Announcements

- Welcome to CSc 460 – Database Design!
- Class Information Sources:
  - D2L (d2l.arizona.edu/d2l/home/1186894)
  - Web (u.arizona.edu/~mccann/classes/460)
  - Piazza (piazza.com/class/l5sr9c7qjh22bi)
- Class Modality: In-person. Per University policy, this class is available only in the modality in the Catalog.
  - Masking is not required in class. Please respect the choices of your fellow students.
  - Lectures will be recorded (when the technology is working) and will be available via D2L (in the Content section, under Lecture Recordings).

Continues . . .

Announcements (continued!)

- Handouts: (1) Summary of the Class Syllabus
  (2) Program #1
  (3) Program Style Requirements
  - We will cover these (inc. the full syllabus) today
- Exam dates are already scheduled (see the syllabus' Topic Schedule). Please schedule your job interviews, weddings, espionage trials, etc., accordingly.
- The Fall CS Career Fairs are coming soon:
  - Virtual: Sept 15th
  - In-person: Sept 16th

  } See Arthur's Aug 16th email
- Any questions before we get started?

Announcements

- Program #1 is now officially assigned
  - Part A is due in one week
    - If you need to fix it later, please do! (Students often discover issues w/ Part A while working on Part B.)
    - We will 'grade' your Part A submission, and deduct late days as appropriate, so please take the Part A due date seriously.
    - One correction so far: The State field is also a string field.

Topic 1:

Databases and Database Management Systems

Definitions of 'Database'

- "A shared collection of logically related data and its description, designed to meet the information needs of an organization." (Connolly/Begg, 6/e)
- "... a collection of persistent data that is used by the application systems of some given enterprise" (Date, 7/e)
- "A collection of related data" (Elmasri/Navathe, 5/e)
- "[A] collection of records kept for a common purpose" (O'Neil[2], 2/e)
- "A collection of data, typically describing the activities of one or more related organizations" (Ramakrishnan/Gehrke, 3/e)
- "A collection of related records stored with a minimum of redundancy that many users can share simultaneously" (Shepherd)
- "[A] collection of [interrelated] data ... contain[ing] information relevant to an enterprise" (Silberschatz/Korth/Sudarshan, 4/e)

Common Definition Themes

Three very popular ideas in the definitions:

- **Collection**
- **(Inter) Related**
- **Data / Records**

Other ideas of note:

- Enterprise / Organization / Common Purpose
- Sharing
- Minimized Redundancy
- Persistence

Questions:

- Is your cell phone's contact list a database? yes
- Is your wallet a database? yes
- Where is the software?

Definitions of 'Database Management System'

DBMS

- "A software system that enables users to define, create, maintain, and control access to the database." (Connolly/Begg, 6/e, p. 16)
- "A computerized record–keeping system" (Date, 7/e, p. 2)
- "A collection of programs that enables users to create and maintain a database" (Elmasri/Navathe, 5/e, p. 5)
- "A program product for keeping computerized records about an enterprise" (O'Neil/O'Neil, 2/e, p. 1)
- "Software designed to assist in maintaining and utilizing large collections of data" (Ramakrishnan/Gehrke, 3/e, p. 4)
- "Cost–effective methods for storing, organizing, retrieving, and managing data" (Shepherd)
- "A collection of interrelated data and a set of programs to access those data" (Silberschatz/Korth/Sudarshan, 4/e, p. 1)

DBMS Components

1. The Database
2. Administration – Database Administrator (DBA)
   - handle maintenance, performance, security, user education, ...
   - regular supervision is needed!
3. Application Programs
   - from text queries to web interfaces
   - there are standard programming interfaces (e.g. JDBC)
4. Hardware
   - often a dedicated cluster of servers

Why use a DBMS? (1 / 3)

1. Data Sharing
   - one common location vs. many individual copies
   - helps maintain data consistency between copies

2. Redundancy Control
   - intra-file redundancy managed by Data Normalization
   - Distributed DBMSes intentionally copy files to maintain accessibility

3. Centralized Control
   - DBAs provide experienced & benevolent oversight
   - Allows users to be users

Why use a DBMS? (2 / 3)

(4) Data Integrity
- Ex: June 31st
- Most integrity info is domain-specific
- DBMSes support user-defined integrity rules

(5) Data Security
- DBMSes can flexibly limit access to users
- Active management by DBA > user management

(6) Views
- are automatically managed presentations of the database
- provides clarity, convenience, & some security

Why use a DBMS? (3 / 3)

⑦ Data Independence
- Idea: User's shouldn't know how data is organized within the DBMS
- 2 types:
  - ⓐ Logical Data Independence (LDI)
    - Conceptual Schema - describes the data and its relationships
    - LDI allows schema to change w/o applications changing
    - Difficult to achieve
  - ⓑ Physical Data Independence (PDI)
    - PDI allows physical storage to change w/o applications changing (e.g., create index)
    - More easily achieved than LDI

Disadvantages of DBMSes

1. DB design is complex
   - matching design to desired function is not trivial
2. Cost
   - organizational licenses are expensive
   - DBAs need to be paid
   - Servers are not cheap
3. Availability
   - A DBMS is a potential "single" point of failure
   - Making backups is important!
4. Speed v. Flexibility
   - Users want high performance, but DBMSes are designed for general purpose use.

Announcements

- Program #1:
  - Part A is due Wednesday. We're looking for:
    - your Prog1A.java file that is...
    - a good attempt at creating the binary file, with...
    - good documentation & coding style
  - Follow the instructions in the Hand In section of the assignment handout to submit your code.
  - Part B is due next Wednesday.
- Our office hours are set – see Piazza and D2L for the schedule. We've started holding them today.

Data Languages

Four types of manipulation, four languages, all within SQL:

- Data Description Language (DDL)
  - turns the conceptual schema into a physical database description
- Data Manipulation Language (DML)
  - insertion / deletion of data
- Data Control Language (DCL)
  - security
- Query Language
  - read-only, to answer questions

Topic 2:

Database Management System Architectures

What is an 'Architecture' of a DBMS?

Definition: Schema

The overall description of a database
– at varying granularities

Definition: Architecture

A description of DBMS components and their interconnections.

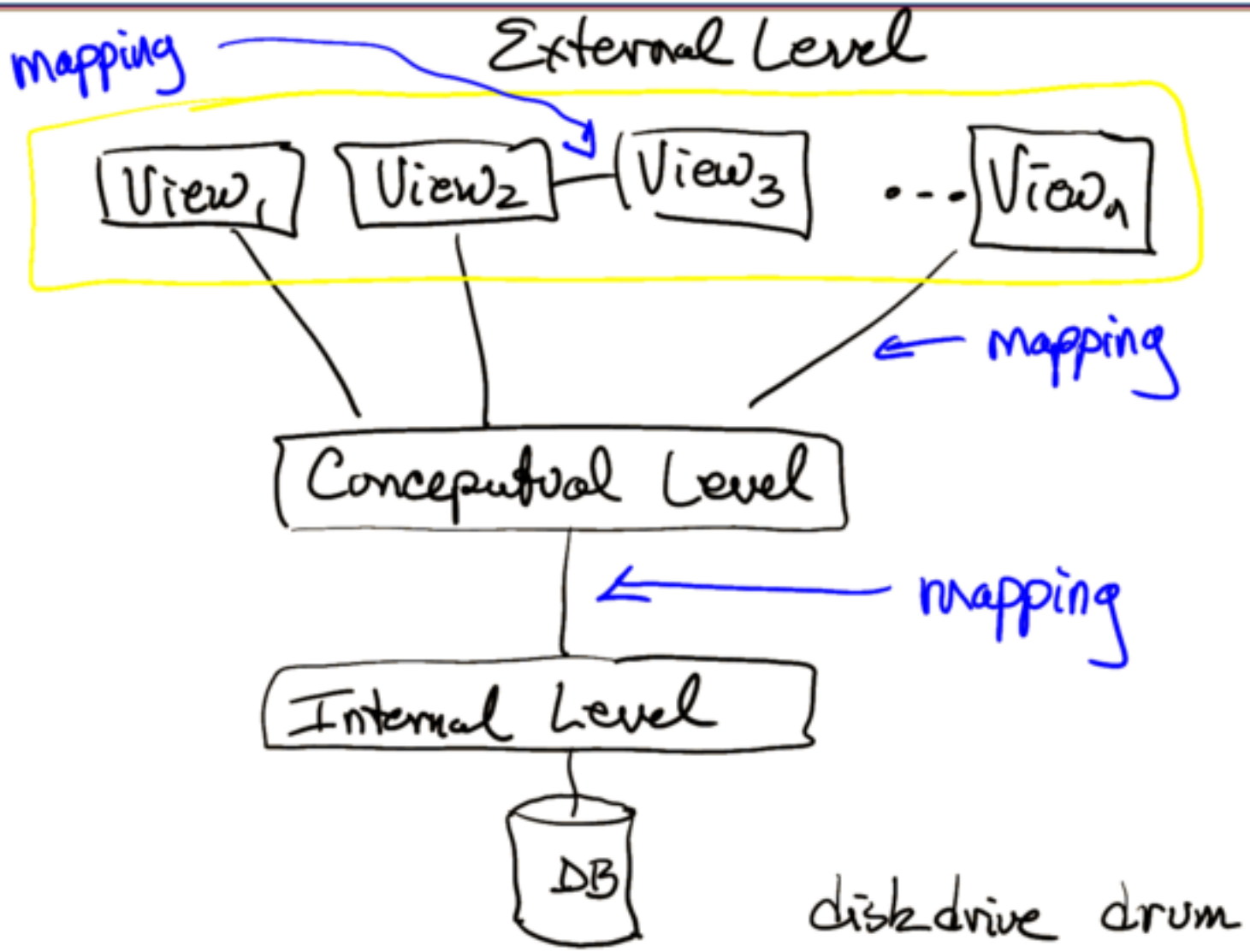
Notes:
- the schema is part of the architecture
- often, the terms are used interchangeably
- I have not seen a text define "architecture" ^ DB

The ANSI/SPARC Architecture (1 / 4): Background

a.k.a. Three-Level Schema

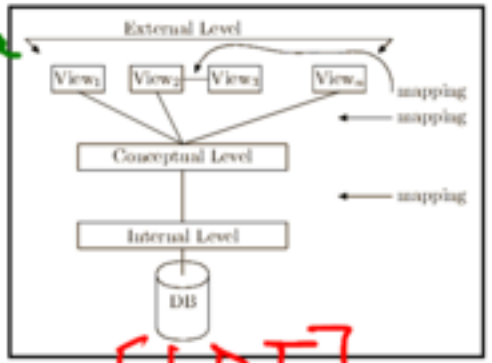
- A product of the Standards Planning and Requirements Committee (SPARC) of the American National Standards Institute (ANSI)
- Never formally adopted as an ANSI or International Standards Organization (ISO) standard, but still very influential
- Created to standardize terms and concepts surrounding DBs and DBMSes
- Goals:
  - Allow for multiple views of the data to satisfy a range of users
  - Allow for a physical (disk-level) description of the database
  - Provide an abstraction layer to separate the two

The ANSI/SPARC Architecture (2 / 4): The Diagram



The ANSI/SPARC Architecture (3 / 4): The Levels

- External Level – described by External Schema
  - defines end-user perspective on the DB



- Conceptual Level – desc. by a Conceptual Schema [CDI]
  - defines field groupings, data relationships, etc.
  - abstract from physical considerations.
- Internal Level – described by an Internal Schema [PDI]
  - defines record sizes, indices, field representations, record ordering, etc.
  - hidden from applications & users

The ANSI/SPARC Architecture (4 / 4): The Mappings

The interfaces between the levels are known as *mappings*.

- External – External Mapping
  - often easier to tweak an existing view
  - usually more efficient to base views on Conceptual Level.
- External – Conceptual Mapping
  - Allows field renamings & arrangements for the user's benefit.
  - provides Logical Data Independence (LDI)
- Conceptual – Internal Mapping
  - Converts logical structure to physical reps.
  - provides Physical Data Indep. (PDI)

Client - Server Architectures (1 / 3): Background

- Originally: DBMSes were built with a centralized architecture.
  - All components (OS, DBMS, compilers, etc.) on one computer
  - “All or nothing” with respect to failures
  - Often a performance bottleneck
- Decentralization became feasible when:
  - Computers became less expensive and more powerful
  - Broadband networking became commonplace

Client - Server Architectures (2 / 3): Two-Tier

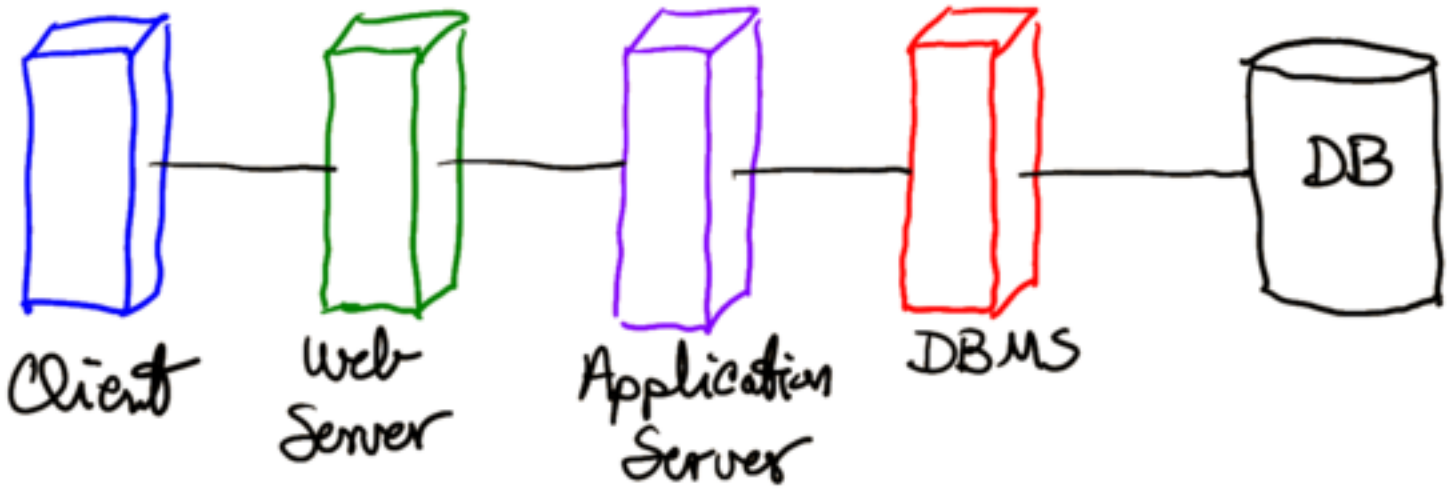
- One possible division of services:
  1. The client – presents info to user, runs applications that get data from --
  2. The server – fields requests from clients, runs the DBMS
- Lots of variations
  - #s of clients & #s of servers
  - parts of DBMS can be executed/stored by clients.

Client - Server Architectures (3 / 3): Multi-Tier

Why add more tiers?

Add new capabilities, spread out old ones, ...

Example of a Four-Tier architecture:



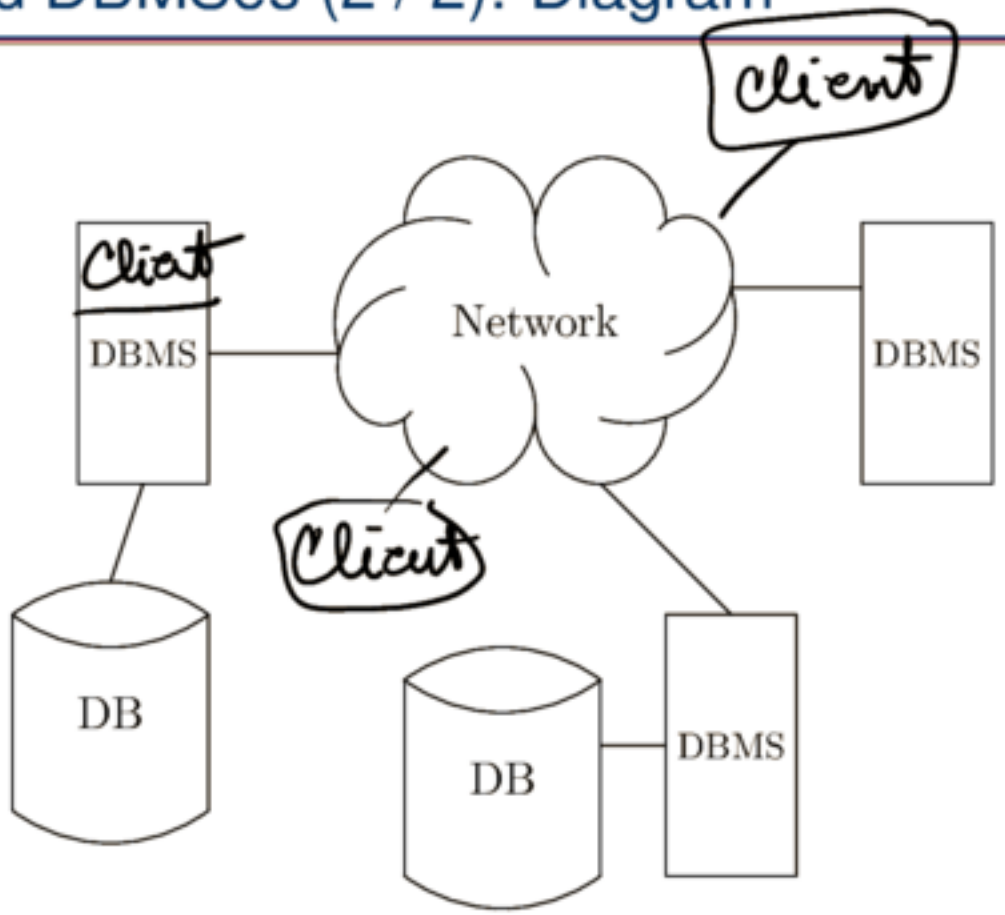
Distributed DBMSes (1 / 2): Motivation

DDBMS

- A single DBMS server (with its single DB) is a single point of failure
- Solution: A DBMS can be operated by several servers.
  - Each server has all, some, or none of the DB stored locally

    (replication is permitted for performance and reliability)
  - DDBMS sites communicate to handle nearly all tasks
  - Goal: Be completely transparent to the users
- Again, details are beyond the scope of this course

Distributed DBMSes (2 / 2): Diagram



Extra Slides

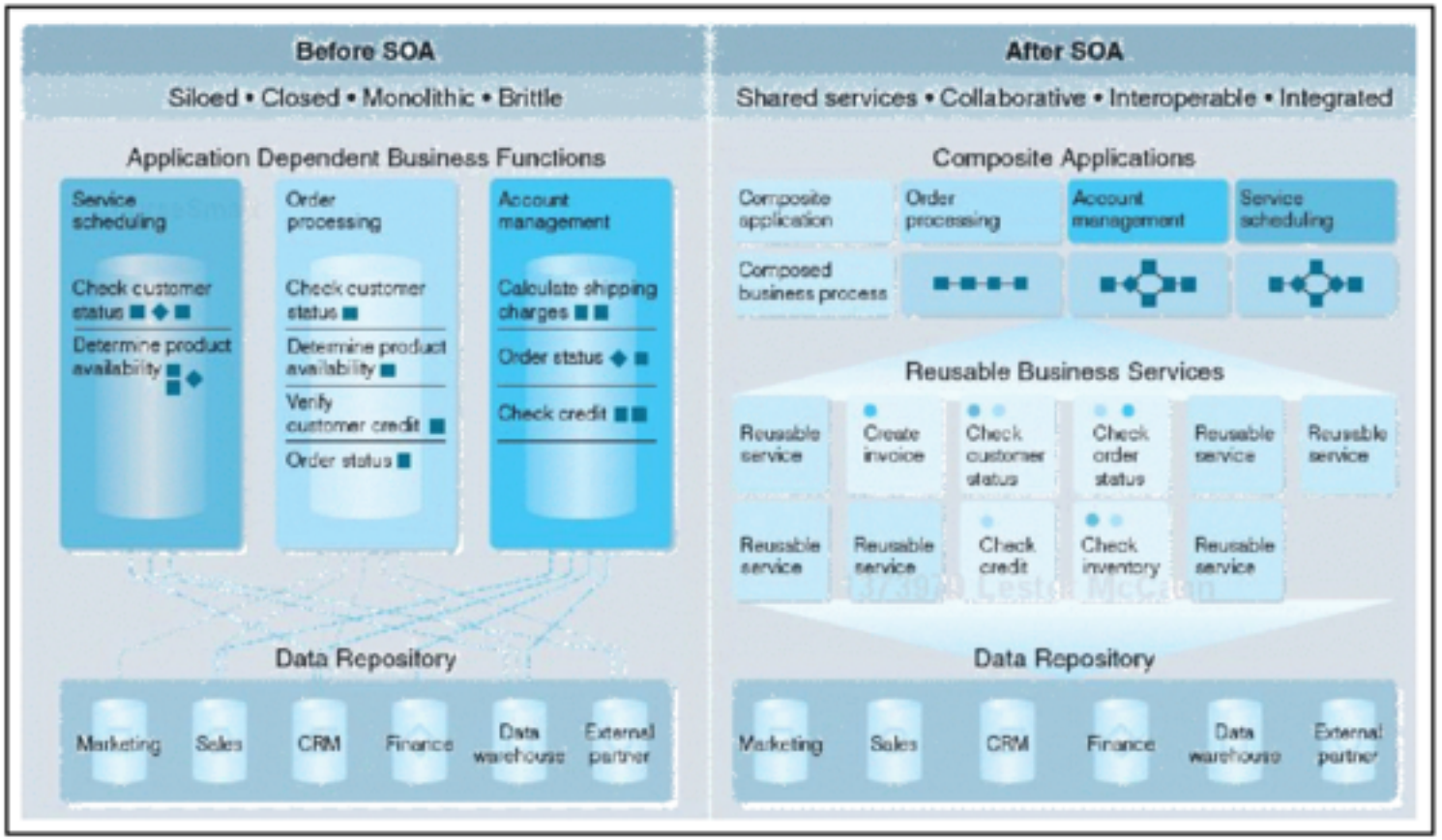
The remaining slides in this topic are some that I no longer cover in class. I won't ask about them on an exam, but they could be referenced on a homework.

Service-Oriented Architectures (1 / 3): Motivation

- SOA is a software design technique:
  - Apps are built using pre-written service modules

    E.g., a data visualization module
  - Modules are located & accessed via a common interface
- Goal is to be flexible with the adoption of new business processes
- A web service is an interface used by service modules
  - That is, it can be a component of an SOA.
- Further details are beyond the scope of this course

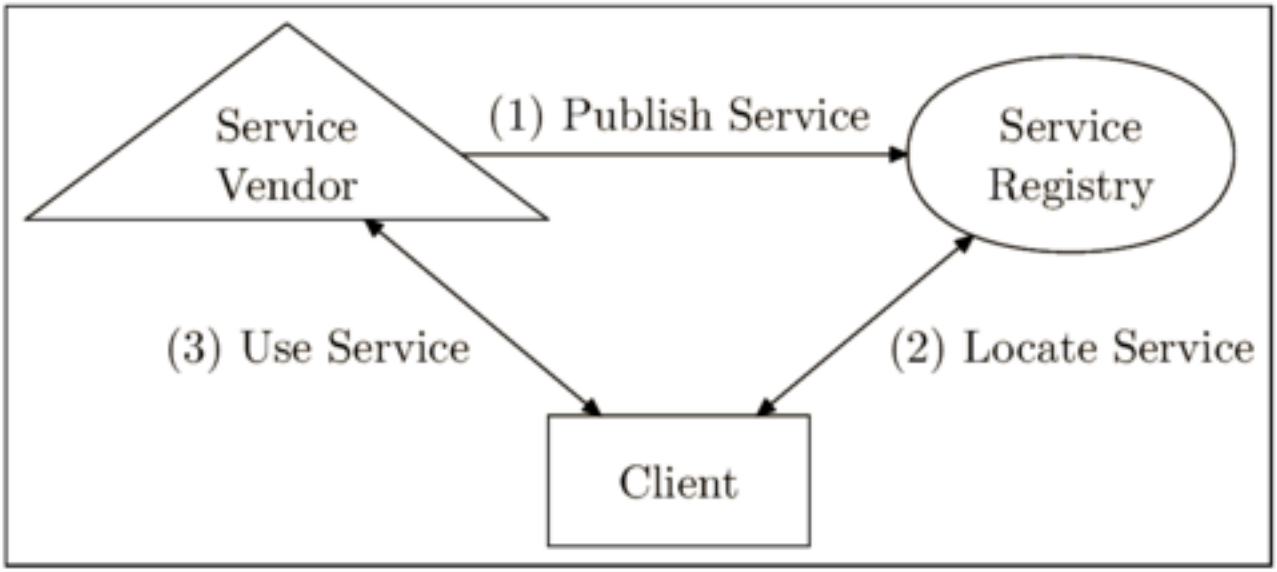
Service-Oriented Architectures (2 / 3): Before/After



Credit: Connolly/Begg, "Database Systems," 6/e, p. 72.

Service-Oriented Architectures (3 / 3): Accessing

Advertising, Finding and Using a Service:



One example: Web browser plug-ins.

(This approach is typical for web services, too.)

Additional DB–Related Architectures

- Web Services
  - Ex: Stock Quotations; Google Docs

  Two types:

  (a) Simple Object Access Protocol (SOAP)-based
  – Typically uses XML

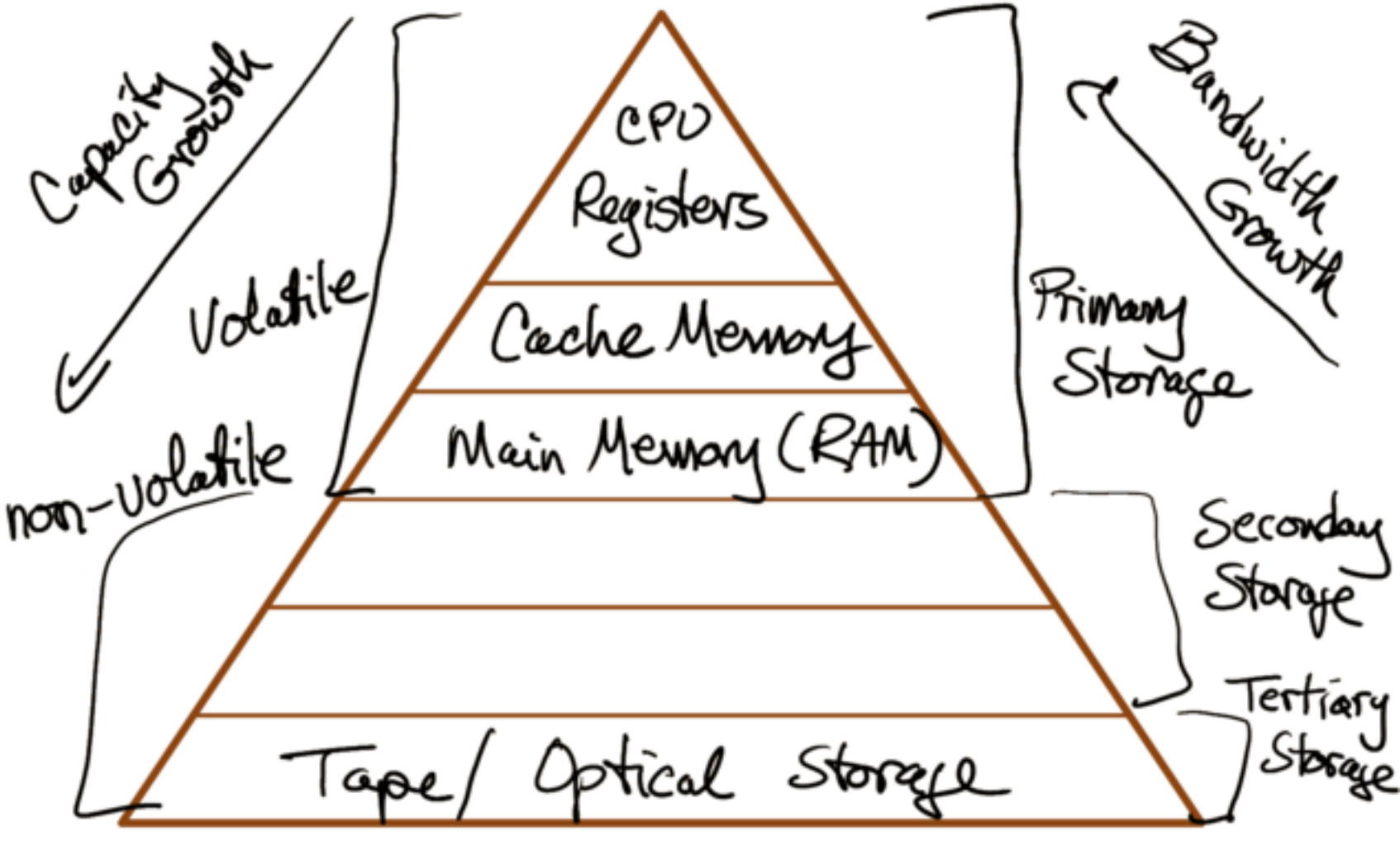
  (b) RESTful (Representational State Transfer) – stateless
  – Ex: HTTP

- Data Warehouses
  - Support decision–making
- Cloud Computing
  - Provides dynamic resource provisioning (of DBMSes!)

Topic 3:

Files and Indexing

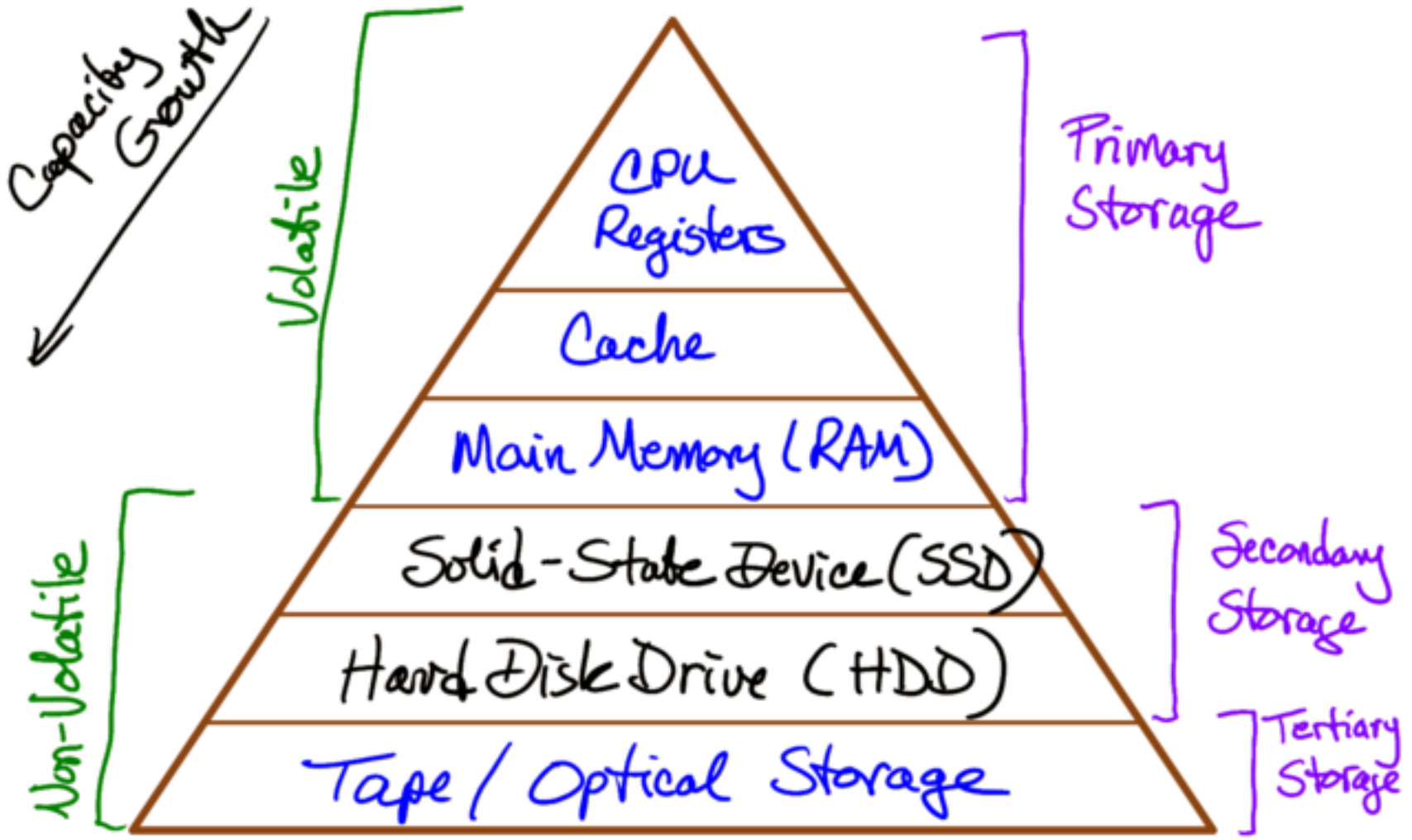
The Storage Pyramid



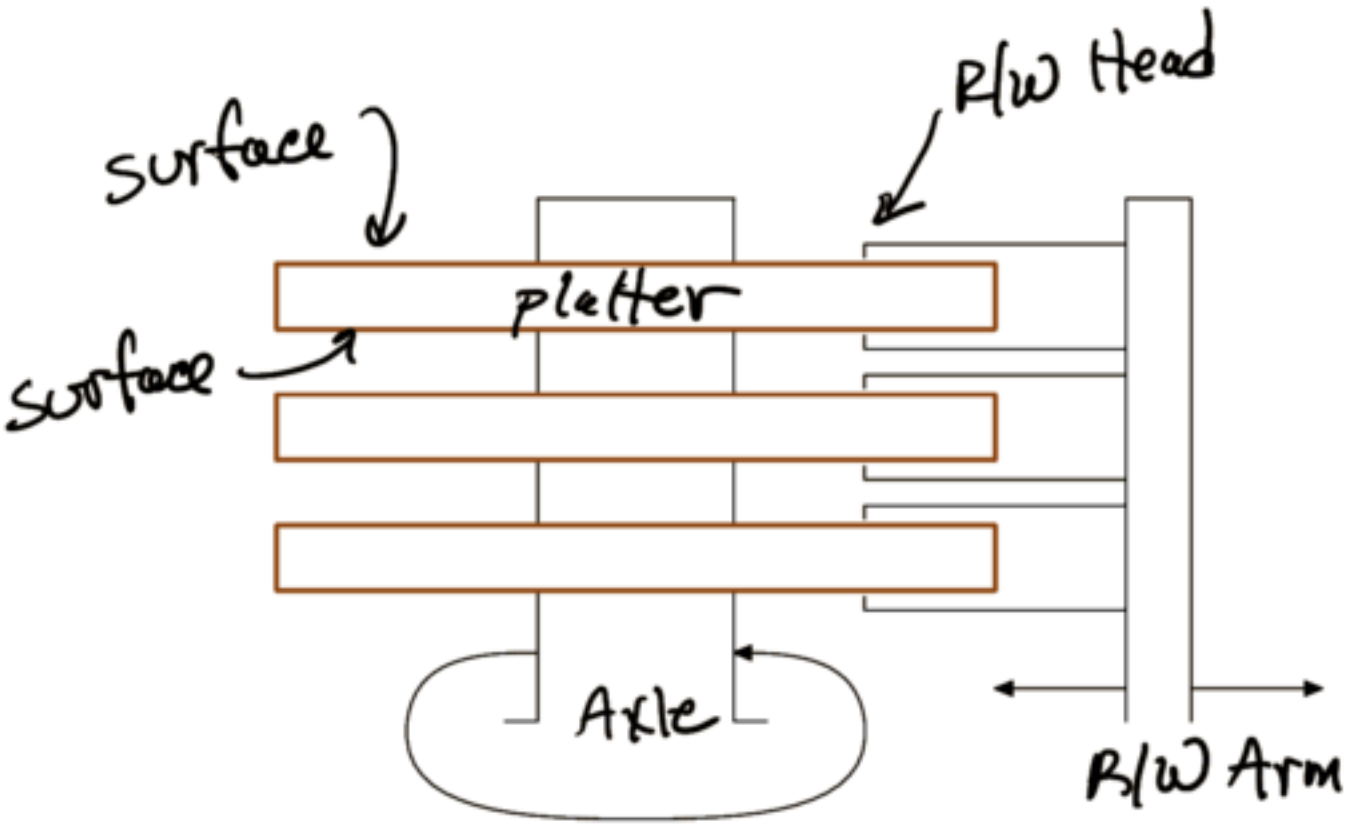
Announcements

- Program 1:
  - Part A "first draft" is due now
    - TAs will only check submission for documentation and style, not for correct operation
  - Complete assignment is due in one week
    - Submit both Part B and Part A, to be sure that they work together correctly
- Program #2 will be assigned next Wednesday
- No classes Monday - Labor Day!
  - Our only day off all semester...
- CS Tutor Center has quite a bit of coverage for 460
- "Old Storage Device Show and Tell" available w/ lecture recordings in DZL.

The Storage Pyramid

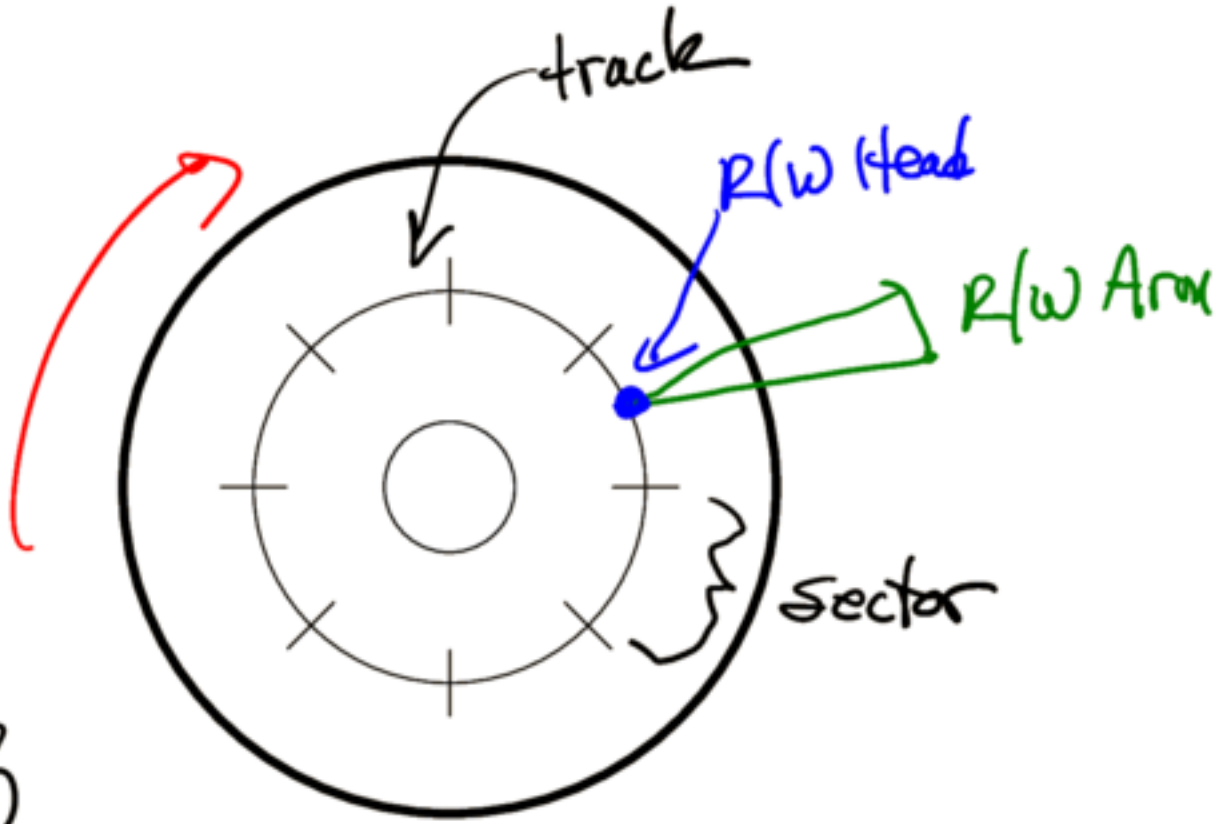


Hard Drive Physical Characteristics (1 / 2)



Side View

Hard Drive Physical Characteristics (2 / 2)



Top View (one surface)

A block is a group of sectors

⇒ The basic unit of HDD storage

Sources of Read / Write Delay

The three major sources of delay (in descending order):

1. Seek Time
   - move R/W head to appropriate track
2. (Rotational) Latency
   - wait for block to rotate to R/W Head
3. Transfer Time
   - read/write data from/to surface.

Western Digital 3.5” Hard Drive Specs

	WD450AA (9/2000)	WD1001FALS (7/2009)	WD4003FZEX (7/2015)	DC HC550 (8/2020)
Size (GB)	45	1,000	4,000	18,000
Platters & Heads	3 & 6	3 & 6	5 & 10	9 & 18
Bytes per Sector	512	512	512	512 / 4096
Sectors per Surface	14,655,144	325,587,528	781,403,717	??
Rotations (RPM)	5400	7200	7200	7200
R/W Seeks (ms)	9.5 / 13.4	?? / ??	?? / ??	?? / ??
Latency (ms)	5.4	4.2	??	4.16
Cache (MB)	2	32	64	512
Buffer to Host (MB/s max.)	66.6	3000.0	6000.0	600.0
[Power] Read / Write (W)	6.2	8.4	9.5	6.5
Idle (W)	6.2	7.8	8.1	5.6
Standby / Sleep (W)	∼1.1	1.0	1.3	?.?

RAID Background (1 / 3): Disk Mirroring

(a) Disk Mirroring

Advantage(s): Can operate w/a failure of a HDD
Can read in parallel

Disadvantage(s): Cost (2x the drives)
Power

RAID Background (2 / 3): Disk Striping

(b) Disk Striping – striping units are distributed across multiple disks.

Example(s):

4 drives (# 0, 1, 2, 3). A file of 5 blocks
Store block b on drive b % 4.

Advantage(s): Performance (parallel reads/writes)

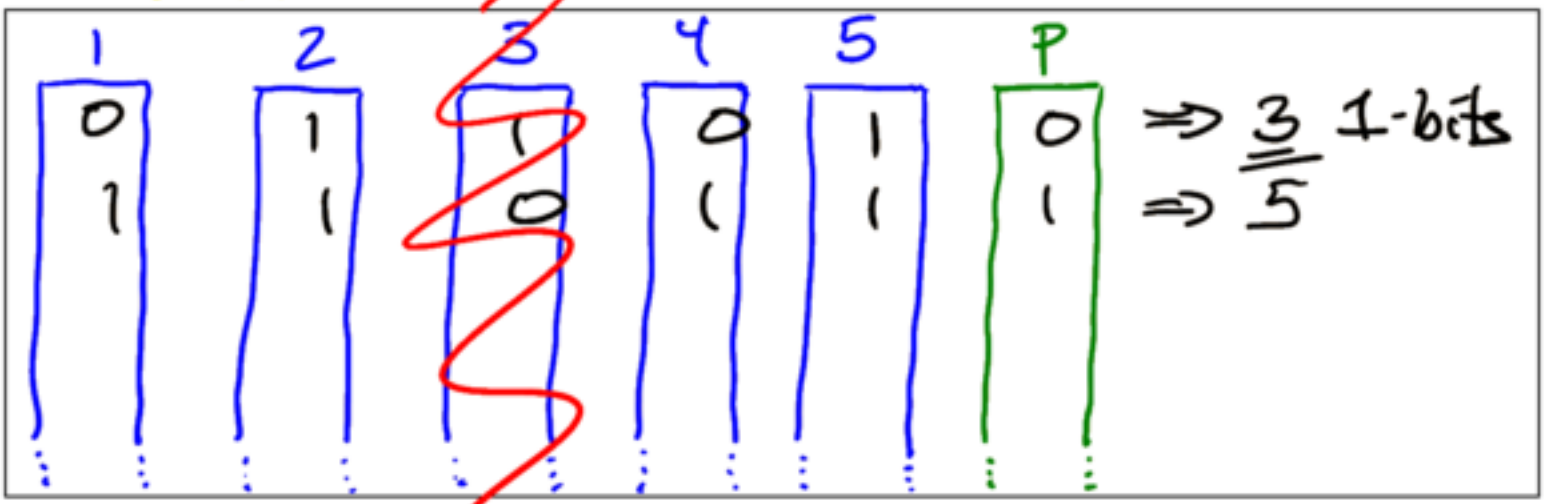
Disadvantage(s): Increased prob. of disk system failure
No data replication

RAID Background (3 / 3): Parity Bits

(c) Parity Schemes

(others: even, mark, none)
odd parity (for this example)
# of 1-bits to be odd

Example(s):



Advantage(s): Can recover 1 failed drive

Disadvantage(s): Buy the parity drive (cost)
Parity drive is popular
only good for 1 failure

Detour: Independent Event Probabilities (1 / 3)

First, some set and probability review!

1. DeMorgan's Laws for Sets:

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$
$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

2. For a sample space S and an event $E \in S$,

   the probability of E's occurrence is:

$$p(E) = \frac{|E|}{|S|}$$

3. $\sum_{e \in S} p(e) = 1$

Detour: Independent Event Probabilities (2 / 3)

Next, Independent Events:

4. Events A and B are *independent* when ...

$$P(A \cap B) = P(A) \cdot P(B) \qquad 1/2 \cdot 1/2 = 1/4$$

Ex: Flip a coin twice

$P(H \cap H) = 1/4$ $\qquad P(H \text{ on } 1^{st} \text{ flip}) = 1/2$

" " 2^{nd} " $= 1/2$

5. Recall: Principle of Inclusion/Exclusion for 2 Sets is:

$$|M \cup N| = |M| + |N| - |M \cap N|$$

Applied to probabilities:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Detour: Independent Event Probabilities (3 / 3)

Probabilities for Independent Events (cont.):

Recall:

4. $p(A \cap B) = p(A) \cdot p(B)$
5. $p(A \cup B) = p(A) + p(B) - p(A \cap B)$

6. Combining (4) and (5):

$$p(A \cup B) = p(A) + p(B) - p(A) \cdot p(B)$$

7. And thanks to DeMorgan's Laws and (4):

$$p(\overline{A \cup B}) = p(\overline{A} \cap \overline{B}) = p(\overline{A}) \cdot p(\overline{B})$$

Probability of Hard Disk Drive Failures (1 / 4)

Factors contributing to HDD failures:

Temperature, manufacturer, age, qty of scan errors, qty of sector remappings

How often does a 'young' (1-3 years old) HDD fail?

Annualized Failure Rate (AFR) = % of devices failing in a year

Refs: http://research.google.com/archive/disk_failures.pdf
https://www.backblaze.com/blog/~~best-hard-drive/~~backblaze-drive-stats-for-q2-2022/

Announcements

- Program #1 is due now!
  - Can use late days if you wish
  - TAs will have grades for you before the due date of ...
- Handout: Program #2 – Linear Hashing Lite
  - I will provide a separate video to help clarify the expectations for the linear hashing index.
- Today: Finish RAID, jump ahead to hash-based indexing, discuss Program #2, then jump back.

Probability of Hard Disk Drive Failures (2 / 4)

What is the p_f for a striped 2-disk system?

$\Rightarrow$ Remember, the system fails when either drive fails!

(Let D#$_f$ be the event of Disk # failing.)

$$
\begin{aligned}
p_f &= p(\mathsf{D1}_f \cup \mathsf{D2}_f) && \text{Either or both!} \\
&= p(\mathsf{D1}_f) + p(\mathsf{D2}_f) - p(\mathsf{D1}_f) \cdot p(\mathsf{D2}_f) && \text{Princ. Inc./Ex. \&} \\
&= 0.02 + 0.02 - (0.02)^2 && \ldots \text{Indep. events} \\
&= 0.0396 \ (3.96\%)
\end{aligned}
$$

Probability of Hard Disk Drive Failures (3 / 4)

New point of view: Be an optimist!

The probability that Disk D# does not fail:

$$p(\mathsf{D\#}_{nf}) = 1 - p(\mathsf{D\#}_{f}) = 1 - 0.02 = 0.98$$

What is the p_{nf} for a striped 2-disk system?

$$\begin{aligned}
p_{nf} &= p(\overline{\mathsf{D1}_f \cup \mathsf{D2}_f}) && \text{[Neither fails!]} \\
&= p(\overline{\mathsf{D1}_f} \cap \overline{\mathsf{D2}_f}) && \text{[De Morgan's]} \\
&= p(\mathsf{D1}_{nf} \cap \mathsf{D2}_{nf}) && [\ \overline{\mathsf{D\#}_f} = \mathsf{D\#}_{nf}\ ] \\
&= p(\mathsf{D1}_{nf}) \cdot p(\mathsf{D2}_{nf}) && \text{[Independent events assumed]} \\
&= p(\mathsf{D\#}_{nf})^2 && \text{[Foreshadowing ...]} \\
&= (0.98)^2 && \text{[From above]} \\
&= 0.9604 \text{ (96.04\%)} && [\ = 1 - 0.0396\ ]
\end{aligned}$$

Probability of Hard Disk Drive Failures (4 / 4)

What if we have *dozens* of HDDs? Say, three dozen?

No problem; being optimistic scales nicely!

$$
\begin{aligned}
p_{nf} &= p(\overline{\mathsf{D1}_f \cup \ldots \cup \mathsf{D36}_f}) && \text{[None fail!]} \\
&= p(\overline{\mathsf{D1}_f} \cap \ldots \cap \overline{\mathsf{D36}_f}) && \text{[Massive De Morgan's]} \\
&= p(\mathsf{D1}_{nf} \cap \ldots \cap \mathsf{D36}_{nf}) && [\ \overline{\mathsf{D\#}_f} = \mathsf{D\#}_{nf}\ ] \\
&= p(\mathsf{D1}_{nf}) \cdot \ldots \cdot p(\mathsf{D36}_{nf}) && \text{[Independent events assumed]} \\
&= (0.98)^{36} && \text{[From last slide]} \\
&= 0.4832\ldots \text{ (48.32\%)} && [\ p_f = 1 - p_{nf} = 0.5168\ ]
\end{aligned}
$$

Remember: Assuming independence is convenient, not realistic!

RAID: Redundant Arrays of Independent* Disks (1 / 2)

* *Originally "Inexpensive"*

Level 0: Striped Volume (N data disks)

⇒ Striping only!

- Good write performance
- Cannot recover from a drive failure
  (Even if we're making backups.)

Level 1: Mirrored (N data disks + N mirror disks)

⇒ Mirroring only!

- Opportunity for parallel reads
- Cost: 2x the # of drives!

RAID: Redundant Arrays of Independent Disks (2 / 2)

Level 5: Block–Interleaved Distributed Parity (N+1 disks)

- striping unit is 1 block
- parity blocks are distributed across disks
- controller design is kinda complex

Level 6: "Double Parity" (N+2 disks)

- Level 5 w/ 2 parity schemes
- Can recover from 2 concurrent failures
- same read perf. as Level 5, but slower write perf.

SSDs: Solid–State Device (Flash) Storage

- NAND–based non–volatile RAM NVRAM
- Not a new idea: Used to have "RAM drives"

  (Even though the 1981 IBM PC had 256 KB RAM – max!)

Advantage(s): No mechanical delays or failures
Excellent read perf.
Consumes little physical space

Disadvantage(s): Write speed lags (erase then write)
# erasures ≈ 100K – 1000K
more expensive than HDDs

Review of Internal Hashing

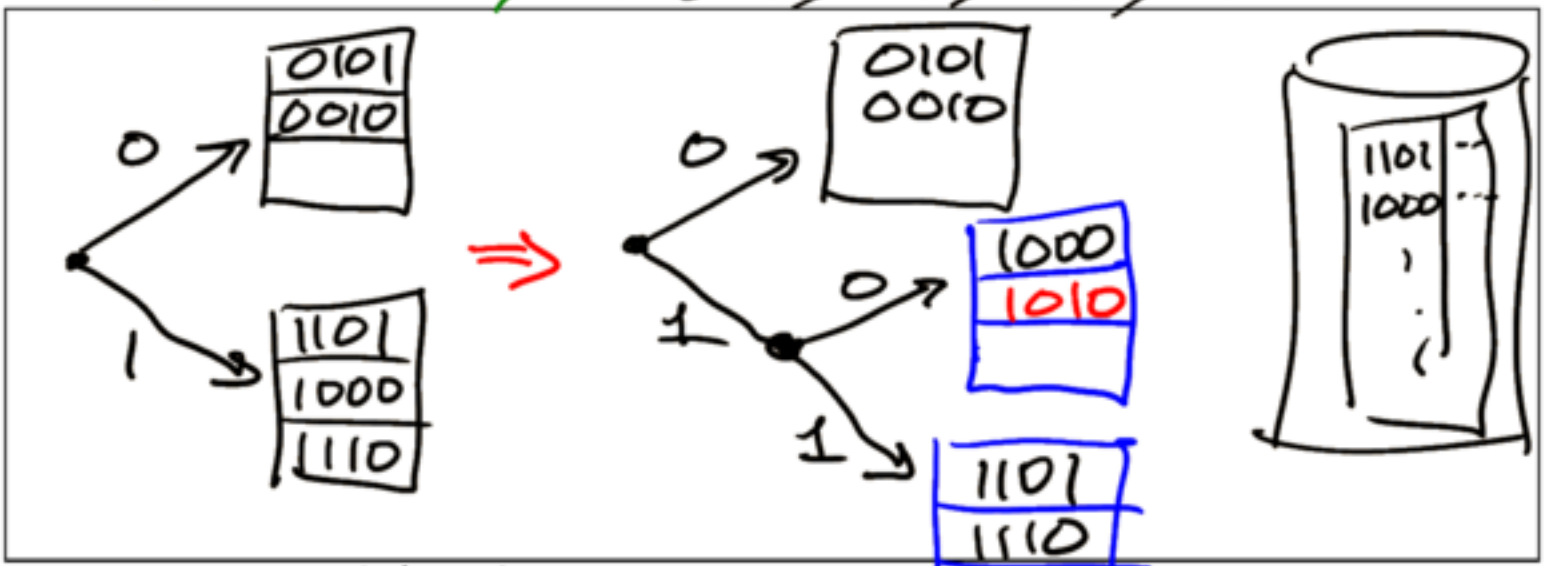
- Goal: $O(1)$ search performance
- Key $\rightarrow$ Hash Coding $\rightarrow$ Compression Mapping $\rightarrow$ Hash Table Index
- Collision Resolution: Chaining v. Open Addressing
- Problem: How can we do hashing on secondary storage?

Dynamic Hashing

Two components:

- A tree-structured directory
- Leaves are disk blocks holding the index entries.

Example(s): Insert 1101, 1000, 0101, 0010, 1110, and 1010:



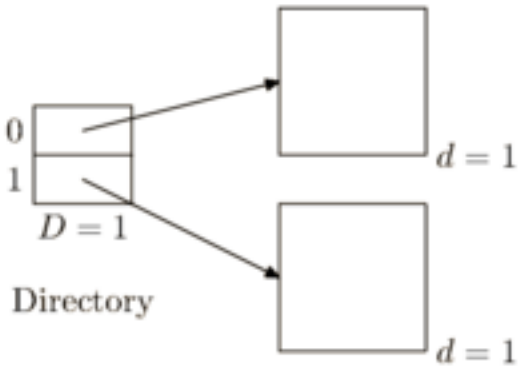
3 entries / block

Problem: How do I store the directory on disk?

Extendible Hashing: Basics

Improvement over Dynamic Hashing:

Directory is an array ⇒



- Each index block has a local depth (d), $d \geq 1$
- The directory has a global depth (D), $D \geq \max(d_i)$
- Directory has k^D pointers, where k is the cardinality of the alphabet set.
  - Ex: When keys are binary, $k=2$

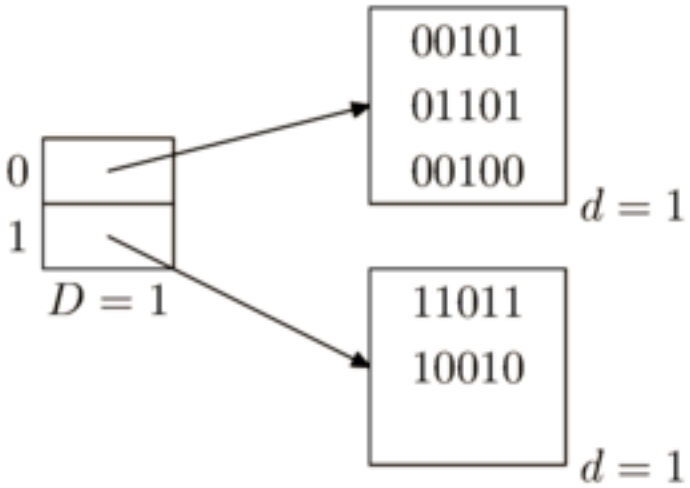
Extendible Hashing: Insertion (1 / 2)

When a key is inserted into a full index block:

- The block becomes k blocks
- The depth of each is one more than the original's
- Existing content is distributed to the new blocks
- If any $d > D$, split ('double') the directory:
  - increase global depth by one
  - create new directory of k^D pointers
  - copy existing block pointers
  - add pointers to new blocks

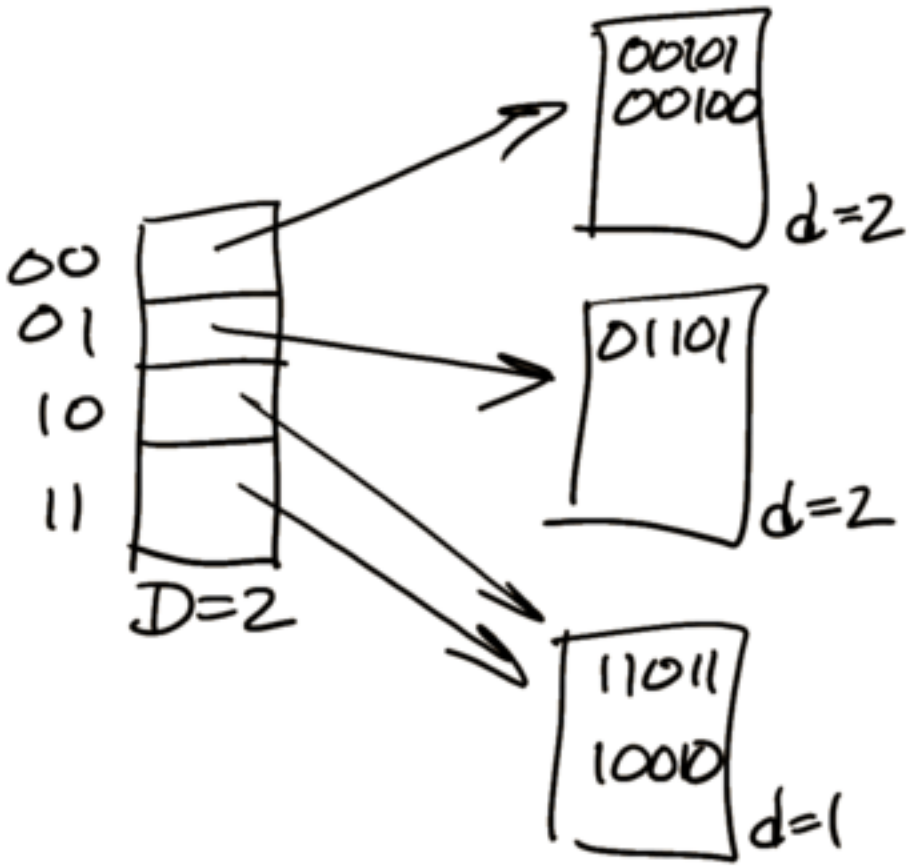
Extendible Hashing: Insertion (2 / 2)

After Inserting 11011, 00101, 01101, 10010, and 00100:



(Assume max. 3 keys/node)

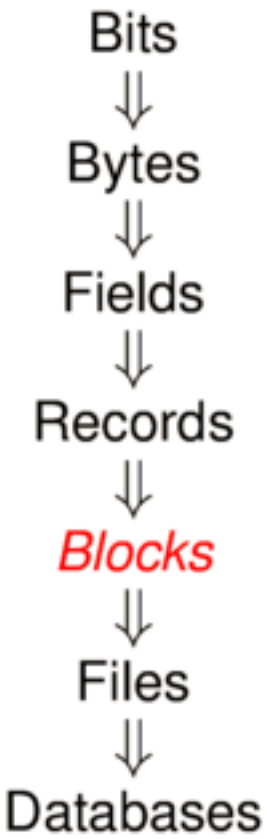
After Inserting 01110:



Announcements

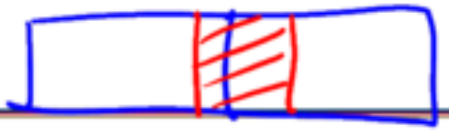
- Program #2 is due a week from Wednesday.
  - The video that shows how to get started is available in D2L.
  - Any questions?
- Today:
  - Finish last slide on Extendible Hashing
  - Jump back to key indexing ideas
  - (Maybe) Start B-Trees
- CS Career Fairs are this week!

File Granularity Hierarchy



File Blocking (1 / 2)

1000/300



Definition: Blocking Factor (bf)

The # of records that can be stored in a block

$$bf = \left\lfloor \frac{\text{block size}}{\text{record size}} \right\rfloor$$

(assuming records are not split between blocks)

Definition: Internal Fragmentation

Unallocatable storage within a block.

8192

Example(s): Block size = 8K bytes; Record size = 55 bytes.

$bf = \left\lfloor \frac{8192}{55} \right\rfloor = 148$ recs/block

$8192 - 148 * 55 = 52$ bytes of internal frag.

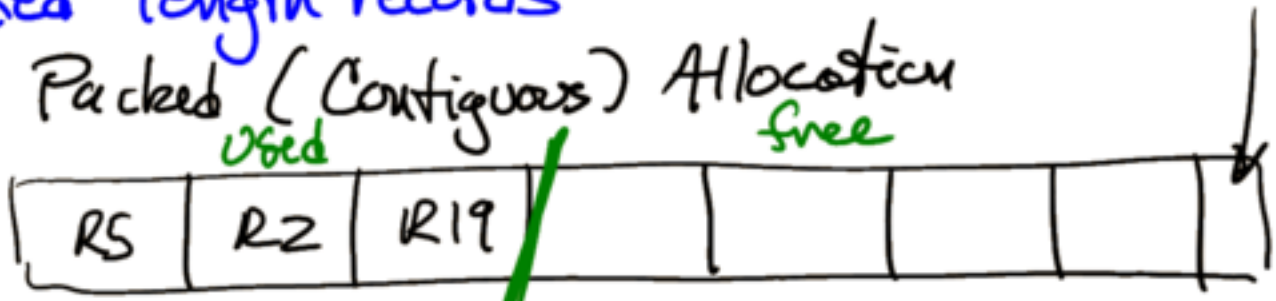
File Blocking (2 / 2)

count or bitmap

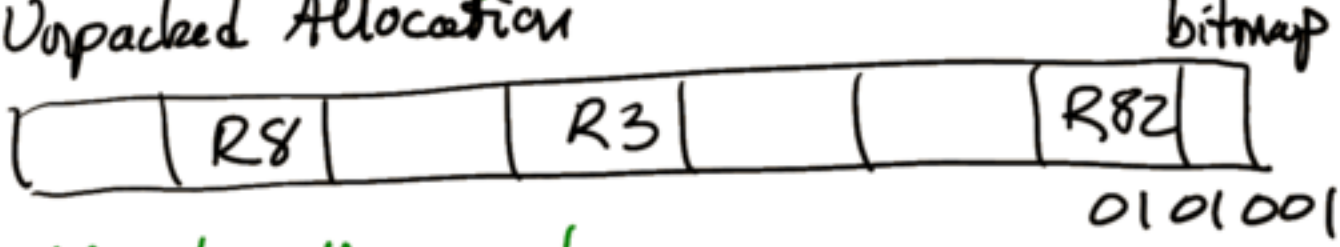
Locating records within blocks:

(1) Fixed-length records

(a) Packed (Contiguous) Allocation



(b) Unpacked Allocation



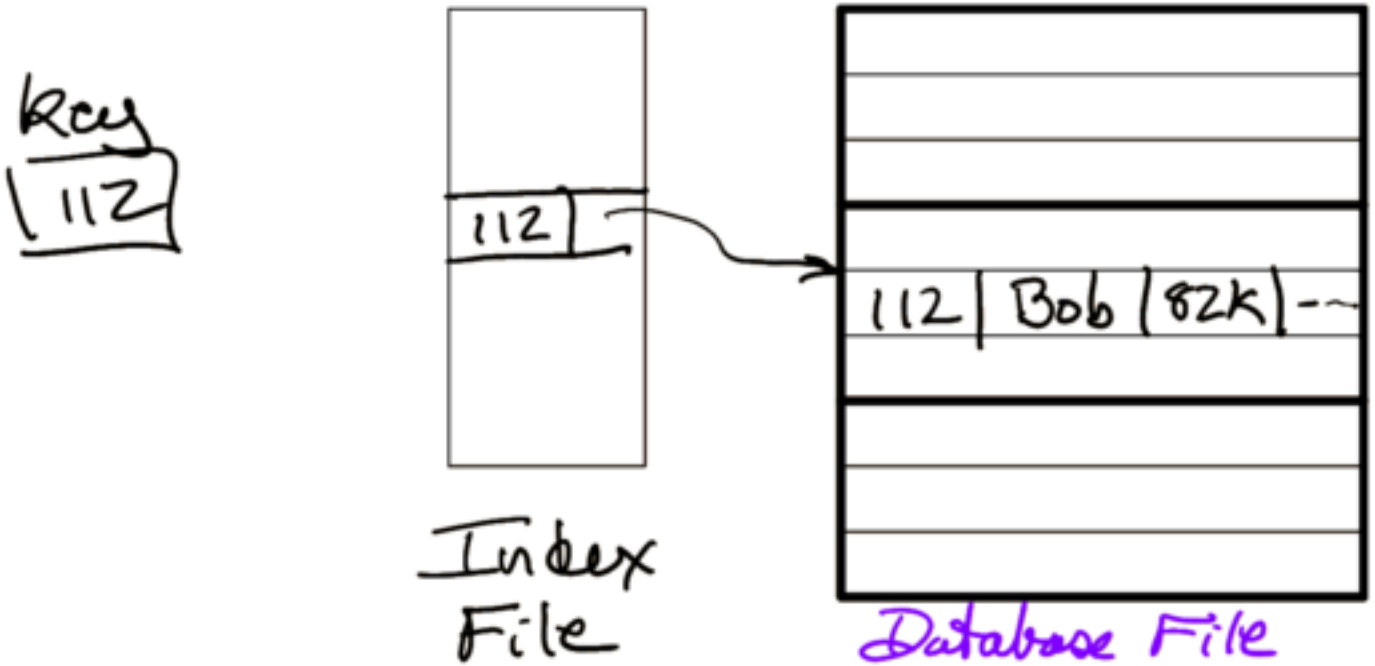
(2) Variable-length records

Replace the bitmap with a slot directory

Indexing

Definition: Index

A file containing structured references to records of another file.



A Few Words about Keys

Some of the types of keys:

- Candidate Key – A key able to uniquely identify a record!
  Ex: StdID, NetID, ~~SSN~~
- Primary Key – The CK chosen to be the identifier
- Secondary Key – Any non-CK
  Ex: SSN, Name, birthday, ---

- Sort Key – The key used to order the records of a file.
- Search Key ---

One Classification of Indices

(1) Ordered

(a) Single-level

Ex: Sorted file of index records

(b) Multi-level

Ex: B^+-Tree

(2) Unordered Index

- Typically hash-based

Ex: Extendible Hashing

Primary Index (1 / 2)

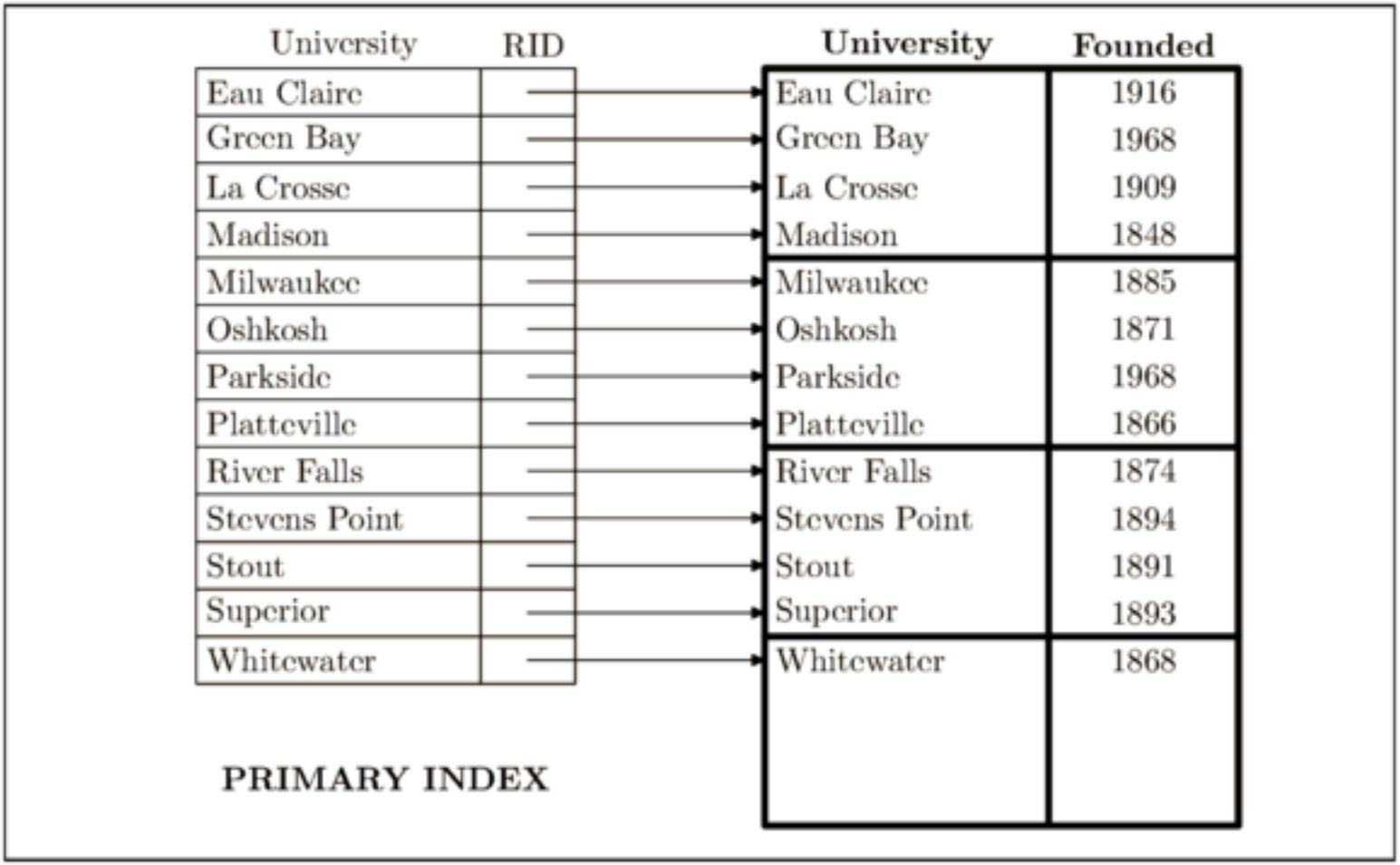
Characteristics:

- The indexed field is a candidate key.
- The index records are sorted on the key.
- The DB file records are sorted on the key.

Notes
- Can have at most one per file
- The term "primary index" has many definitions.

Primary Index (2 / 2)

Example(s):



Clustered Index (1 / 2)

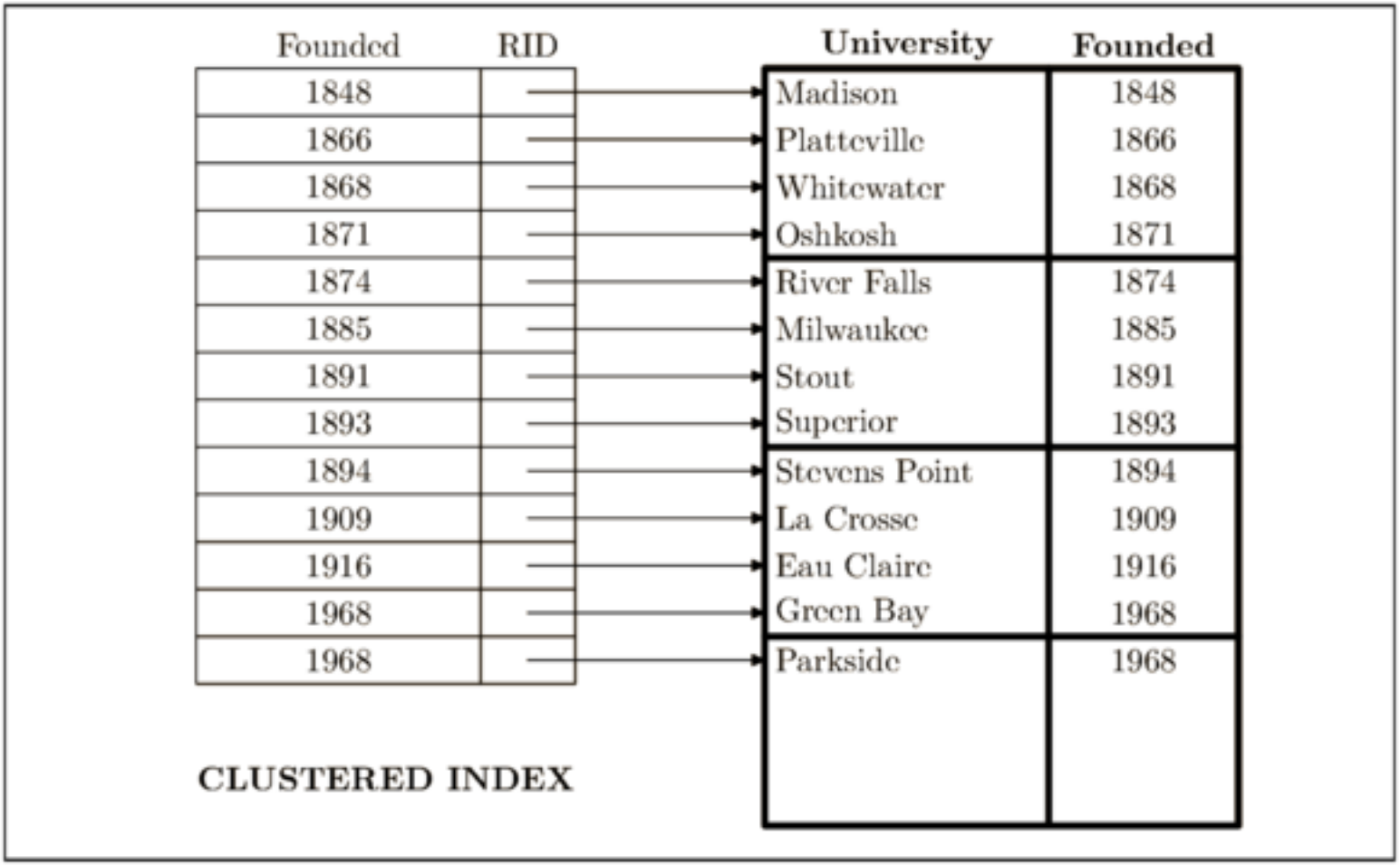
Characteristics:

- The indexed field is a secondary key. (Not a CK!)
- The index records are sorted on the key.
- The DB file records are sorted on the key.

Notes:
- per DB file, the # of primary indices plus the # of clustered indices can be no greater than one.
- Some see primary indices as a special case of clustered.

Clustered Index (2 / 2)

Example(s):



Secondary Index (1 / 2)

Characteristics:

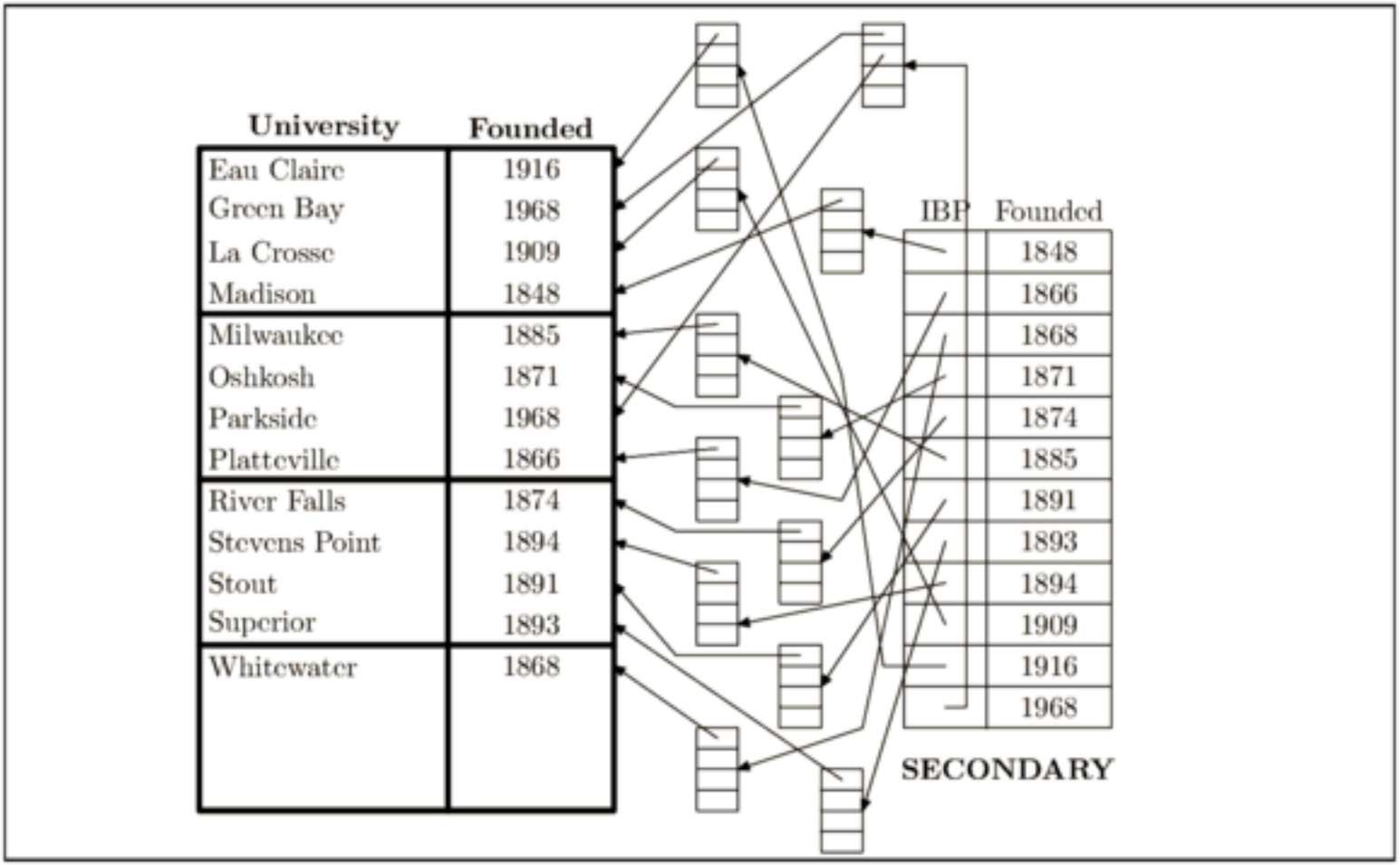
- The indexed field is any field! .
- The index records are sorted on the key.
- The DB file records are not sorted on the key.

Notes

- can have as many of these as you want
- special index construction is needed.

Secondary Index (2 / 2)

Example(s):



Extendible Hashing: Deletion

Question: Do you have lots of disk space available?

If so: Just delete the entry
(leave the space for future insertions)

If not: Collapse the sibling nodes
and shrink the directory if
necessary/possible

Announcements

- Program #2 (Linear Hashing Lite) is due in a week (Wed, Sept. 21)
  - you'll need your .bin file that Program 1 (Part A) created
- Next assignment will be Homework #1, to be assigned next Wed., and due a week later
- Exam #1 is in 3 weeks.

Another Index Categorization: Dense vs. Sparse (1 / 2)

Dense Indices:

- Hold one index record for every file record (excepting perhaps dup. keys)
- Permits existence queries using just the index

Sparse Indices:

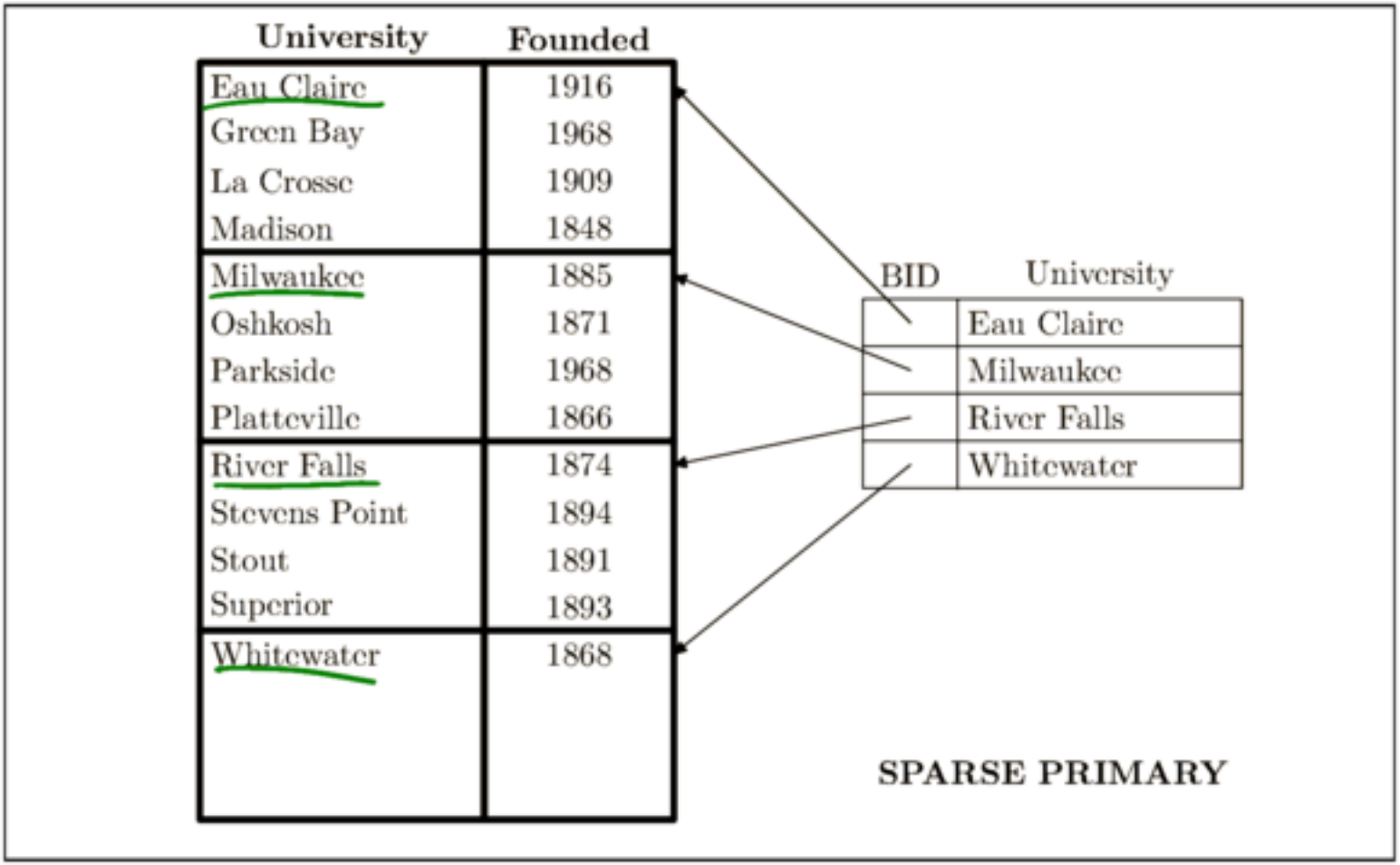
- Index has a subset of the field values of the key
- Smaller, ∴ faster to search

Notes:

- All of the examples of ordered indices from Monday are dense.
- Any ordered index type can be either dense or sparse.

Another Index Categorization: Dense vs. Sparse (2 / 2)

Example(s):

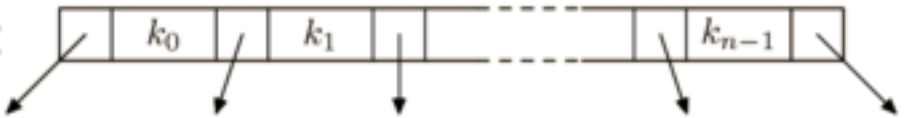


B–Trees: Structure

But first: Know that "B" does not stand for "binary"!

"Bayer"? (Rudolf Bayer & Edward McCreight, '72)
"Balanced"? (It is!) "Boeing"? (McCreight's employer?)

Structure of a B–Tree node:



— A node holding n keys holds $n + 1$ pointers

— Each key is stored in the index exactly once (∴ dense)

— A node's keys are stored in (ascending) sorted order

— Pointer 0's subtree has all keys $<$ key k_0

— Pointer i's subtree has all keys $> k_{i-1}$ and $< k_i$

— Pointer n's subtree has all keys $> k_{n-1}$

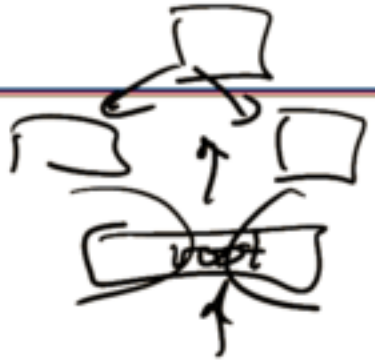
B-Trees: Definition

Definition: B-Tree of Order M (*a la* D. Comer[♮])

- Each node contains at most 2M keys (⇒ 2M+1 pointers)
- Each node (except the root) must contain at least M keys.
- A non-leaf node node has at least 2 children
- All leaf nodes are at the same level
- A non-leaf node with n keys has exactly n+1 children.

[♮] Comer, D. "The Ubiquitous B–Tree," ACM Computing Surveys 11(2), June 1979, pp. 121-137.

B-Tree: Insertion (1 / 2)



- Find the leaf node that should contain the new key value
- If leaf has capacity, insert the key into it.

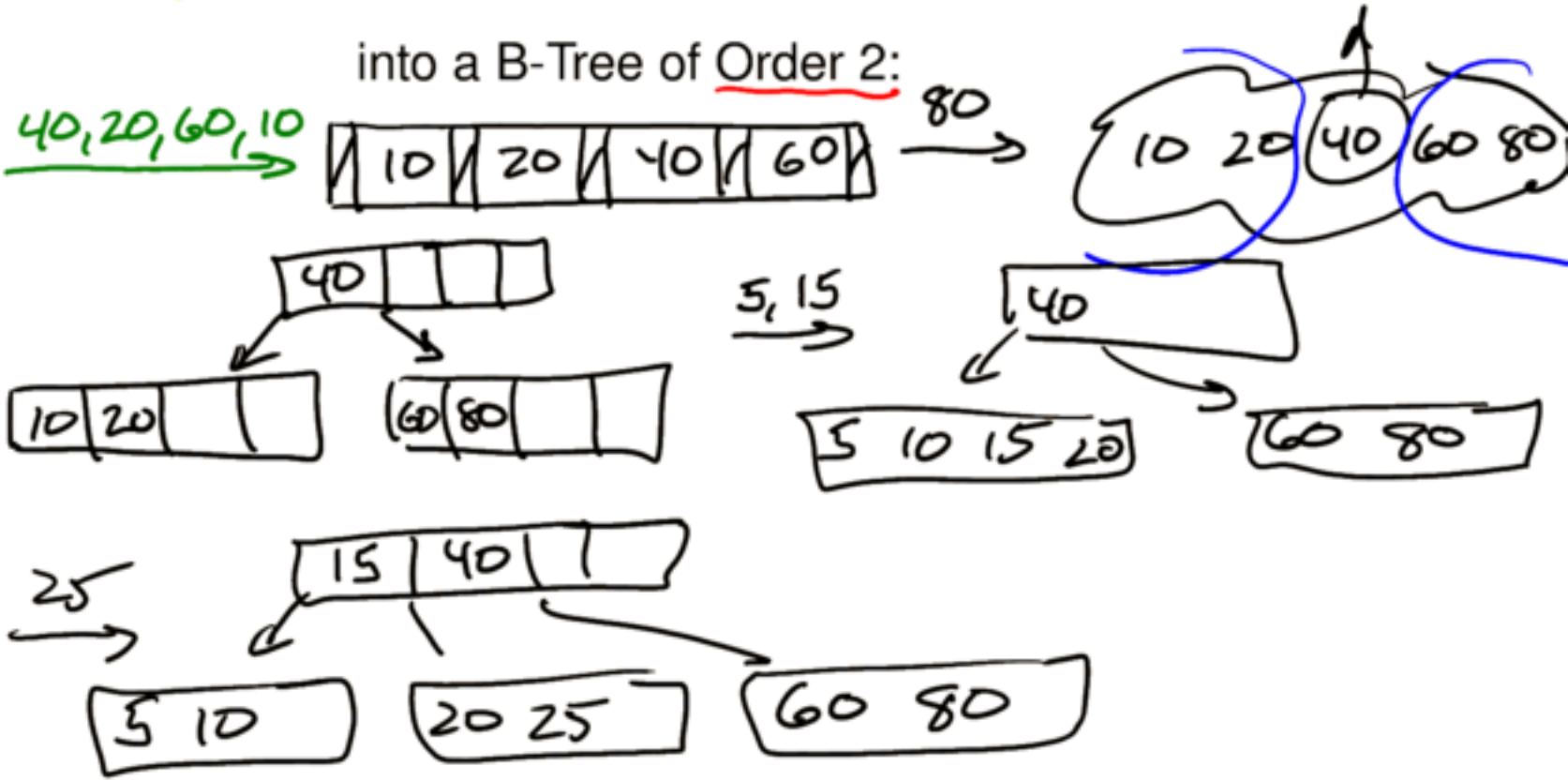
  Otherwise: (when leaf node has 2M keys)
  - Form a set of the leaf's keys plus the insertion key
  - Promote the set's median value to the parent
  - Create two nodes to hold the key values that are $<$ and $>$ the median, respectively.
  - Attach nodes as children on either side of the median

2M+1 keys

B-Tree: Insertion (2 / 2)

Example: Insert 40, 20, 60, 10, 80, 5, 15, and 25

into a B-Tree of Order 2:



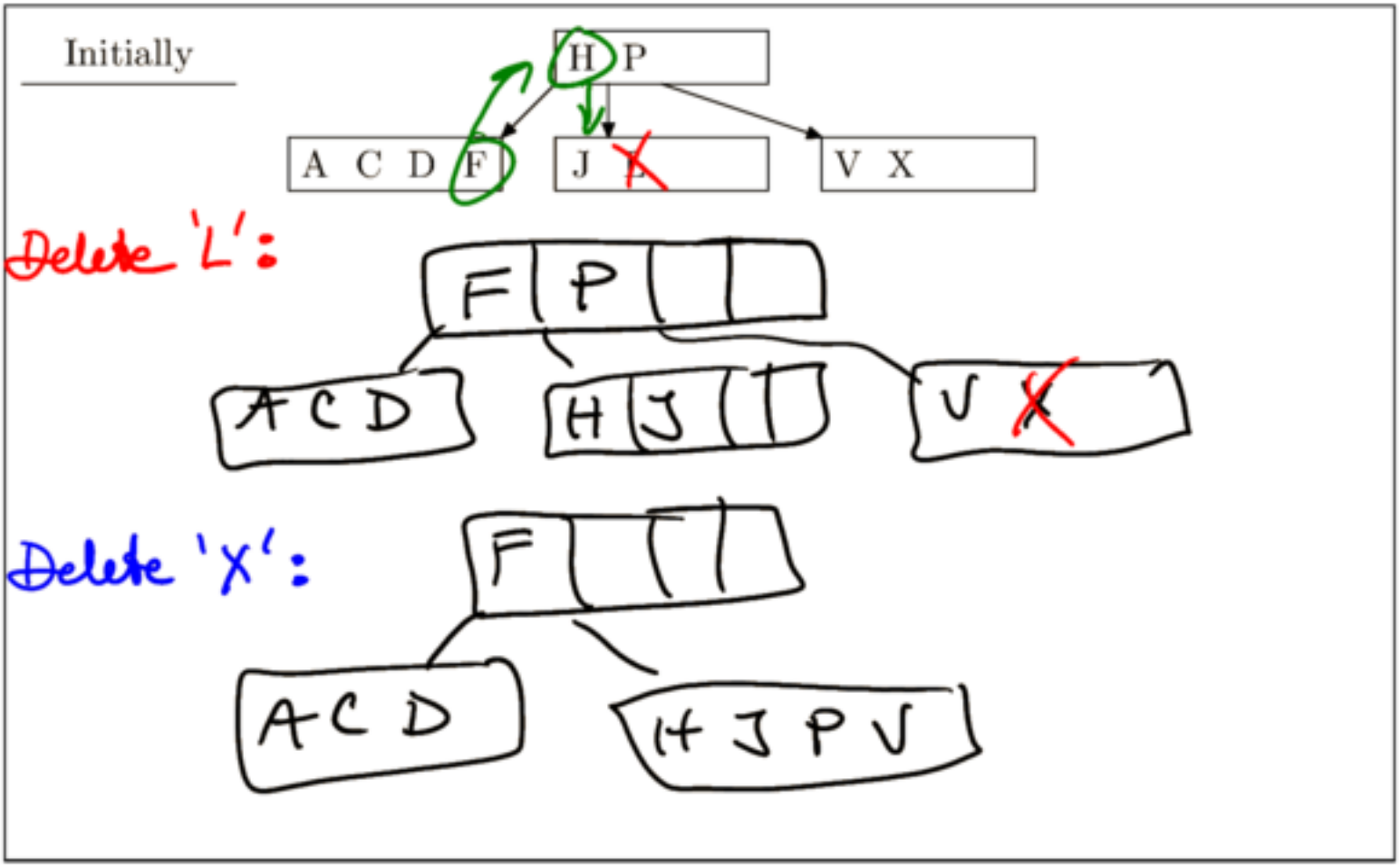
B-Tree: Deletion (1 / 2)

When a deletion leaves a node under-full:

- If the under-full node is a leaf node:
  - If a neighboring sibling has above-minimum occupancy, borrow:
    - Move separating value from parent to under-full node
    - Move appropriate value (smallest / largest) from neighbor to parent
  - Otherwise, concatenate:
    - Merge node, a neighboring sibling, and the parent's separating value into one node
    - (Note that this can leave the parent under-full, so recurse!)
- Otherwise, the under-full node is an internal node:
  - Replace deleted key with its inorder predecessor or successor
  - Recurse if necessary

B-Tree: Deletion (2 / 2)

Example(s): Still assuming M = 2



B-Tree: Capacity

What is the key capacity of a B-Tree of Order M?

Example(s): Order 128

First Level (the Root): 256 keys

Second Level: 257 nodes * 256 keys
= 65,792 keys

Third Level: 257^2 nodes * 256
= 16,908,544 keys

Total = 16,974,592 keys.

B-Tree: Order Determination

The Idea: Select order to best fit disk block capacity

Remember: A node of a B-Tree of Order M can hold

$2M$ keys and $2M + 1$ pointers

Example(s): Assume 8K bytes/block, keys of 12 bytes, ptrs of 8 bytes

8192

The amount of storage required by a node, in terms of M:

$$(2M)(12) + (2M+1)(8) \leq 8192$$

$$M \leq 204.6$$

So $M = 204$

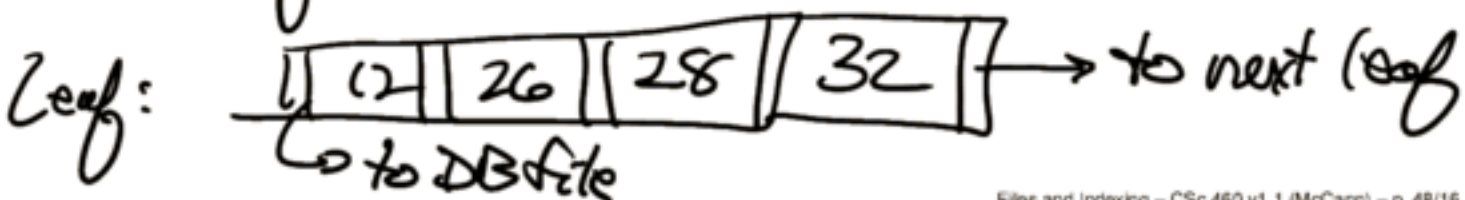
Announcements

- Program #2 is due Wednesday @ the start of class
  - Submit your code via turnin to the cs460p2 folder
- Up next: Homework #1
  - Assigned Wednesday, due next Wednesday
- Program #1 was graded & scores emailed to you over the weekend.
  - Have grading concerns? Start by directing your questions to Priya & Aayush. If you reach an impass, come talk with me about it.

B^+–Tree: A B-Tree for Indexing

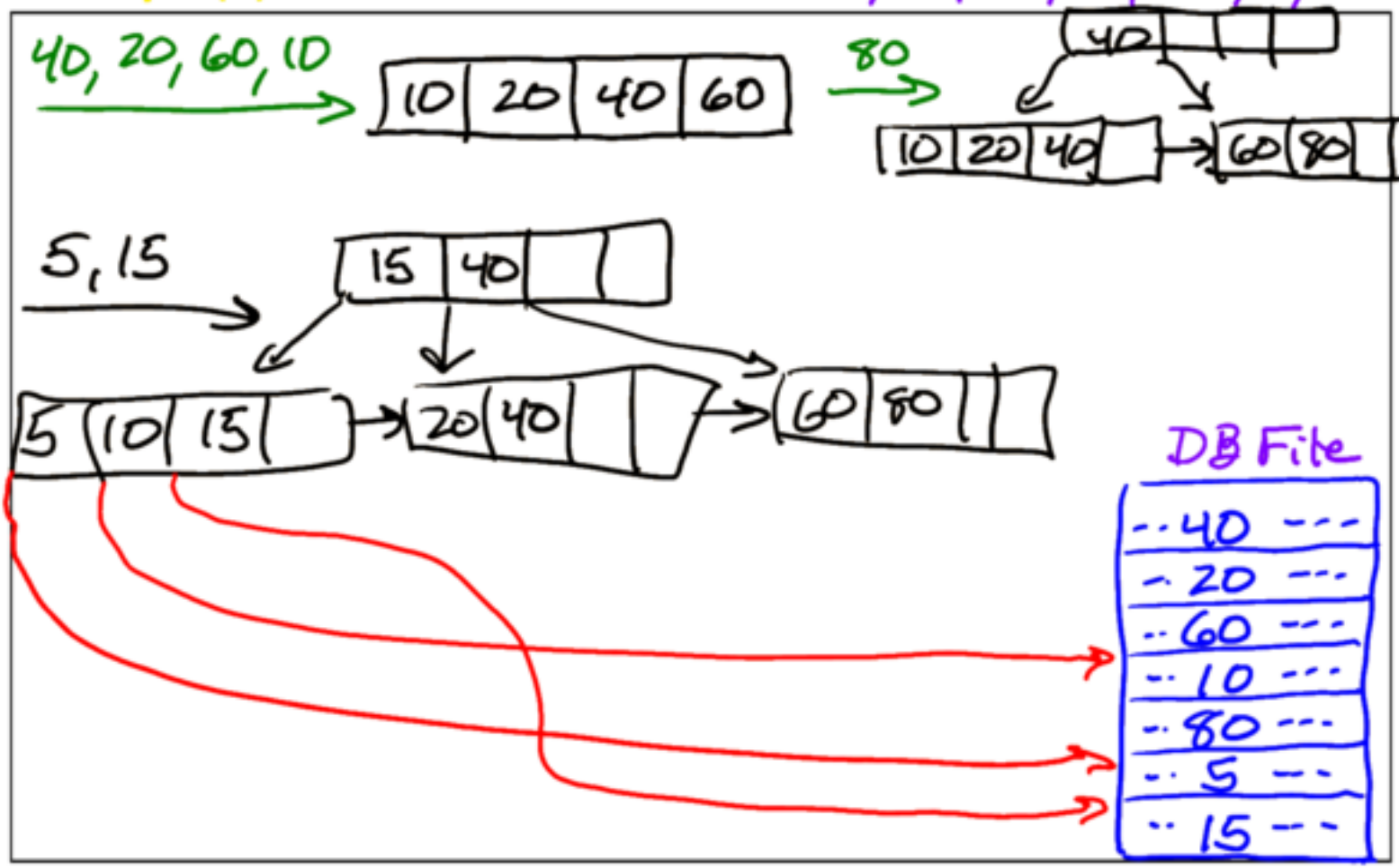
Like a B-Tree, but:

- All of the keys are stored in leaf nodes
- Copies of some keys occupy internal nodes (to maintain the tree structure)
- Leaf nodes are linked sequentially (L→R) to form the sequence set
  - facilitates linear searching
  - requires no additional pointers



B^+–Tree: Insertion

Example(s): Order 2. Insert 40, 20, 60, 10, 80, 5, and 15.



B^+–Tree: Advantages and Disadvantages over B-Trees

Advantage(s):

- Has built-in disk pointer space in leaf nodes
- Supports exact-match ("find 26") and range ("Find 20-60") queries.

Disadvantage(s):

- "waste" of storage in internal nodes (but a tiny amount)
- insertions / deletions are a little more complex.

Topic 4:

E-R

DB Design and the Entity–Relationship Model

Review of File Schemata

- Recall: Fields $\Rightarrow$ Records ($\Rightarrow$ Blocks) $\Rightarrow$ Files
- A record represents a real–world item or concept

  Example: A student record in a grading program
- A basic DB file's records all have the same construction

  (Same fields, same types, same order)
- Identification:

  — Fields: By assigned name

  — Records: By primary key
- Together, these items define the file's schema

Date's Supplier-Part-Project Schema

Also see the SPJ Schema handout!

Used by C. J. Date in his papers and textbooks.

Consists of four files:

Supplier (S)	S#	Sname	Status	City	
Part (P)	P#	Pname	Color	Weight	City
Project (J)	J#	Jname	City		
SPJ	S#	P#	J#	Qty	

Notes:

- We underline the PKs
- All fields of a composite key are underlined.

Three Tangents

- These are topics of importance to the creation of file schemas.
- They need to be introduced sometime; might as well be now!
- They are:
  - Nulls
  - Foreign Keys
  - A Few Types of Data Integrity

Nulls

Definition: Null

A null is a marker that indicates that a field does not have a value.

Notes:

- The phrase "null value" is contradictory here.
- Nulls are useful when values are not known.
  Ex: Students w/o a local address
- Nulls are somewhat controversial
  - Relational Model is based on logic
  - But they are useful!

Foreign Keys

Definition: Foreign Key

A field in one file whose values are drawn from the PK field of (usually another) file.

Notes:

- Unlike PKs, FKs may be null.
- Every FK value must also be a PK value
- Ex: In SPJ schema, the SPJ table holds 3 FK fields.

A Few Types of Data Integrity

Foreign keys are essential to three types of data integrity:

1. Key Constraint
   - PKs and FKs are designed into a DB.

2. Entity Integrity
   - No PK field may be null (b/c PKs are record IDs)

3. Referential Integrity
   - This is the name for the idea that an FK value must refer to an existing PK value.

⇒ All take effort to enforce.

DB Design: Overview

- Very similar to software development processes
- Everyone has his/her own n step design process
  - The one we'll present is rather generic
- Some ideas to keep in mind:
  - Any design process is iterative
  - Processes can be categorized as being either . . .
    - top-down vs. bottom-up, or
    - data-driven vs. function-driven, or
    - . . .

DB Design: Phases 1 & 2

Phase 1: Requirements Analysis – Learn what the customer needs!

... by asking lots of questions!

Phase 2: Conceptual Design – create the conceptual schema

- the high-level description of the DB's structure
- includes data requirements, data relationships, data constraints,
- does <u>not</u> include implementation details
- example modeling tool: The E-R Model

DB Design: Phases 3 & 4

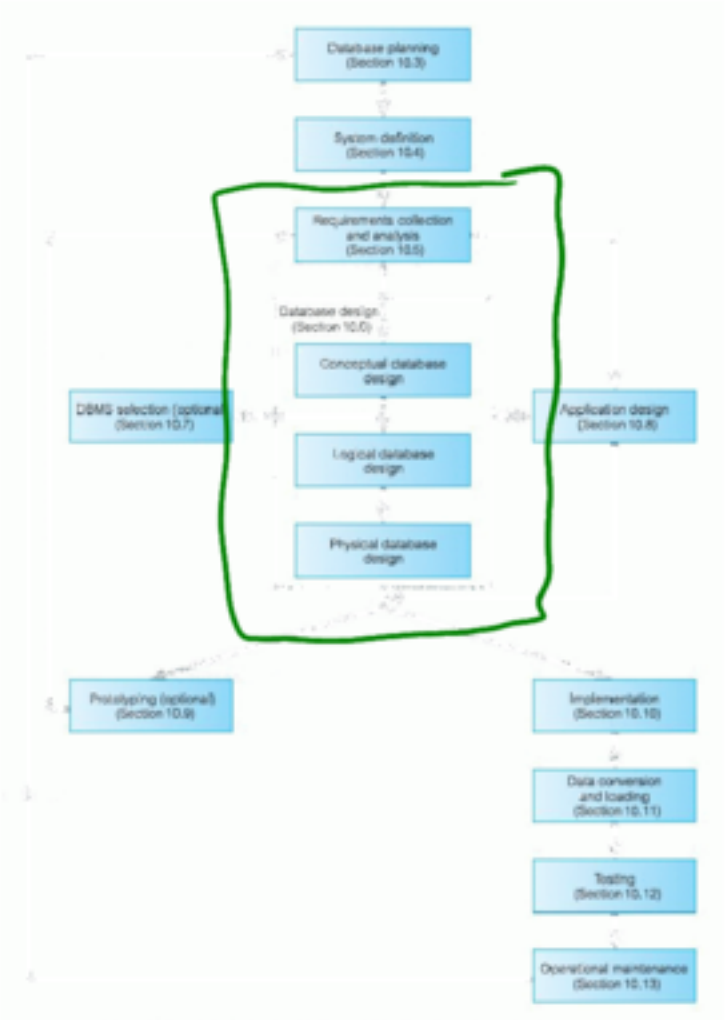
Phase 3: Logical Design – convert conceptual design into a DBMS' implementation data model (See Topic 5)

- Ex: The Relational Model
- Conversion includes data normalization

Phase 4: Physical Design – specify the DB's storage requirements

- Such as: indices, record order, data types
- all based on expected usage of the DB. (Learned in Phase 1.)

DB Design: In Context



Credit:
Connolly/Begg, 6/e,
Figure 10.1, p. 300.

Announcements

- Program #2 is due now!
  - Submit files using turnin to cs460p2 folder.
- Handout: Homework #1 ← Note: #9's (13.7's) topic will be covered Monday.
  - Pay heed to the directions:
    - Submissions must be PDFs (yes, you can write answers by hand & scan to PDF)
    - Upload your PDF to Gradescope.
  - you can use at most 1 late day on homeworks because ...
  - we will be making solutions available 24 hours after the due date/time, to help you study for ...
- Exam #1 is two weeks from today (Oct. 5th).

What are 'Entities'?

Definition: Entity

An entity is an instance of a general classification

DBs store representations of entities

Example(s):

Each of you is an instances of the class 'human'!

Lincoln's Gettysberg Address is a speech instance.

What are 'Relationships'?

Credit: "Mother Goose and Grimm" by Mike Peters, 2009– 02–12

Definition: Relationship

A relationship is an association between sets of entities.

Example(s):

Instructors teach classes.
Companies visit career fairs

One-to-One Relationships

Definition: One-to-One Relationship

In a binary relationship between two entity sets A and B, an entity from A is associated with at most one entity from B, and an entity from B is associated with at most one entity from A.

Symmetric

Example(s):

Employee manages Department

Patient — Vaccine Syringe

One-to-Many Relationships

Not Symmetrical!

(a.k.a. Many-to-One Relationships)

Definition: One-to-Many Relationship

In a →, one entity from A is associated with 0 or more entities from B, but each from B is associated with at most one from A.

Example(s):

Professor – Department (if multiple appts are not possible)

Barber – Client

(Biological) Father – Child

Many-to-Many Relationships

Definition: Many-to-Many Relationship (M-to-N)

A is a pair of 1-to-N relationships.

(Back to symmetry)

Example(s):

Employee – Project

Patient – Doctor

Other Varieties of Relationships

This list is by no means exhaustive! Some others:

- 1:1, 1:N, M:N with Varied Multiplicities

  Ex: Zero or One vs exactly one

- Ternary (a.k.a. 3-Way, Degree 3)

  Ex: SPJ Schema

  Ex: Course – Time – Room

- Recursive (a.k.a. Cyclic)

  Ex: Supervise : Employee – Employee

  — there are more! —

The E-R Model: Origins

- First proposed by Pin-Shan (Peter) Chen in a 1976 paper
  - — Extended many times since

    Example: Enhanced E-R (E-E-R) Model
  - — Has an annual conference devoted to it

    (Int'l Conf. on Conceptual Modeling)
- Easily the most popular conceptual model in use today
- Many of its ideas appear in the Unified Modeling Language (UML)
- Diagrammatic variants abound

An E-R Example (1 / 6): A Bank Database

Description:

Consider a (very) simple database for a bank. We need to store information about the bank's customers. Of course, the customers have accounts with the bank, and they perform transactions on those accounts.

Question: What are the entity sets for our database?

Customer
Account
Transaction

Hint: Use singular names!
Bank as an entity set? (NO)

An E-R Example (2 / 6): Fields

Question: **Which pieces of information do we need for each entity set?**

Customer: Name, SSN, CID, Address, ...

Account: Acct#, Balance

Transaction: Date, TID, Type, Amount, ...

Notes:

- A weak entity set is one whose existence depends upon another entity set. Ex: Transaction
  - symbol: Dashed underline is a partial key
- What Customer & Account are strong entity sets (have good primary keys)

An E-R Example (3 / 6): Relationships

Question: **Which relationships connect these entity sets?**

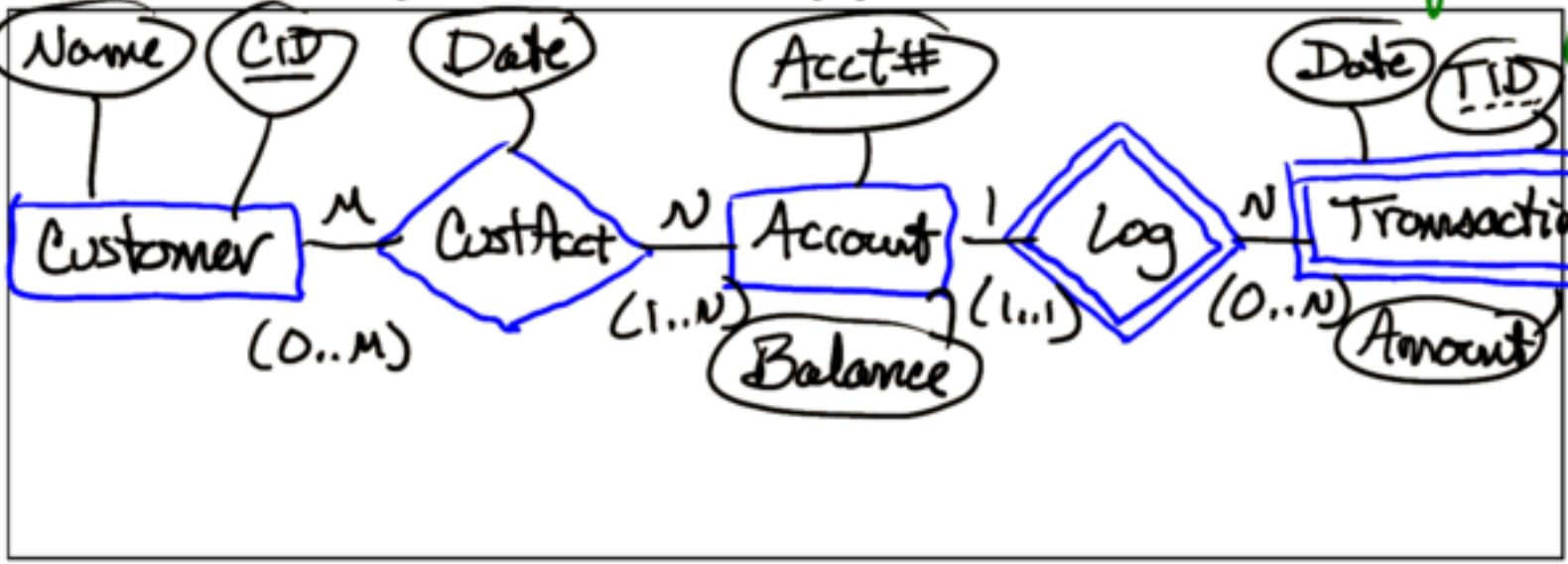
(1) Customer - Account (Cust Acct)
Cardinality: M:N
Attribute: Date of Creation

(2) Account – Transaction (Log)
Cardinality: 1:M

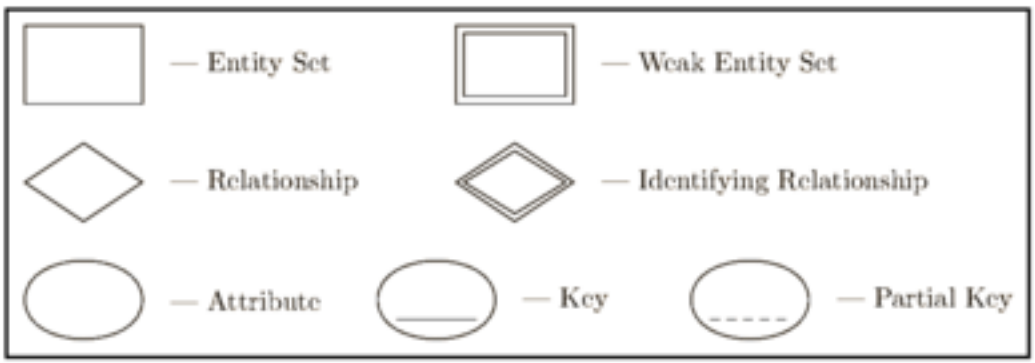
⇒ The relationship supporting a weak entity set is called an identifying relationship.

An E-R Example (4 / 6): Diagram (Chen's Notation)

Question: **Can you draw a lovely picture of all of this?**



Legend:

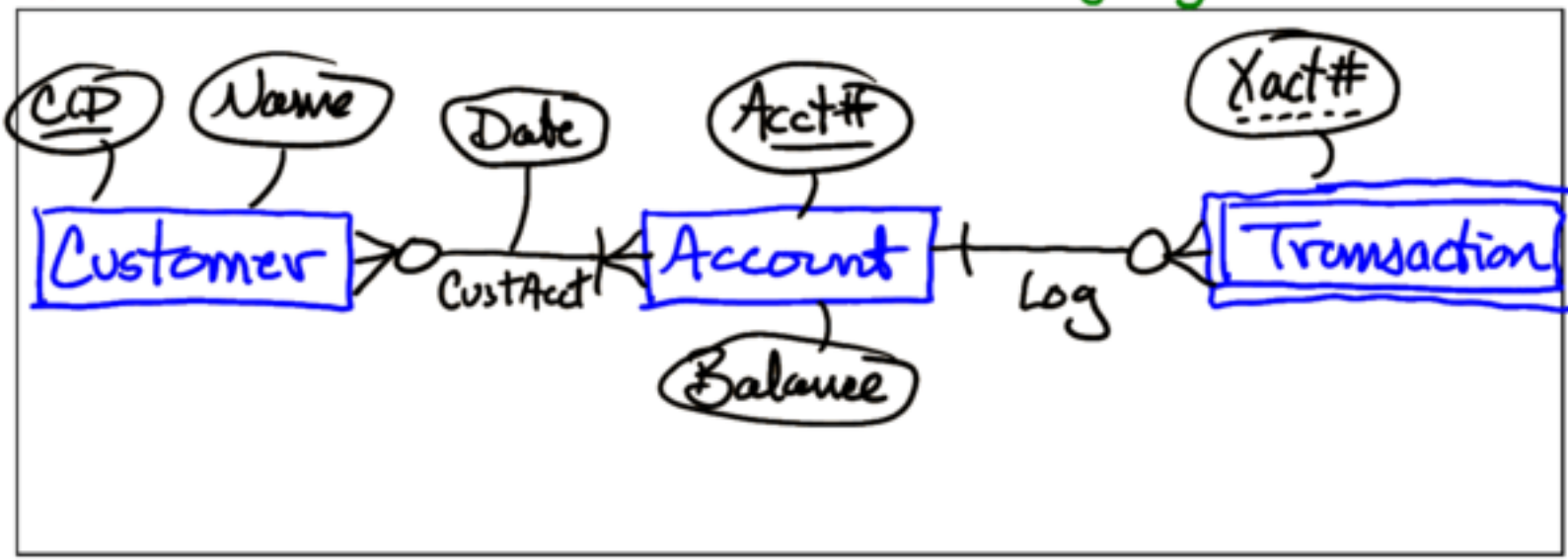


Announcements

- Homework #1 is due @ the start of class on Wednesday.
  - Remember, submit your highly-legible PDF to Gradescope!
  - You can use a late day, but only one.
  - Answers will be posted after 3:30pm Thursday
- Exam #1 is Wednesday, Oct 5th.
  - Topics: 1-5 (hopefully!)
  - Sorry, no sample exam & no review session. (We're confident that you know how to study for exams by now!)
- We are behind pace, which isn't unusual, but is a problem...

An E-R Example (5 / 6): Diagram (Crow's Feet Notation)

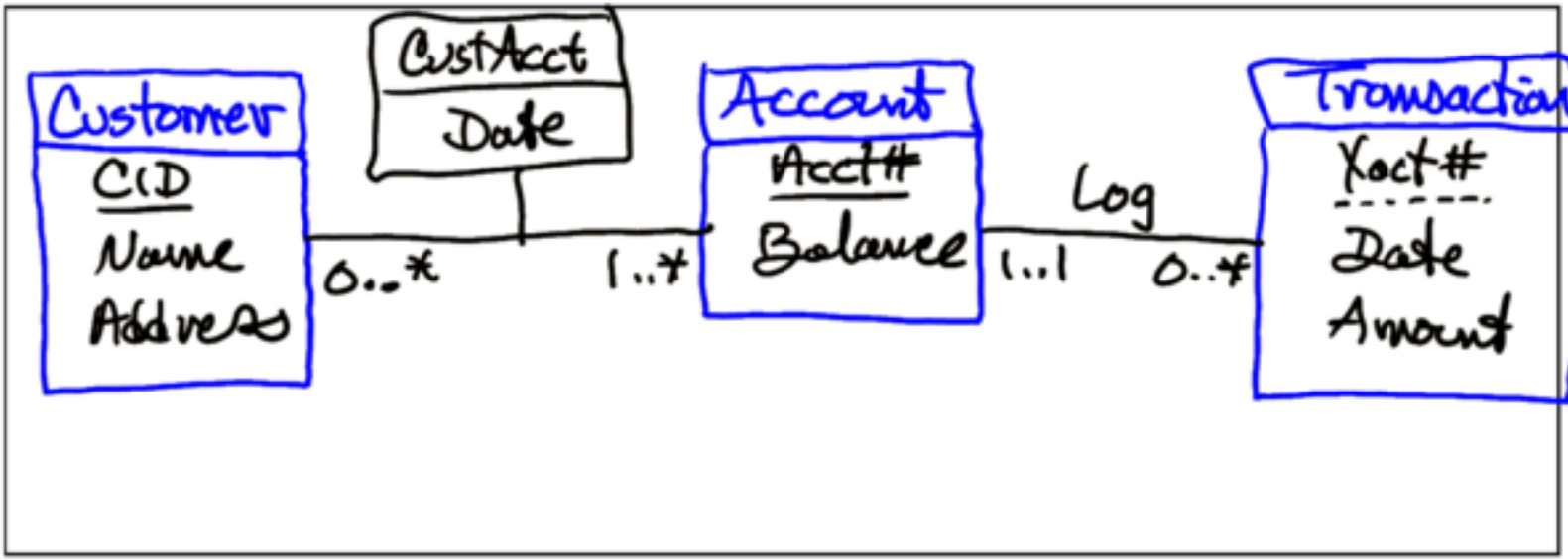
Question: **Is there another notation?** Sure; why not?



Legend (not completely standardized . . .):

An E-R Example (6 / 6): Diagram (UML Notation)

Question: **Doesn't UML include these concepts, too?** Yup!



Notes People mix and match from these styles
Lots of other notation options on each of these.

Another E-R Example: Faculty

Try it!

Description:

University faculty members teach classes that are offered by departments. Faculty are members of departments. Each department has a chairperson.

Question: What are the entity sets and relationships?

E-R Modeling Rules of Thumb

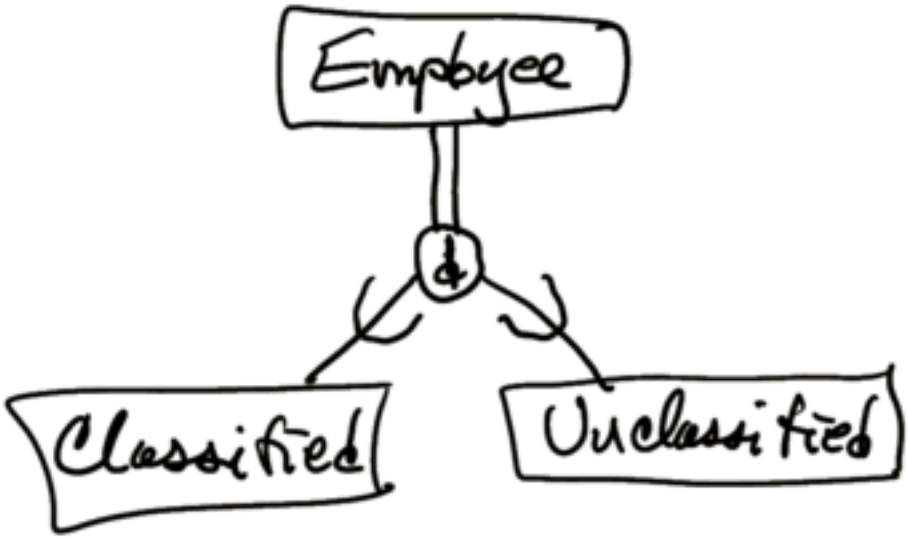
- Choose singular (v. plural) names for entity sets
- Naming relationships can be a challenge; concatentation of the names of participating entity sets is an option
- If you can't find a candidate key, perhaps the entity set is weak
  - If so, remember that the relationship is *identifying*.
- Mixing & matching notation is common
- Make your model as informative as possible

Enhanced E-R Model: Motivation

- The basic E-R Model was designed for 'business data'
  - Basically, text and numeric fields
- Now that computers are more common, more capable, and used for a wider variety of purposes, additional representational power is required to model user concepts.
- Generally, this is called *semantic modeling*.
- Some semantic modeling suggestions have been added to E-R modeling

EER: Specialization / Generalization ("is-a")

Consider inheritance in an O-O programming language . . .



Manager ⊂ Employee

═══ means "total specialization" ——— means "partial ..."

d means 'disjoint', o means 'overlap is ok', u means 'union'

EER: Aggregation ("has-a")

Several types:

1. An entity is formed from a collection of attribute values

   Ex: A person "has-a" name, id#, . . .

2. An entity formed from other entities

   Ex: A car is engine, tires, doors, . . .

3. An entity formed from a relationship to a relationship

   Ex: A job interview (relationship between Company and Applicant) resulting in a job offer

   Notation is often just a line between relationship diamonds.

Topic 5:

Implementation Data Models

A Wee Bit O' History

There are four data models of note:

1. Hierarchical (early 1960s, IBM)
2. Network (early 1970s, CODASYL/DBTG)
3. Relational (early 1970s, Codd @ IBM)
4. Object (???)

Notes: All need "tricks" to handle M-to-N rels.

A data model has (a) data structure and (b) operations

⇒ like an ADT.

Hierarchical Model: Background and Ideas

Background:

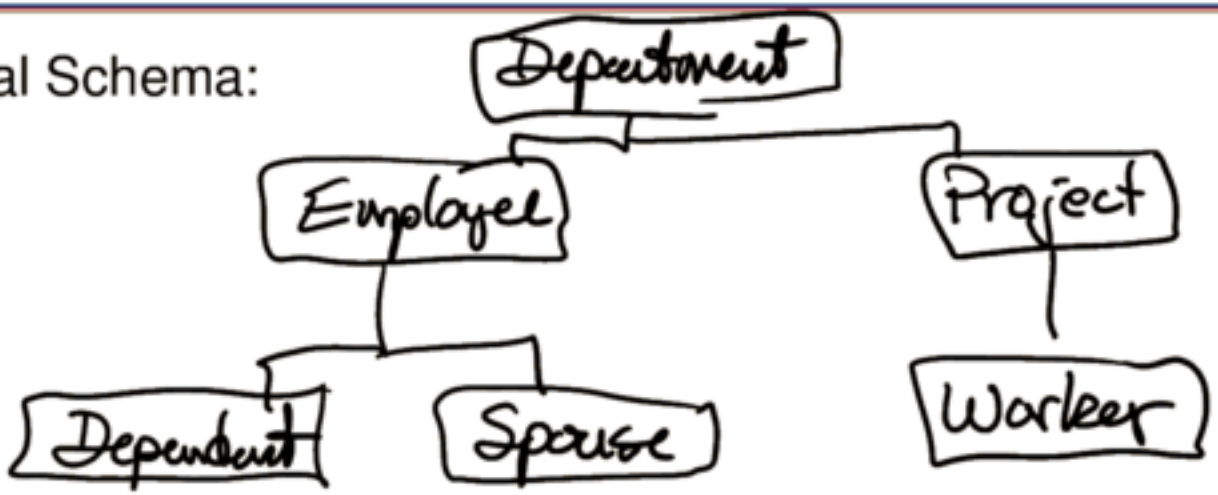
- John F. Kennedy, May 25, 1961: '. . . man on the moon . . .'
- Rockwell needed to organize parts for the Apollo CM & SM
- IBM created IMS (Information Management System) in 1968
  - original name: ICS/DL/I; thankfully renamed in '69
  - both used and introduced the Hierarchical Model
  - still sold today!

Ideas:

Schema is tree structured
only 1-N relationships are directly supported.
- 1 parent, many children.

Hierarchical Model: Terminology

Sample Logical Schema:

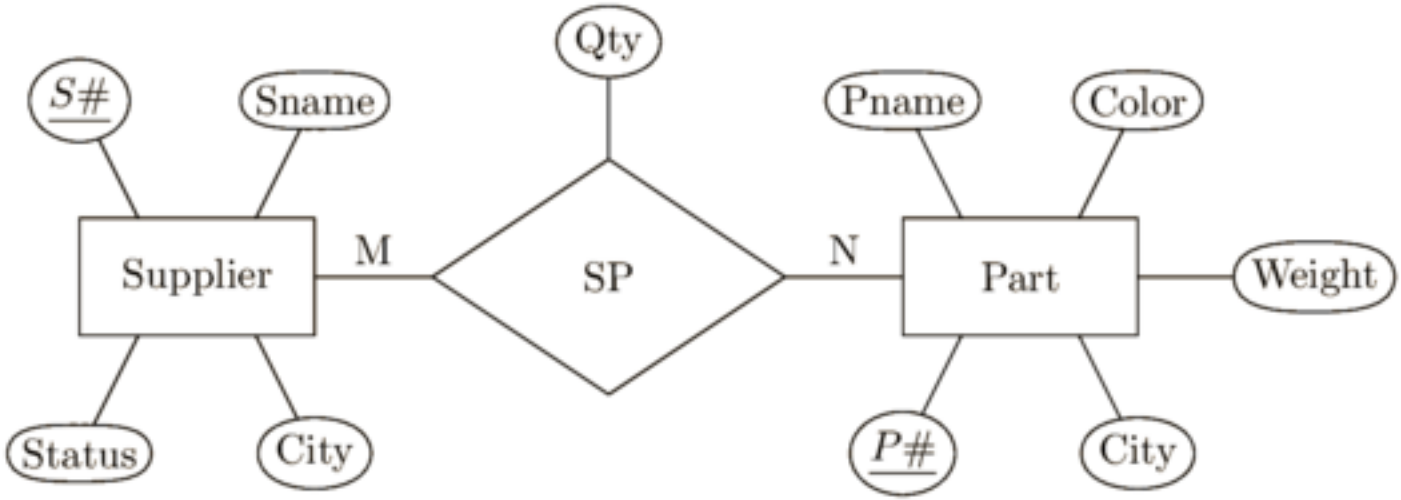


Terminology:

E-R Model		Hierarchical Model
Entity Set	≡	Segment Type
Entity	≡	Segment Occurrence (or Instance)
Attributes	≡	Fields or Data Items
Relationships	≡	PCRs - Parent-Child Relationships

Hierarchical Model: Supplier–Part Schema

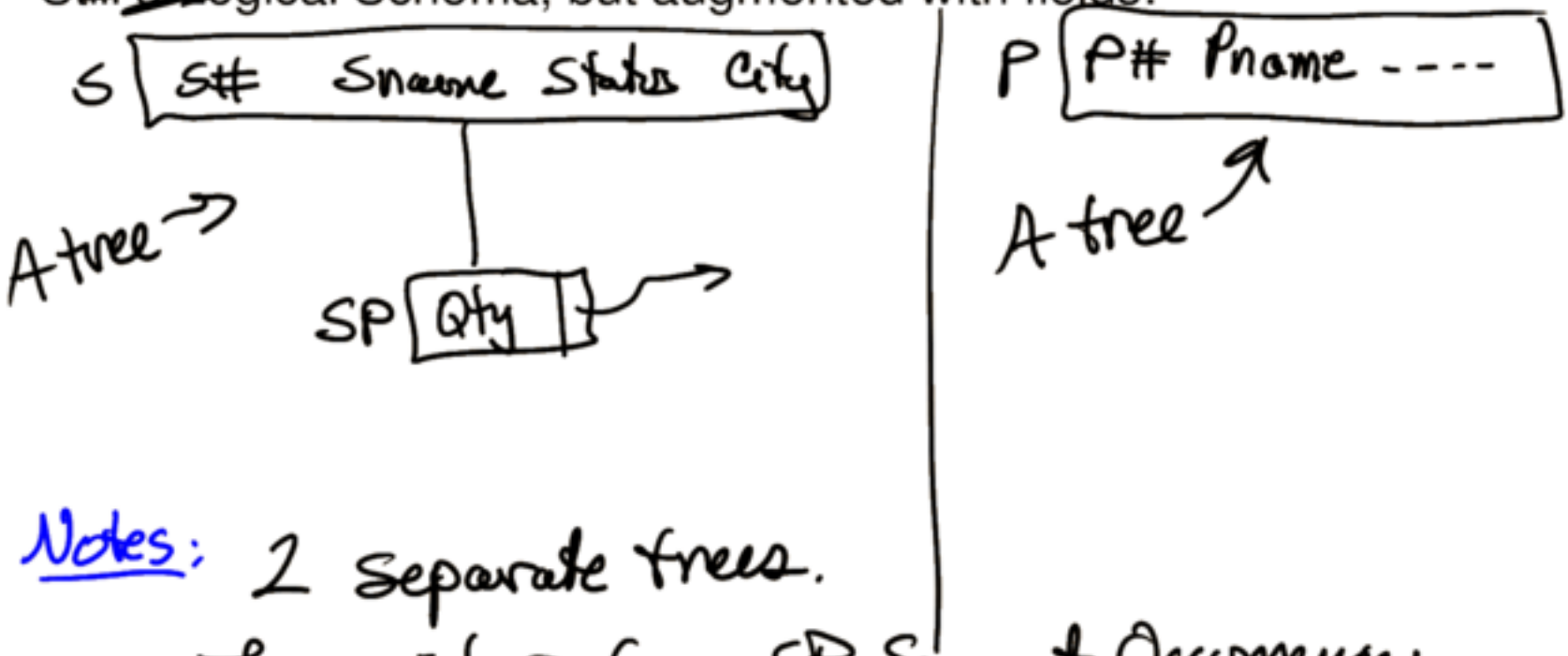
Consider this subset of Codd's SPJ schema:



– Don't need Project here.

Hierarchical Model: M:N Relationships (1 / 3)

~~Still a Logical~~ Physical Schema, but augmented with fields:

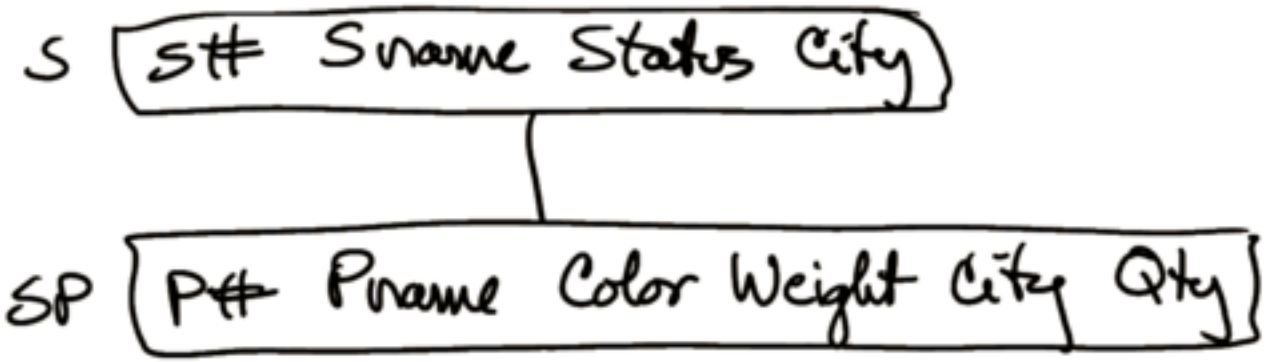


Notes: 2 separate trees.

The pointers from SP Segment Occurrences will point to seg. occurrences of P

Hierarchical Model: M:N Relationships (2 / 3)

Logical
~~Physical~~ Schema:

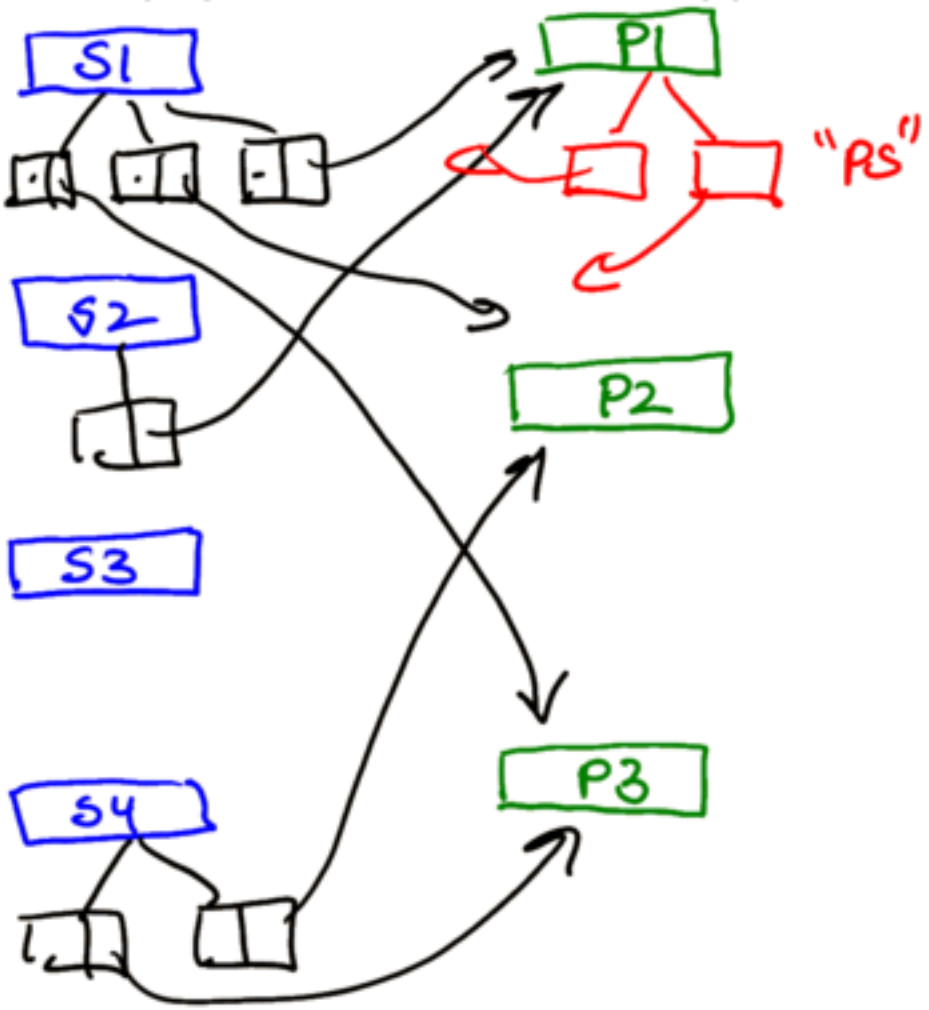


Notes: SP is the merger of the SP & P fields

It appears that if n suppliers supply a part, we will need n copies of that part.

Hierarchical Model: M:N Relationships (3 / 3)

The physical schema for Supplier–Part w/ sample data:



Questions:

1. Which parts can S1 supply?

2. Which suppliers can supply P3?

Network Model: Background and Ideas

- Created in the early 1970s by CODASYL's

  (Conference/Committee on Data Systems Languages)

  DBTG (Database Task Group)
- Goal: A standard theory of DB systems.

  Origin of the ideas of DML and DDL
- Became an ISO standard in 1987 (ISO 8907:1987)

  (And was withdrawn in 1998!)
- Graph–based instead of tree–based

Network Model: Terminology

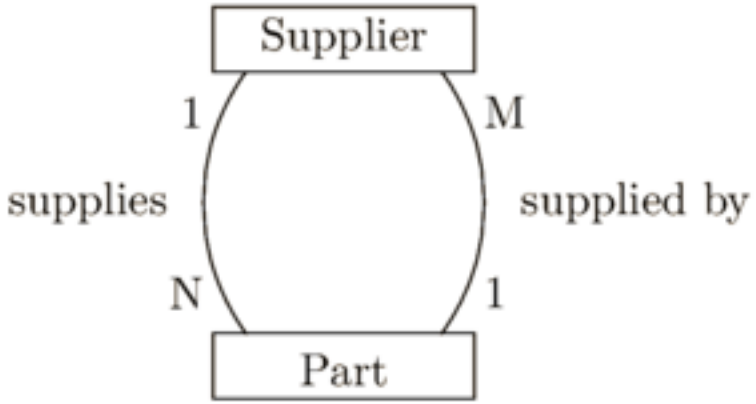
A Sample Logical Schema:

Terminology:

E-R Model		Network Model
Entity Set	≡	Record Type
Entity	≡	Record Occurrence
Attributes	≡	Data Items
Relationships	≡	Set or LinkType (orig. 'DBTG sets')

Network Model: M:N Relationships (1 / 2)

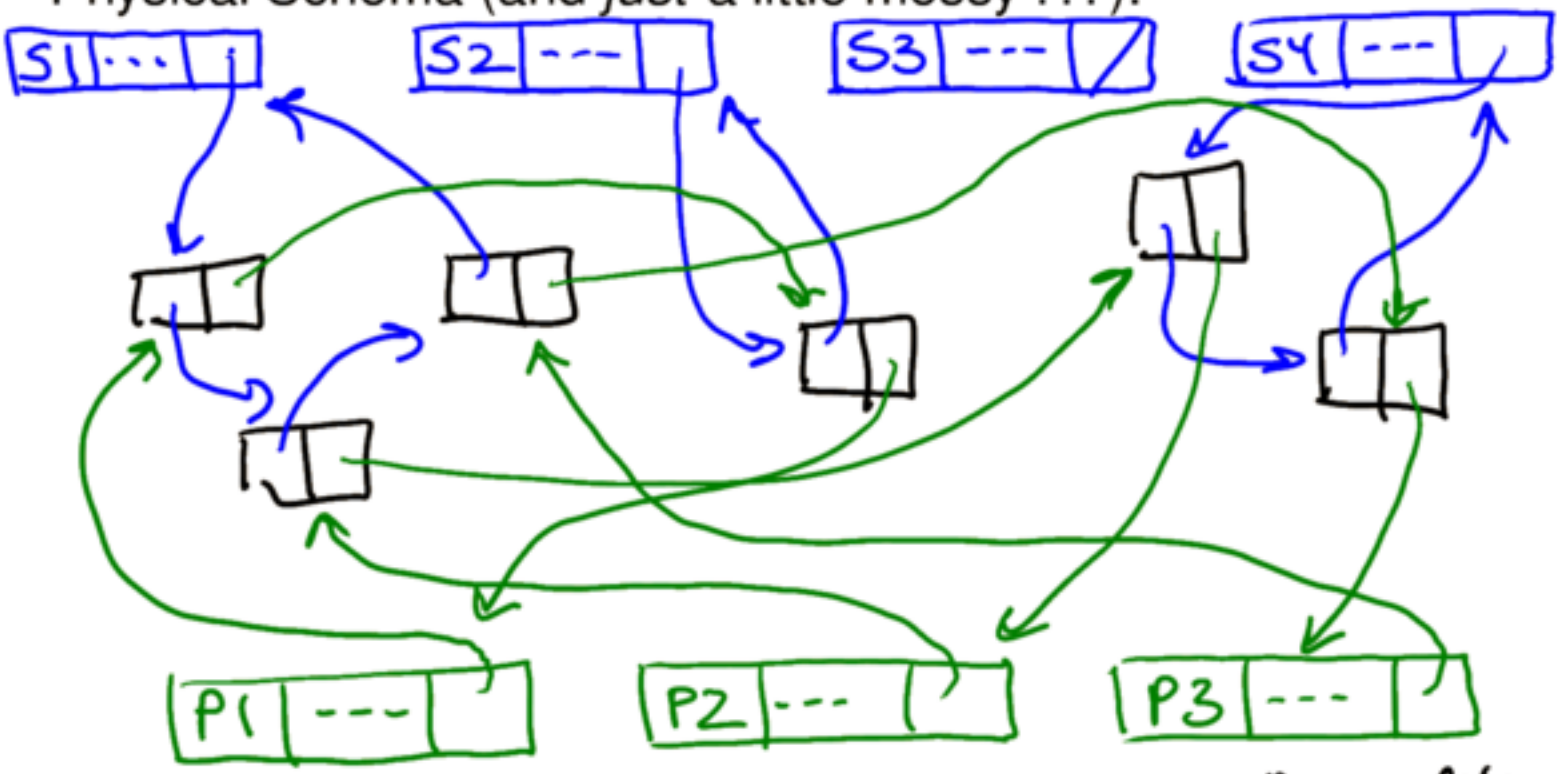
Logical Schema (of the M:N Supplier - Part relationship):



Note: An undirected graph – can travel either way.

Network Model: M:N Relationships (2 / 2)

Physical Schema (and *just* a little messy . . .):



Imagine the quantities are part of the Link Record Occurrences

① Which parts can S1 supply?

② Which suppliers can supply P3?

Relational Model: Background and Ideas

- Created by Edgar F. Codd. Famous paper:

  "A relational model of data for large shared data banks," 1970.

- Theoretical foundation: Set Theory
- Uses foreign keys instead of pointers
- No distinction between logical and physical schemas

Relational Model: DMLs

Codd proposed two types of DMLs:

1. Relational Calculus (non-procedural)
2. Relational Algebra (procedural)

SQL has elements of each

Watch for video!

Announcements

- Homework #1 is due now.
  - Late submissions can use at most one late day
  - Solutions will be posted sometime after 3:30pm tomorrow.
- Exam #1 is one week from today! (Oct 5th)
  - Topics: 1–5
- The video covering Topic 6 (Relational Calculus) is available in D2L (under Content/Lect. Recordings)
  - Today, we'll finish Topic 5 and will start Topic 7 (Relational Algebra).
- Homework #2 will be assigned Monday, due 9 days later.

Relational Model: DMLs

Partial Review

Codd proposed two types of DMLs:

1. Relational Calculus (non-procedural; "declarative")
   (Lecture Topic 6 – see video for coverage!)
2. Relational Algebra (procedural)
   (Lecture Topic 7)

Notes: Neither is suitable for practical use.

Codd's RC is now called Tuple RC, the later version is Domain RC.

Relational Model: Terminology (1 / 2)

Sample Supplier - Part Schema:

S

S#	Sname	Status	City
S1	Acme	10	Omro
S2	Fubar	10	Fisk
S3	Snafu	20	Ring

P

P#	Pname	Color	Weight	City
P1	Nut	Pink	0.2	Anton
P2	Bolt	Blue	0.9	Borea

Terminology:

E-R Model		Relational Model
Entity Set	≡	Relation (or table)
Entity	≡	Tuple (pronounce: TOO-ple)
Attributes	≡	Fields
Relationships	≡	Relationships!

Relational Model: Terminology (2 / 2)

Sample Department – Employee Schema:

PK

DEPARTMENT

DeptNum	DeptName	ManagerID	ManagerStartDate
1	Shipping	364	2001-04-01
2	Payroll	NULL	NULL
3	Billing	298	2000-11-17

FK

EMPLOYEE

Surname	GivenName	EmpNum	DeptID	Salary
Spade	Sam	786	1	48000
Trune	Joe	410	2	49500
Smith	Megan	364	1	75000
Maher	Mary	298	3	72000

cardinality ≡ # of tuples

degree (or arity) ≡ # of attributes

unary relation ≡ A relation of Degree 1

n-ary relation ≡ A relation of Degree N

Relational Model: Misc. Notes

- Order of tuples in a relation is irrelevant (Why?)

  logically!

  Relations are sets of tuples!

- Fields are single-valued by default (vs. set-valued)

  This is frequently violated in practice.

- Relationships are supported by foreign keys

  May be stored in separate relations.
  (stay tuned!)

Relational Model: 1:N Relationships

We've already seen how to do this! (Just two slides ago!)

PK

DEPARTMENT

DeptNum	DeptName	ManagerID	ManagerStartDate
1	Shipping	364	2001-04-01
2	Payroll	NULL	NULL
3	Billing	298	2000-11-17

FK

EMPLOYEE

Surname	GivenName	EmpNum	DeptID	Salary
Spade	Sam	786	1	48000
Trune	Joe	410	2	49500
Smith	Megan	364	1	75000
Maher	Mary	298	3	72000

Notes: A dept can have many employees; each employee has 1 dept.

Relational Model: 1:1 Relationships

- Just a restriction of 1:N relationships:
  - We still store a FK in the 'many' relation
  - Must constrain the field's values to be unique; two options:

(1) Insist that the FK be a Candidate Key.

(2) Create a unique index on the FK
(the DBMS enforces the uniqueness of the attr's content)

Relational Model: M:N Relationships

PK

S

S#	Sname	Status	City
S1	Acme	10	Omro
S2	Fubar	10	Fisk
S3	Snafu	20	Ring

PK

P

P#	Pname	Color	Weight	City
P1	Nut	Pink	0.2	Anton
P2	Bolt	Blue	0.9	Borea

FK FK

attr. of the relationship

SP

S#	P#	Qty
S1	P1	50
S1	P2	150
S2	P2	25
S3	P1	300

Notes:

- called a relationship relation
- necessary because won't work to store both FKs in source relations (try it!)

Object Model: Ideas

- OO programming languages have existed since Simula in 1967
- We'd like to be able to store objects in a DBMS
  - provides object persistence
  - can do it by mapping object instance fields to relational tuples, but that's clunky
- The Object Data Management Group (ODMG) defined an object-based DBMS standard
  - finished ODMG 3.0 in 2001 (and then disbanded!)

Object Model: Object DBMS Types

Two major varieties:

1. Object Oriented DBMS (OODBMS)
   - Marriage of an OOPL and a DBMS
2. Object Relational DBMS (ORDBMS)
   - A relational DBMS with added objects

Most popular
— get to leverage existing customer base.

Topic 7:

Relational Algebra

Looking for Topic 6 (Relational Calculus)? See:
- Video covering it (in D2L)
- Slides are in a separate file in the completed slides area of the class web page.

yes, Relational Calculus will be on Exam #2 and the Final Exam.

Background

- Introduced by Codd (along with the Tuple Relational Calculus)
- Relational Algebra . . . :
  - Is procedural, like most programming languages

    — we need to supply an ordering of operations
  - Would not be a good replacement for SQL in a DBMS
  - Is a good introduction to the operators provided by SQL

Relational Operators (1 / 2)

Relations are *closed* under Relational Algebra operators

- That is, they accept relations as operands, and produce relations as results.
- Example: Integers are closed under $+$ and $-$.

The eight basic Relational Algebra operators are:

$\times$	Cartesian Product	$\bowtie$	Join
$-$	Difference	π	Project
$\div$	Division	σ	Select
$\cap$	Insection	$\cup$	Union

Relational Operators (2 / 2)

The eight Relational Algebra operators can be grouped in two ways:

1. Set vs. Relational:
   - Set: $\cup$, $\cap$, $-$, $\times$, $\div$
   - Relational: σ, π, $\bowtie$
2. Fundamental vs. Derived:
   - Fundamental: σ, π, $\times$, $\cup$, $-$
   - Derived: $\cap$, $\bowtie$, $\div$

The Fundamental Operators

- Select (σ)
- Project (π)
- Cartesian Product ($\times$)
- Union ($\cup$)
- Difference ($-$)

Select (σ, sigma) (1 / 2)

- A unary (single argument) operator
- Chooses full tuples from a relation based on a condition
- Form: $\sigma_{condition}(relation)$

Example(s):

List all information of the employees in department #5:

$\sigma_{dept\# = 5}(Employee)$

Who are the active suppliers in Paris?

$\sigma_{city = 'Paris' \wedge status > 0}(S)$

Select (σ, sigma) (2 / 2)

Notes:

- Conditions may be as complex as is necessary
- Select is commutative:

$$\sigma_A(\sigma_B(\mathsf{r})) \equiv \sigma_B(\sigma_A(\mathsf{r}))$$

- Cascades of selects $\equiv$ conjunction in a single select:

$$\sigma_A(\sigma_B(\sigma_C(\mathsf{r}))) \equiv \sigma_{A \wedge B \wedge C}(\mathsf{r})$$

Project (π, pi) (1 / 2)

Pronunciation: PRO-ject (not pro-JECT, not PRAH-ject)

- Also a unary operator
- Chooses named columns from a relation
  - Resulting group of tuples may include duplicates . . .
  - . . . which we drop to preserve entity integrity
- Form:

Project (π, pi) (2 / 2)

Example(s):

Find the names & salaries of the employees in department 5:

$$\pi_{\text{givenname, surname, salary}}(\sigma_{\text{dept\#=5}}(\text{Employee}))$$

Alternatively:

$$\text{temp} \leftarrow \sigma_{\text{dept\#=5}}(\text{Employee})$$

$$\pi_{\text{givenname, surname, salary}}(\text{temp})$$

Cartesian Product ($\times$) (1 / 2)

- A binary operator (form: R $\times$ S)
- 'Marries' all pairings of tuples from the given relations
  - resulting cardinality = card(R) $\cdot$ card(S)
  - resulting degree = degree(R) + degree(S)

Example(s):

A

m	n
2	i
3	iv
7	x

B

o	p
3	β
7	α

A × B

m	n	o	p
2	i	3	β
2	i	7	α
3	iv	3	β
3	iv	7	α
7	x	3	β
7	x	7	α

Cartesian Product ($\times$) (2 / 2)

Example(s):

What are the names of the active suppliers of nuts?

The complete query:

$$\pi_{\text{Sname}}(\sigma_{\text{Status}>0 \wedge \text{Pname='Nut'}}(\sigma_{\text{S.S\#} = \text{SP.S\#}}(\sigma_{\text{SP.P\#} = \text{P.P\#}}(S \times (SP \times P)))))$$

For a visualization of this query step–by–step with sample data, see the handout:

"Examples of the Relation Algebra Operations σ, π, and $\times$"

Topic 6:

Relational Calculi

Meanings of 'Calculus'

- Calculus refers to any method or system of calculation
- 'Calculus' is derived from the Latin word for pebble
- Modern uses of the word include:
  - Differential Calculus: rates of change
  - Integral Calculus: limits of sums
  - Lambda Calculus: functional abstraction & application
  - Predicate Calculus: reasoning about symbolic logic

Review of First-Order Predicate Calculus (FOPC) (1 / 2)

Why? Because FOPC ⇒ Relational Calculus ⇒ SQL

FOPC (a.k.a. First-Order Logic):

- uses predicates (e.g., P(x))
  - predicate
  - subject
- subjects may be values; they may not be predicates.
- FOPC can formalize all of set theory.

Review of First-Order Predicate Calculus (FOPC) (2 / 2)

Supplied Primitives:

Variables, Logical operators, Quantifiers

Constructs of our creation:

Constants, Predicates, Functions

Example(s):

Feathers(x): x has feathers, $x \in$ Animal

Bird(x): x is a bird, $x \in$ Animal

Feathers(Eagle) $\rightarrow$ Bird(Eagle)

$\forall x$ [Feathers(x) $\Rightarrow$ Bird(x)], $x \in$ Animal

$\exists x$ [Bird(x) $\wedge \neg$Fly(x)], $x \in$ Animal

Relational Calculi: Ideas

Relational Calculi are "what, not how" languages

There are two forms:

Tuple Relational Calculus (TRC)

Domain " " (DRC)

Tuple Relational Calculus (TRC): Background

- Proposed by Codd in 1972
- So named because the variables in TRC represent tuples
- TRC queries have this basic form:

$$\{ t \mid \text{predicate}(t) \}$$

where:

t is a free (unbound) tuple variable

predicate is a wff

TRC: Query #1

Question: What is the content of the Employee Relation?

$\{ e \mid Employee(e) \}$

Recall these schemas:

DEPARTMENT	DeptNum	DeptName	ManagerID	ManagerStartDate	
EMPLOYEE	Surname	GivenName	EmpNum	DeptID	Salary

TRC: Query #2

Question: **What are the names and salaries of the people in department #5?**

$$\{ e.\text{givenname}, e.\text{surname}, e.\text{salary} \mid \text{Employee}(e) \wedge e.\text{deptid} = 5 \}$$

DEPARTMENT	DeptNum	DeptName	ManagerID	ManagerStartDate
EMPLOYEE	Surname	GivenName	EmpNum	DeptID	Salary

"5

TRC: Query #3

Question: **What are the names of the parts that can be supplied by individual suppliers in quantity > 200?**

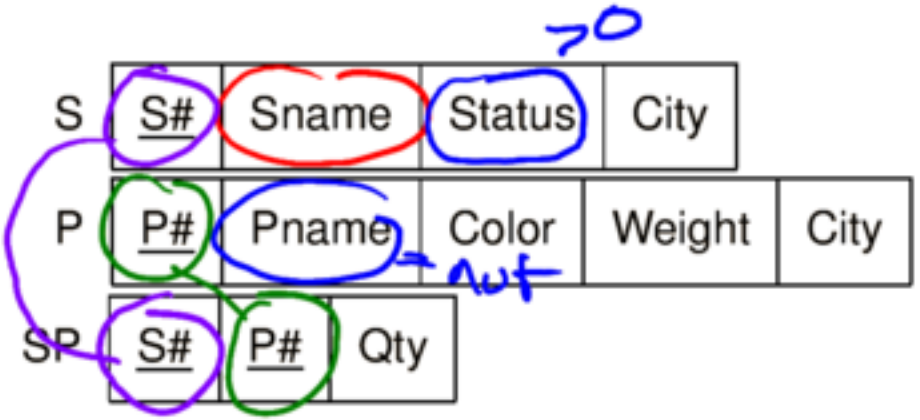
$$\{ p.pname \mid P(p) \wedge \exists(q)(SP(q) \wedge q.P\# = p.P\# \wedge q.Qty > 200) \}$$

S	S#	Sname	Status	City	
P	P#	Pname	Color	Weight	City
SP	S#	P#	Qty	= 200	

TRC: Query #4

Question: **What are the names of the active suppliers of nuts?**

$$\{s.sname \mid S(s) \wedge s.status > 0 \wedge \exists(q)(SP(q) \wedge q.S\# = s.S\# \wedge \exists(p)(P(p) \wedge p.P\# = q.P\# \wedge p.pname = \text{'Nut'}))\}$$



Aside: Expression Safety

Definition: Expression Safety

An expression is safe if the expression's domain is the source of all resulting values.

Example(s):

Safe: $\{ e \mid Employee(e) \}$

Unsafe: $\{ e \mid \neg Employee(e) \}$

No way to prevent unsafe queries.

Domain Relational Calculus (DRC): Bkgd.

- Proposed by Lacroix and Pirotte in 1977 to supply a formalism for IBM's Query By Example (QBE) product.
- In DRC, a variable represents just one field of a tuple
- DRC queries have this basic form:

$$\{ \langle abcd \ldots \rangle \mid condition(\langle abcd \ldots \rangle) \}$$

where:

$\langle abcd \ldots \rangle$ is a set of ≥ 1 unbound domain variables, one per field of the relation

condition is a DRC w.f.f.

DRC Query #1

Question: What is the content of the Employee Relation?

$\{ \langle efghi \rangle \mid \langle efghi \rangle \in Employee \}$

Note the helpful field labels! ⇓

	a	b	c	d
DEPARTMENT	DeptNum	DeptName	ManagerID	ManagerStartDate

	e	f	g	h	i
EMPLOYEE	Surname	GivenName	EmpNum	DeptID	Salary

DRC Query #2

Question: **What are the names and salaries of the people in department #5?**

$$\{ \langle fei \rangle \mid (\exists h)(\langle efghi \rangle \in Employee \wedge h = 5) \}$$

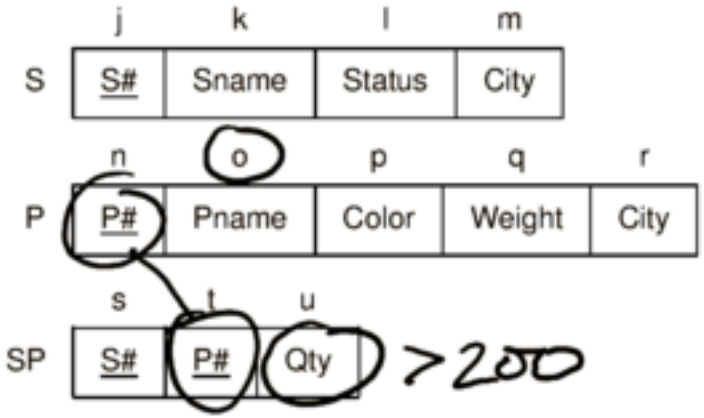
	a	b	c	d
DEPARTMENT	DeptNum	DeptName	ManagerID	ManagerStartDate

	e	f	g	h	i
EMPLOYEE	Surname	GivenName	EmpNum	DeptID	Salary

DRC Query #3

Question: **What are the names of the parts that can be supplied by individual suppliers in quantity** > 200**?**

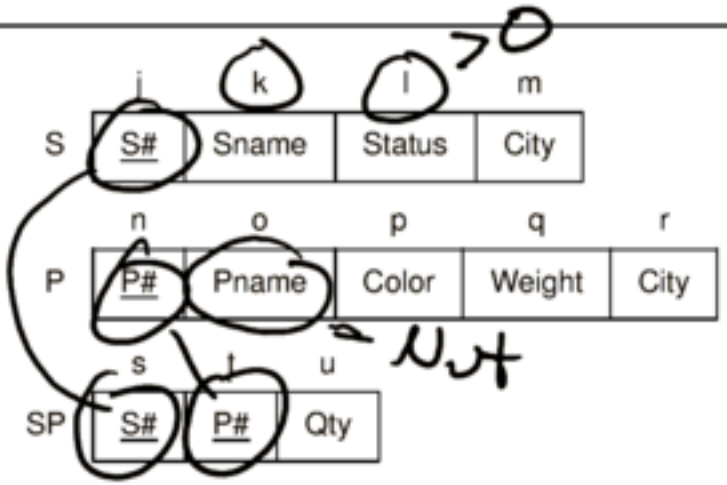
$$\{\langle o \rangle \mid (\exists n)(\langle nopqr \rangle \in P \wedge (\exists tu)(\langle stu \rangle \in SP \wedge n = t \wedge u > 200))\}$$



DRC Query #4 (1 / 2)

Question: **What are the names of the active suppliers of nuts?**

$$\{ \langle k \rangle \mid (\exists j l)(\langle jklm \rangle \in S \wedge l > 0 \wedge (\exists st)(\langle jtu \rangle \in SP \wedge j = s \wedge (\exists no)(\langle topqr \rangle \in P \wedge n = t \wedge o = \text{'Nut'})))\}$$



Shortcut is optional.