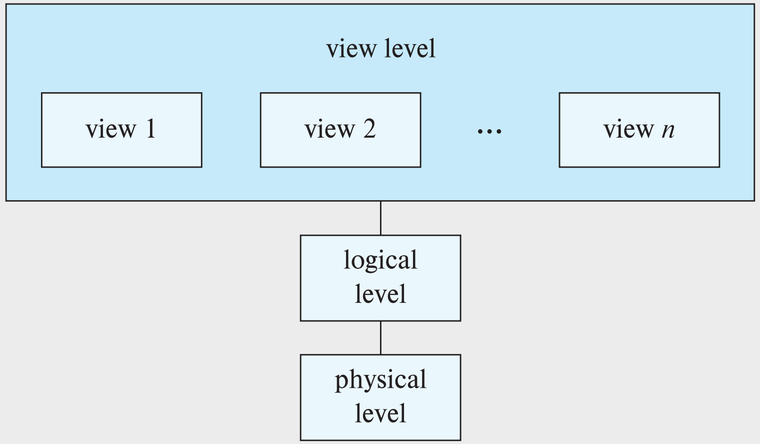
- Even to this day, engineers will occasionally propose custom-developed file systems.

- So, when should we code from scratch, and when do we buy a DBMS??
  - How much data?
  - How sophisticated is the processing of that data?
  - How many concurrent users?
  - What level of security?
  - Is data integrity an issue?
  - Does the data change at all?

Concept #2: Levels of Abstraction

Levels of Abstraction

- Physical level - defines low-level details about how data item is stored on disk.
- Logical level - describes data stored in a database, and the relationships among the data (usually conveyed as a data model, e.g., an ER diagram).
- View level - defines how information is presented to users. Views can also hide details of data types, and information (e.g., salary) for security purposes.



Levels of Abstraction

- *Physical data independence* is the ability to modify the physical schema without having an impact on the logical or view levels.

- Physical data independence is important in any database or DBMS.

Example of Physical Data Independence

CANVAS

Canvas Status Update

Higher Education Studio
30 pt. Headline

Scheduled maintenance on
October 30, 2021** at **12:00 AM MDT / 6:00 AM UTC.

Hi Admins—

We'll be performing a brief maintenance event on your **Roll Call Attendance** instance on **October 30, 2021.**

Maintenance Details:
Changing an index in the database on a large table

This event will occur at **12:00 AM MDT / 6:00 AM UTC**. Your users will not be able to access **Roll Call Attendance** for up to **2 Hours** while the event is underway.

Please contact your customer success manager if you have any questions or concerns.

Thanks,

Instructure Team

CTA Text ➔ CTA Text ➔

Concept #3: Instances vs. Schemas

Instances vs. Schemas

- The difference between a *database schema* and a *database instance* is similar to the difference between a data type and a variable in a program.

- A database *schema* defines the structure or design of a database.

- More precisely:
  - A *logical* schema defines a database design at the logical level; typically, an entity-relationship (ER) or UML diagram.
  - A *physical* schema defines a database design at the physical level; typically, a DDL file.

- An *instance* of a database is the combination of the database and its contents at one point in time.

Concept #4: Data Models

What is a Data Model?

- **According to the book:**
  - A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints
- The phrase *"data model"* is used in a couple of different ways.
- Frequently used (use #1) to refer to an overall approach or philosophy for database design and development.
- For those individuals, groups and corporations that subscribe to a specific data model, that model permeates all aspects of database design, development, implementation, etc.

What is a Data Model?

- Common data models:
  - Relational model
  - Object-oriented model
  - Object-relational model
  - Semi, and non-structured data models (XML)
  - Various other NoSQL models (graph, document, key/value)
- Legacy data models:
  - Network
  - Hierarchical

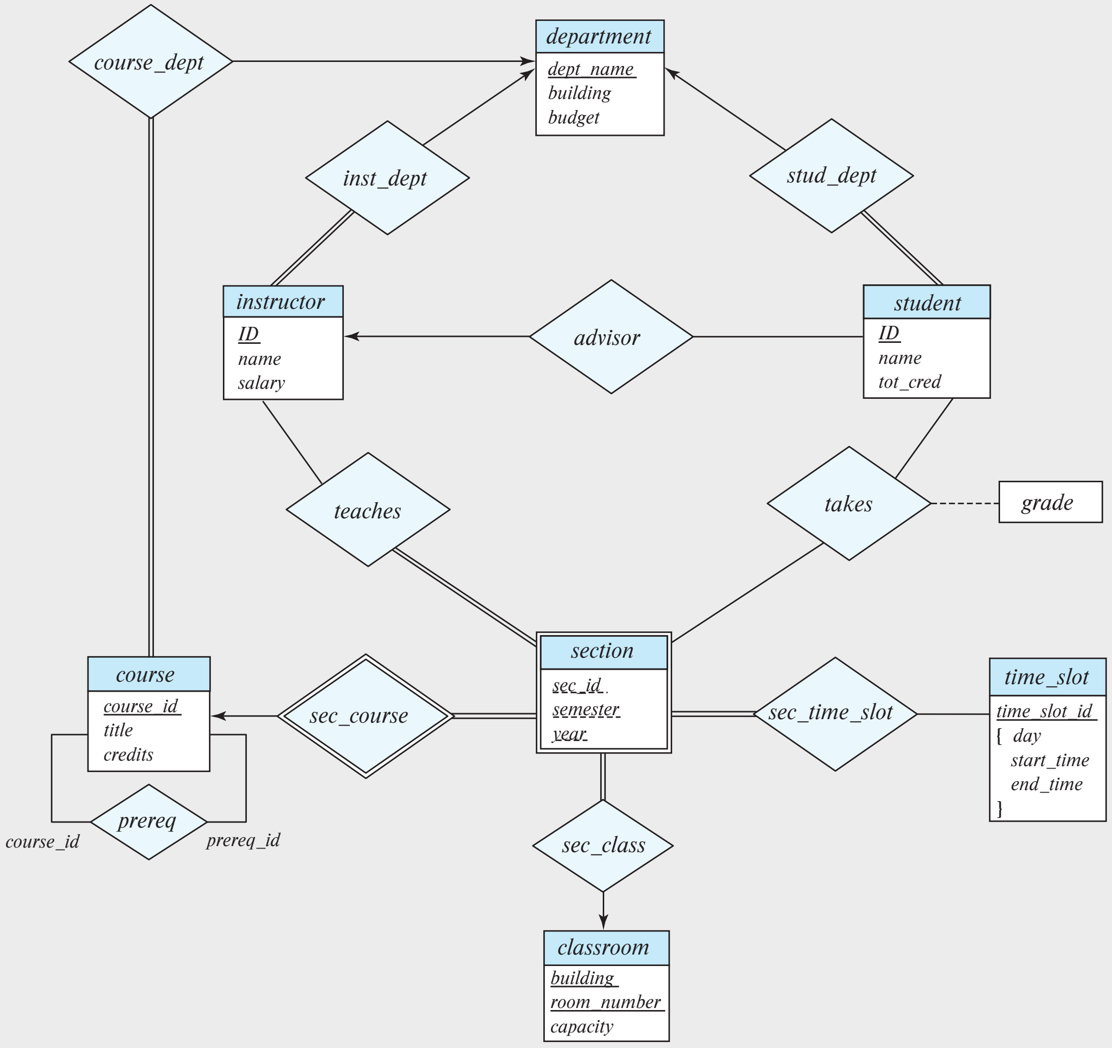
What is a Data Model, Cont?

- During the early phases of database design and development, a *"data model"* is frequently developed (**use #2**).
- The purpose of developing the data model is to define:
  - Data
  - Relationships between data items
  - Semantics of data items
  - Constraints on data items

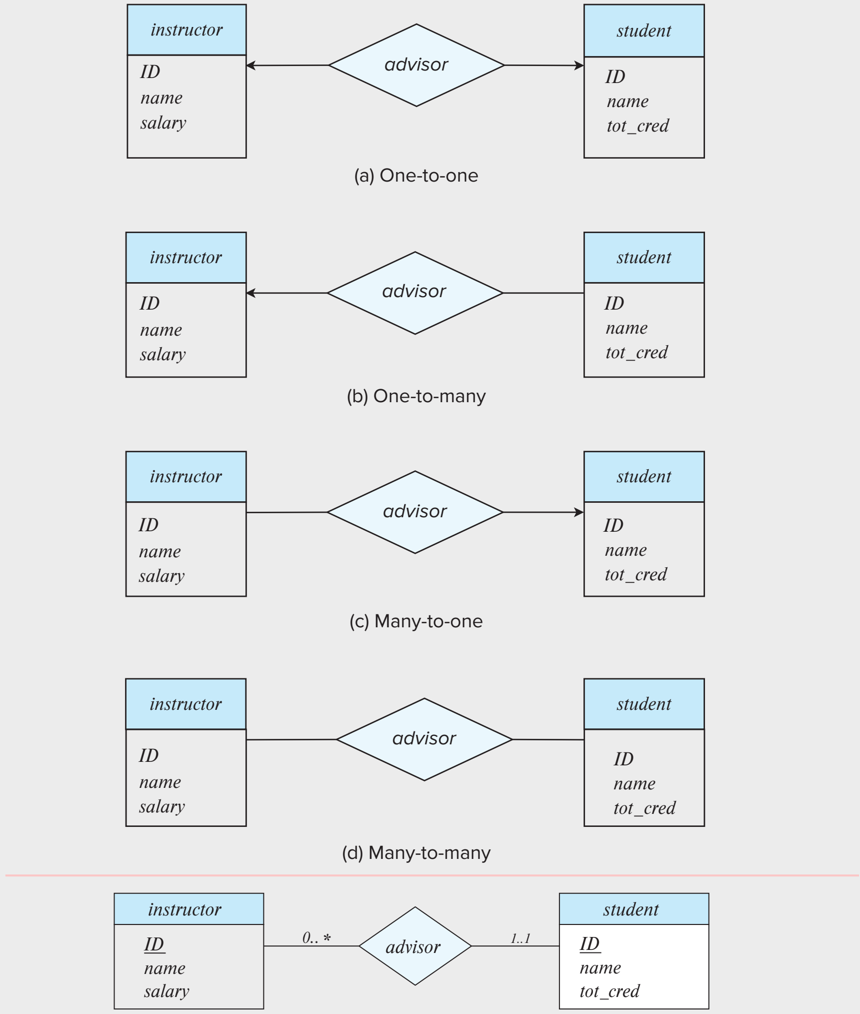
  In other words, a data model defines the logical schema, i.e., the logical level of design of a database.

- A data model is typically conveyed as one or more diagrams (e.g., ER or UML diagrams).
- This early phase in database development is referred to as *data modeling*.

Entity-Relationship Diagrams



Cardinalities



A Sample Relational Database

university schema

classroom(<u>*building*</u>, <u>*room_number*</u>, *capacity*)
department(<u>*dept_name*</u>, *building*, *budget*)
course(<u>*course_id*</u>, *title*, *dept_name*, *credits*)
instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)
section(<u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *building*, *room_number*, *time_slot_id*)
teaches(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)
student(<u>*ID*</u>, *name*, *dept_name*, *tot_cred*)
takes(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *grade*)
advisor(<u>*s_ID*</u>, *i_ID*)
time_slot(<u>*time_slot_id*</u>, <u>*day*</u>, <u>*start_time*</u>, *end_time*)
prereq(<u>*course_id*</u>, <u>*prereq_id*</u>)

course

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

section

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

prereq

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Concept #5: Query Languages

Query Languages

- A *query language* is used to create, manage, access, and modify data in a database.

- The list of query languages is quite long:
  - http://en.wikipedia.org/wiki/Query_languages

- The most widely used query language is *Structure Query Language (SQL).*

- At a high-level, SQL consists of two parts:
  - *Data Definition Language* (DDL)
  - *Data Manipulation Language* (DML)

Data Definition Language (DDL)

- DDL is used for defining a (physical) database schema (see the book for a more complete example):

```
create table account (
    account-number          char(10),
    branch-name             varchar(16),
    balance                 integer,
    primary key (account-number))
```

- Given a DDL file, the DDL compiler generates a set of tables.

- The authors also define a subset of DDL called *Data storage and definition language* for specifying things such as:
  - Location on disk
  - Physical-level formatting
  - Access privileges

Data Manipulation Language (DML)

- DML is used for accessing and manipulating a database.
- Two classes of DMLs:
  - *Procedural* – user specifies how to get the required data.
  - *Non-procedural* – user specifies what data is required, but not how to get that data.
- SQL is usually referred to as a non-procedural query language.

Non-Procedural SQL Examples

- Find the name of the customer with customer-id 192-83-7465:

  select *customer.customer-name*
  from *customer*
  where *customer.customer-id* = '192-83-7465'

- Find the balances of all accounts held by the customer with customer-id 192-83-7465:

  select *account.balance*
  from *depositor*, *account*
  where *depositor.customer-id* = '192-83-7465' **and**
  depositor.account-number = *account.account-number*

- Nonprocedural also called declarative programming

Procedural SQL Example

- Find the name of the customer with customer_id 192-83-7465:

```
try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery("select customer.customer_name" +
                                       "from customer" +
                                       "where customer.customer_id = '192-83-7465'");
        while(rs.next){
            String s = rs.getString(1);
        }
    } catch (SQLException e){}
```

- Resultset declares what data is needed, which are included in the line of the SQL query:

  select customer.customer-name **from** customer **where** customer.customer-id = '192-83-7465'

- The while loop states the way to retrieve the data.

SQL Examples

- Databases are typically accessed by:
  - Users through a command line interface
  - Users through a query or software editing tool, e.g., MySQL Workbench
  - Application programs that (generally) access them through embedded SQL or an application program interface (e.g., ODBC/JDBC)

Concept #6: Database Users

Database Users

Users are differentiated by the way they interact with the system:

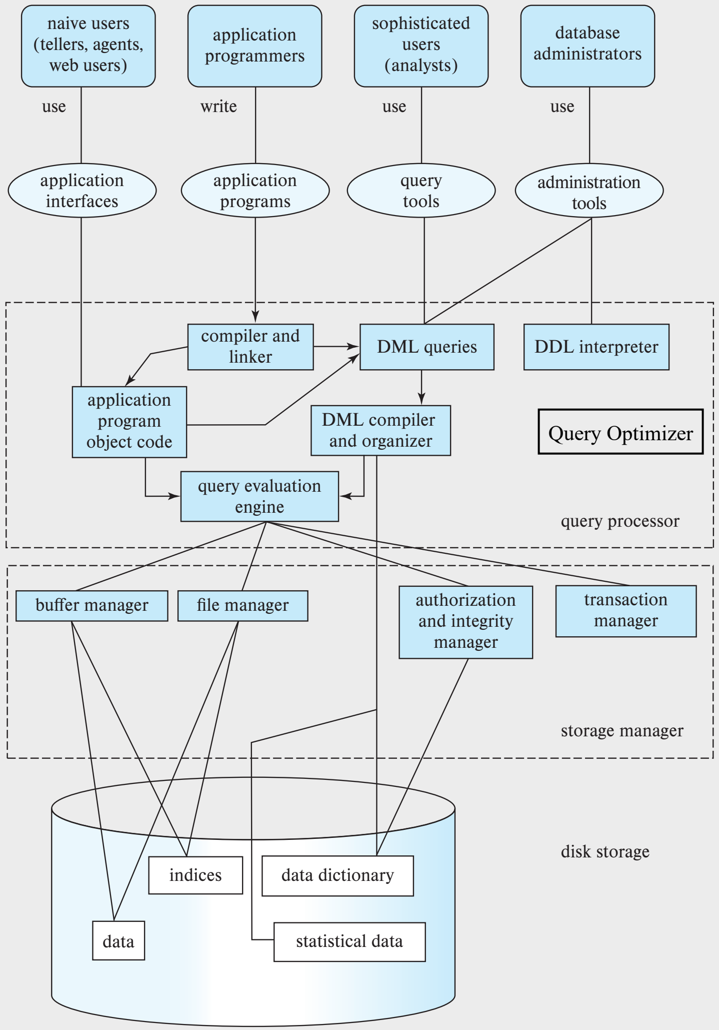
- *Naïve users*
- *Application programmers*
- *Specialized users*
- *Sophisticated users*

Database Administrator (DBA)

- The DBA coordinates all the activities of the database system; has a good understanding of the enterprise's information resources and needs.
- DBA duties:
  - Granting user authority to access the database
  - Acting as liaison with users
  - Installing and maintaining DBMS software
  - Monitoring performance and performance tuning
  - Backup and recovery
- According to the book, the DBA is also responsible for:
  - Logical and Physical schema definition and modification
  - Access method definition
  - Specifying integrity constraints
  - Responding to changes in requirements

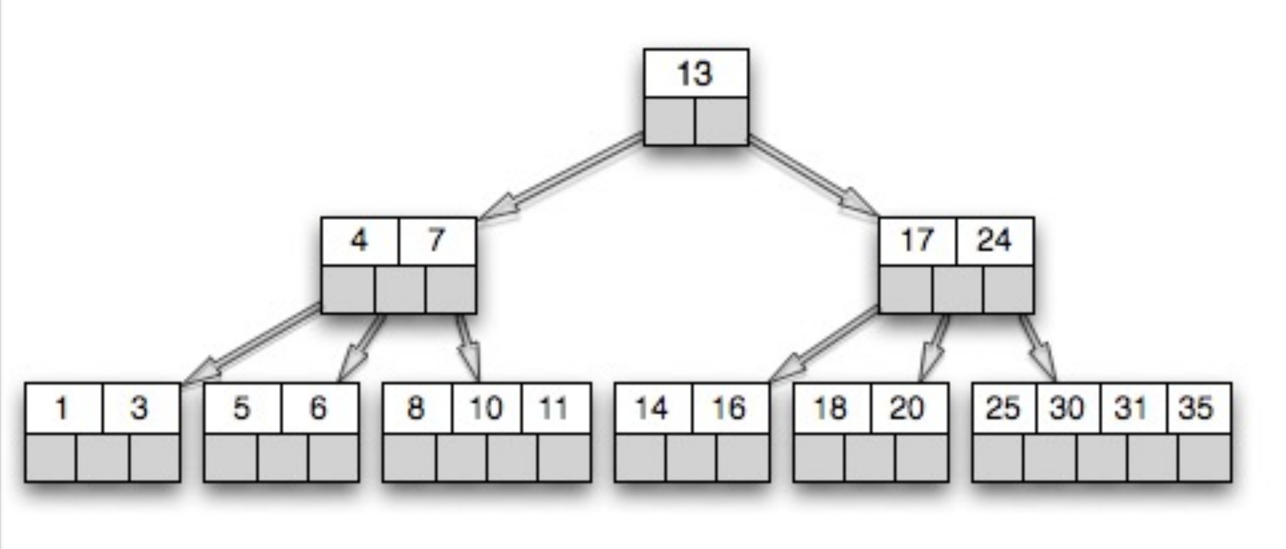
Concept #7: DBMS Structure

Overall DBMS Structure

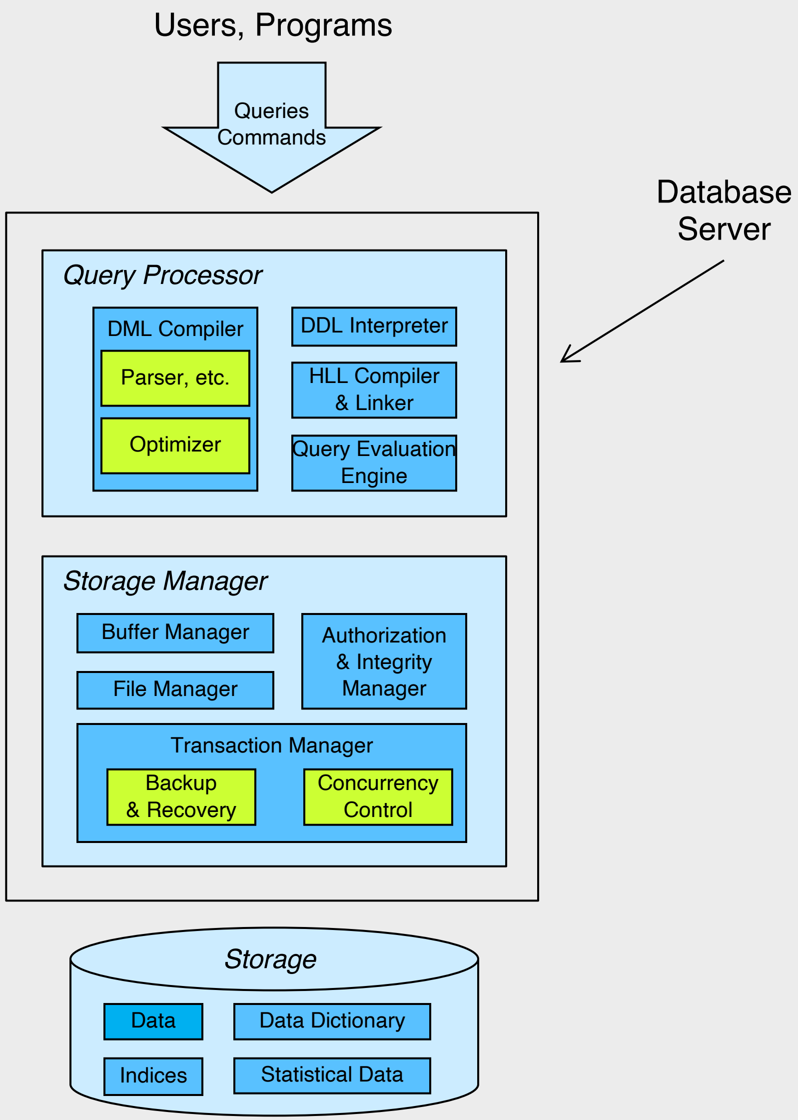


Why is data retrieval usually very fast in a database?

- The index is usually stored as B-trees or hash indexes. MySQL uses R-tree



Overall DBMS Structure



Overall DBMS Structure

The following components of a DBMS are of interest to us:

- transaction manager
- buffer manager
- file manager
- authorization and integrity manager
- query optimizer

Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application
- The *transaction manager* performs two primary functions:
  - backup and recovery
  - concurrency control
- *Backup and recovery* ensures that the database remains in a consistent (correct) state despite failures:
  - system, power, network failures
  - operating system crashes
  - transaction failures.
- *Concurrency-control* involves managing the interactions among concurrent transactions.

Storage Management

- The *buffer manager* loads data into main memory from disk as it is needed by the DBMS and writes it back out when necessary.
- The buffer manager is responsible for:
  - loading pages of data from disk into a segment of main memory called "the buffer"; a.k.a. "the cache"
  - determining which pages in the buffer get replaced
  - writing pages back out to disk
  - managing overall configuration of the buffer, decomposition into memory pools, page time-stamps, etc.

Storage Management

- The *file manager* is responsible for managing the files that store data.
  - formatting the data files
  - managing free and used space in the data files
  - defragmenting the data files
  - inserting and deleting specific data from the files

Authorization & Integrity Management

- The *authorization & integrity manager* performs two primary functions:
  - data security
  - data integrity

- Data security:
  - ensure that unauthorized users can't access the database
  - ensure that authorized users can only access appropriate data

- Data integrity:
  - in general, maintains & enforces integrity constraints
  - maintains data relationships in the presence of data modifications
  - prevents modifications that would corrupt established data relationships

Query Optimization

- A given query can be implemented by a DBMS in many different ways.
- The *query optimizer* attempts to determine the most efficient strategy for executing a given query.
- The strategy for implementing a given query is referred to as a *query plan*.

End of Chapter 1