UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 3380 Spring 2024

Database Systems

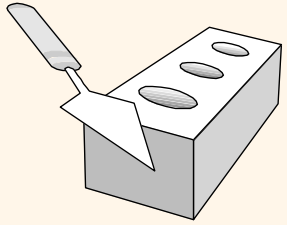
M & W 4:00 to 5:30 PM

Prof. Victoria Hilford

PLEASE TURN your webcam ON (must have)

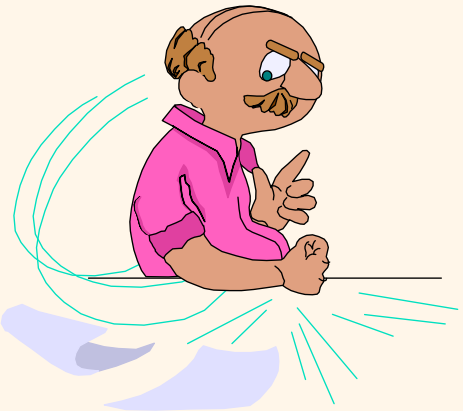
NO CHATTING during LECTURE

VH, unhide Section 14: SET 3 STORAGE II



COSC 3380

4 to 5:30



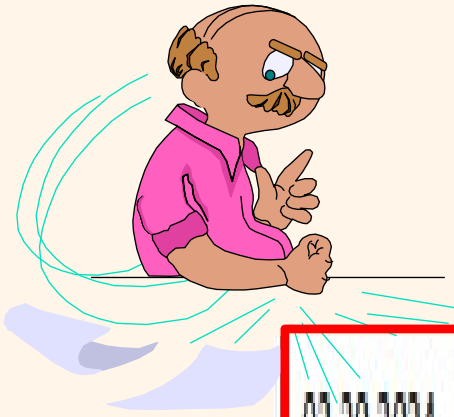
PLEASE
LOG IN
CANVAS

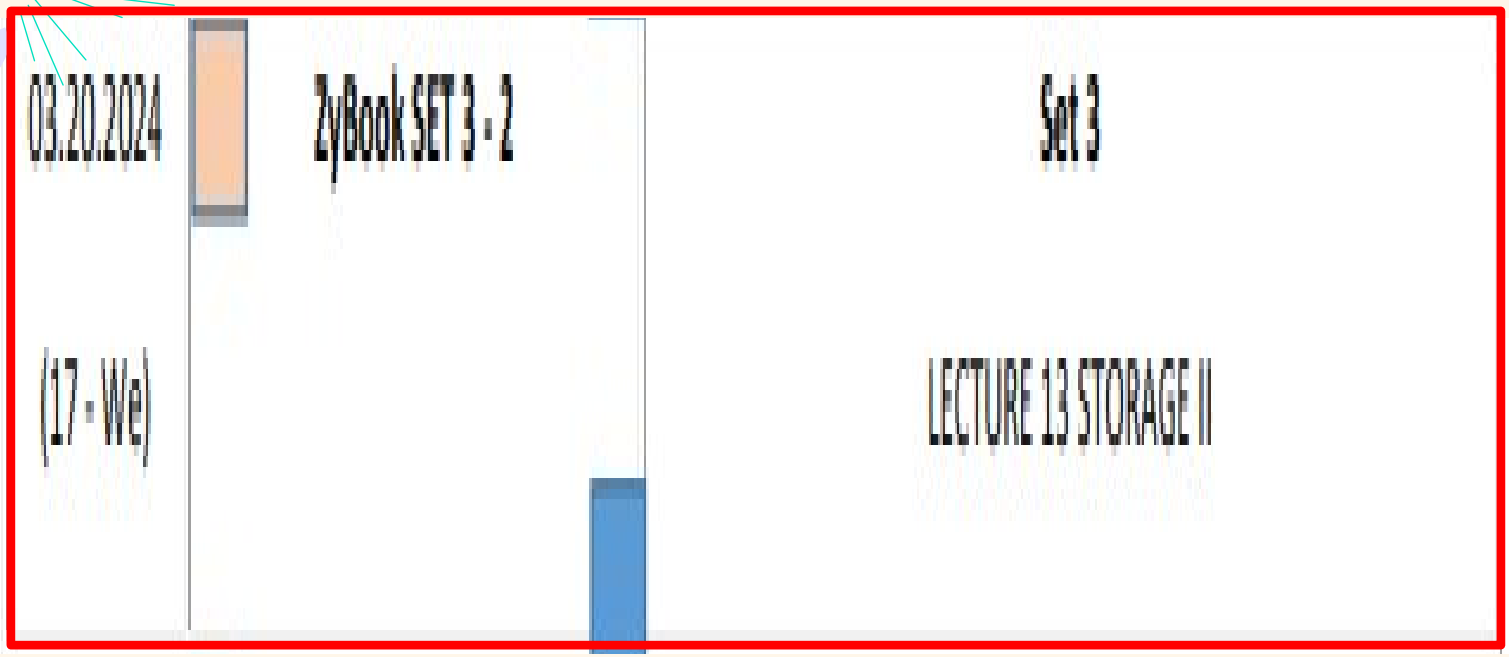
Please close all other windows.

03.20.2024 (17 - We)	**ZyBook SET 3 - 2**	**Set 3** LECTURE 13 STORAGE II
03.25.2024 (18 – Mo)	**ZyBook SET 3- 3**	**Set 3** LECTURE 14 STORAGE III B TREE INDEX
03.27.2024 (19 - We)	**ZyBook SET 3- 4**	**Set 3** LECTURE 15 STORAGE IV HASH INDEX
04.01.2024 (20 - Mo)		**EXAM 3 Practice** **(PART of 20 points)**
04.03.2024 (21 – We)	**TA Download** **ZyBook SET 3 Sections** **(4 PM)** **(PART of 30 points)**	**EXAM 3 Review** **(PART of 20 points)**
04.08.2024 (22 - Mo)		**EXAM 3** **(PART of 50 points)**

COSC 3380

Class 17





From 4:00 to 4:07 PM – 5 minutes.

03.20.2024		ZyBook SET 3 - 2	Set 3
(17 - We)			LECTURE 13 STORAGE II

CLASS PARTICIPATION 20 points — 20% of Total

STORAGE II

Class 17 BEGIN PARTICIPATION

Not available until Mar 20 at 4:00pm | Due Mar 20 at 4:07pm | 100 pts

VH, publish

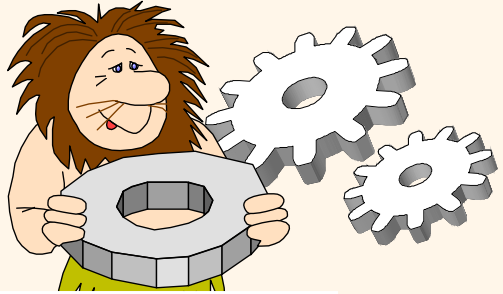
This is a synchronous online class.

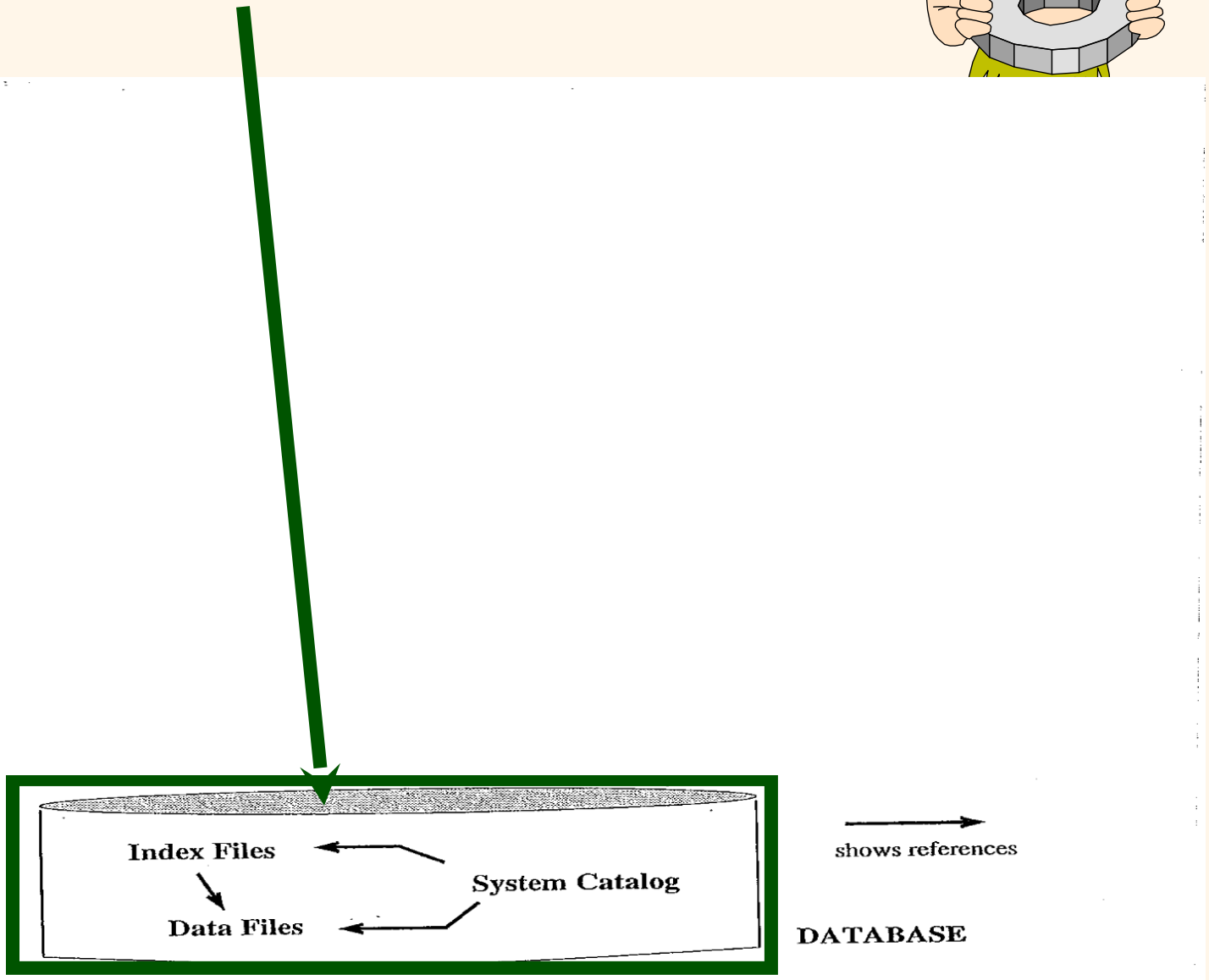
Attendance is required.

Recording or distribution of class materials is prohibited.

1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**

A DBMS





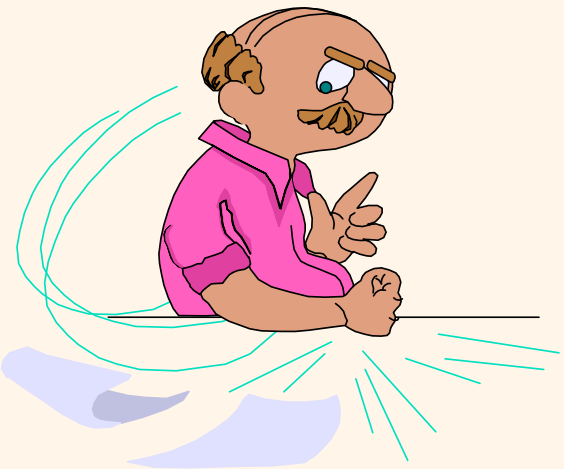
Tables
Indexes

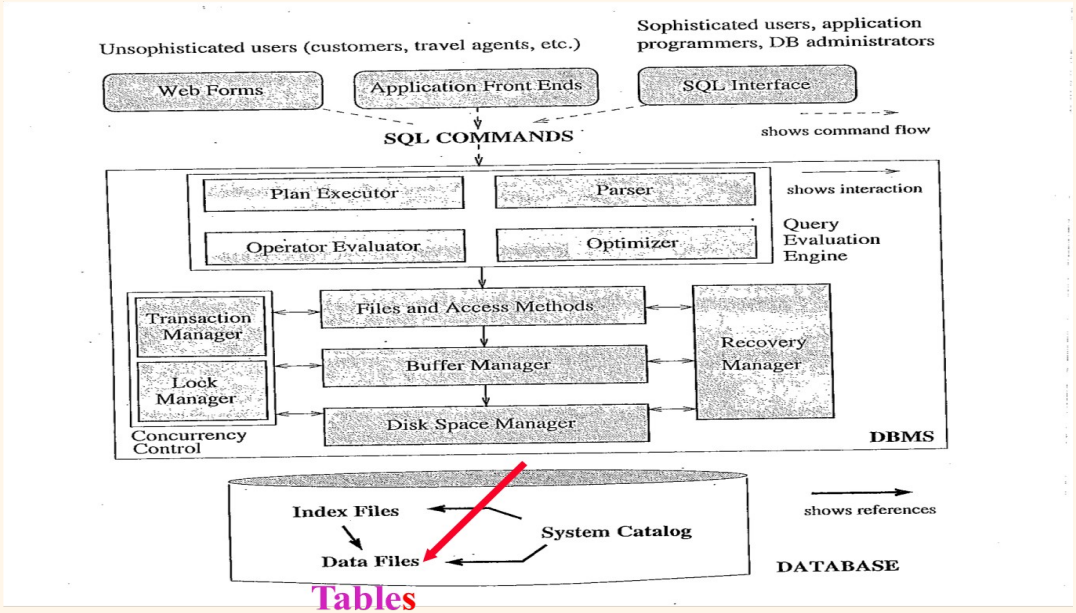
From 4:07 to 5:00 PM – 53 minutes.

COSC 3380

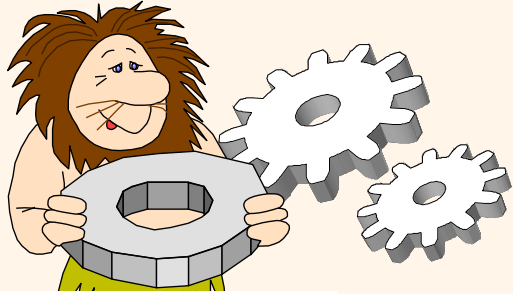
Lecture 13

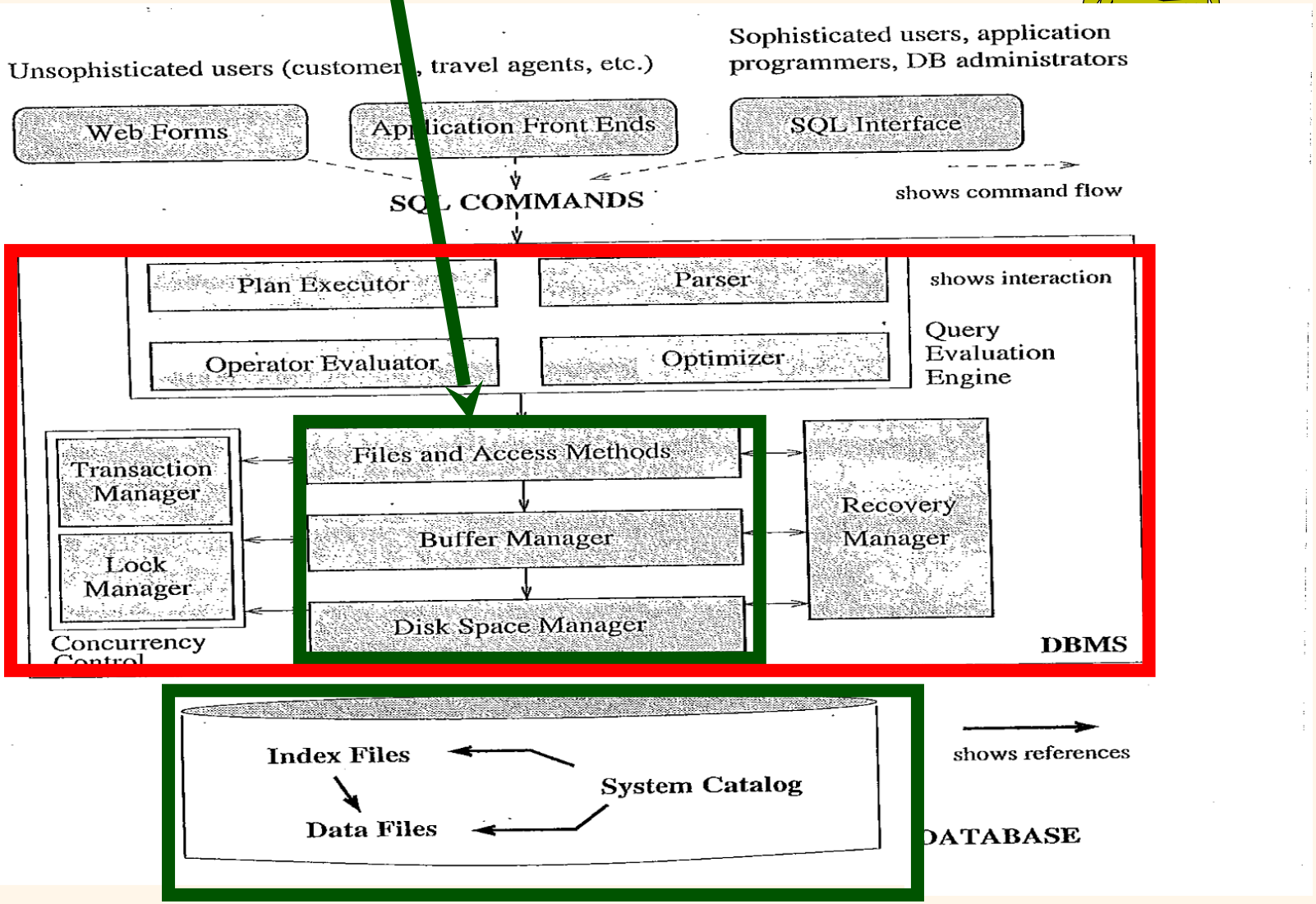
*Storing **Data**: **Disks** and **Files***



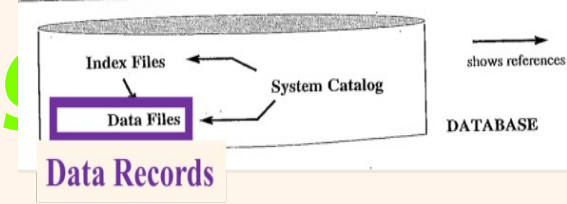


A DBMS




Unsophisticated users (customers, travel agents, etc.)
Sophisticated users, application programmers, DB administrators
Web Forms
Application Front Ends
SQL Interface
SQL COMMANDS
shows command flow
Plan Executor
Parser
shows interaction
Operator Evaluator
Optimizer
Query Evaluation Engine
Transaction Manager
Lock Manager
Concurrency Control
Files and Access Methods
Buffer Manager
Disk Space Manager
Recovery Manager
DBMS
Index Files
System Catalog
Data Files
shows references
DATABASE

Data on External Storag



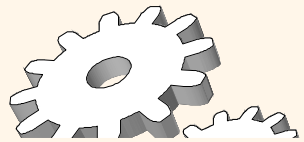
Architecture:

- Files and Access Methods processes **Pages** obtained by asking the *Buffer Manager* to fetch the **Page *rid***.

- Buffer Manager reads **Pages** from **Disk** into Memory for processing & writes **Pages** from Memory to **Disk** for persistent storage.

- Disk Space Manager allocates/deallocates **Disk Pages** when the *Files and Access Methods* needs additional space to hold new **Data Records** in a **File**/no longer needs one of its **Disk Pages**.

Disk Space Management



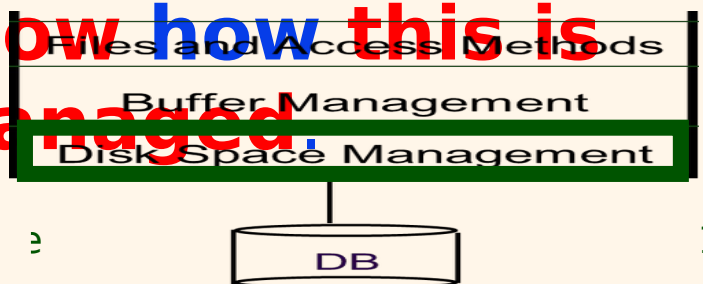
Lowest layer of **DBMS** software manages space on Disk.

Higher levels call upon this layer to:

- allocate/de-allocate a **Page**
- read/write a **Page**

Request for a ***sequence* of Pages** must be satisfied by allocating the **Pages** sequentially on **Disk**!

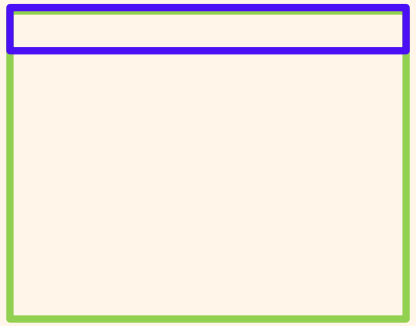
Higher levels **don't need to know how this is done**, or **how free space is managed**.



Data Record (Tuple/Row) Formats?

Students(sid:string,name:string,login:string,age:integer,gpa:real)

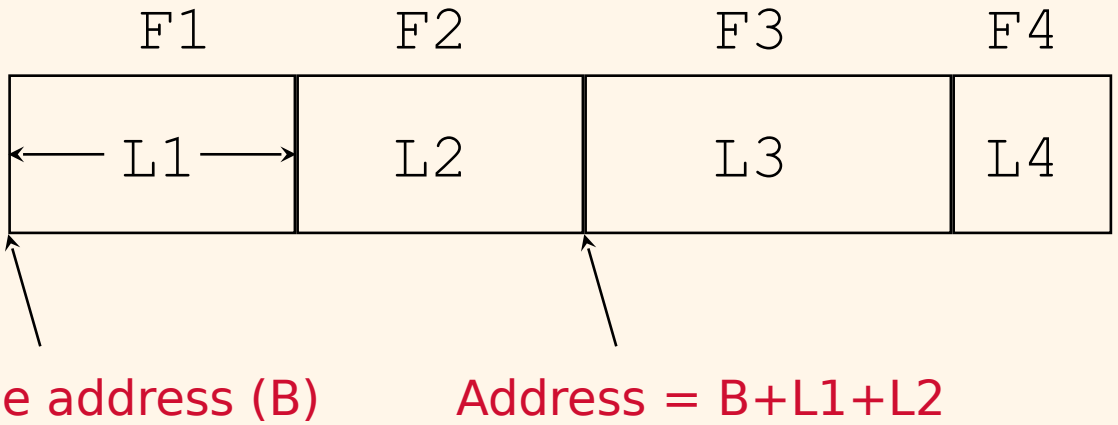
```
CREATE TABLE Students
        (sid: CHAR(20),
         name: CHAR(20),
         login: CHAR(10),
         age: INTEGER,
         gpa: REAL)
```

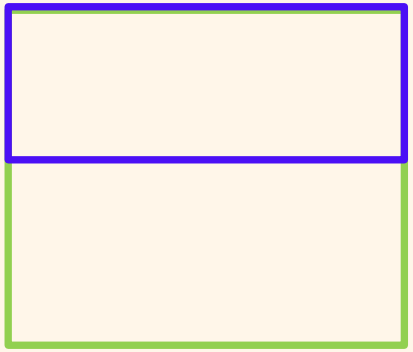


F1 F2 F3 F4 F5

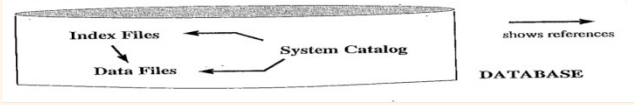
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Record (Tuple) Formats: Fixed Length





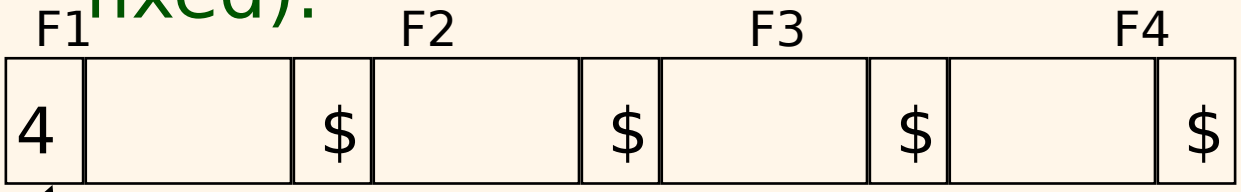
- Information about field types same for **all Data Records** in a **File**; stored in *system* ***Catalogs***.



- Finding ***i'th* field** requires **scan** of **Data Record**.

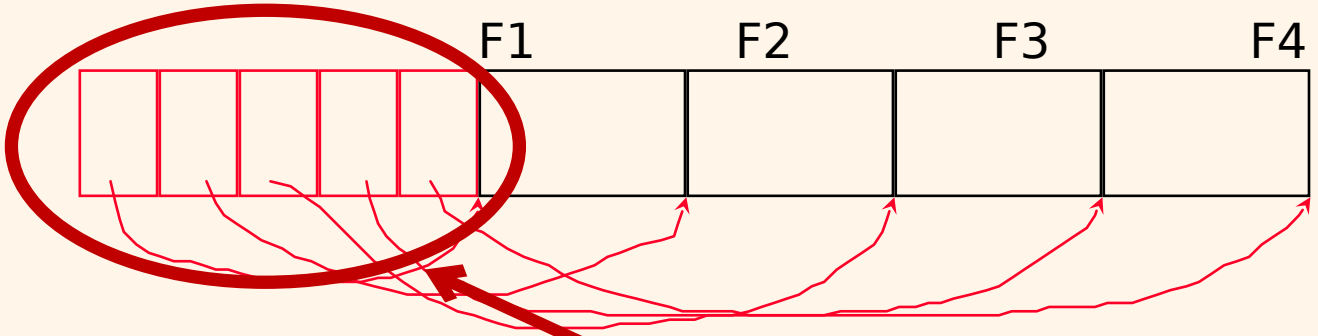
Record (Tuple) Formats: Variable Length

- Two alternative formats (**# fields** is fixed):

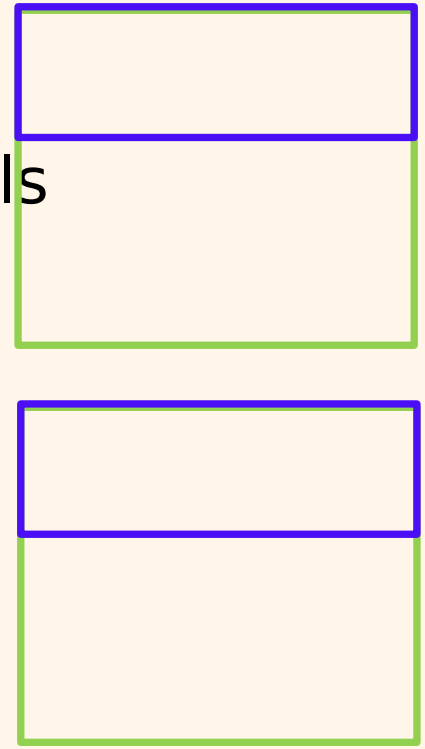


Fields Delimited by Special Symbols

Field Count



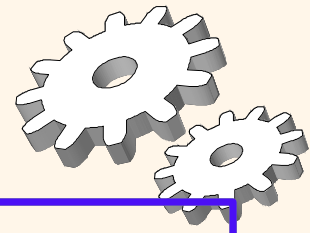
Array of **Field** Offsets



☛ Second offers direct access to **i'th field**, efficient storage

of *nulls* (special *don't know* value); small **Directory**

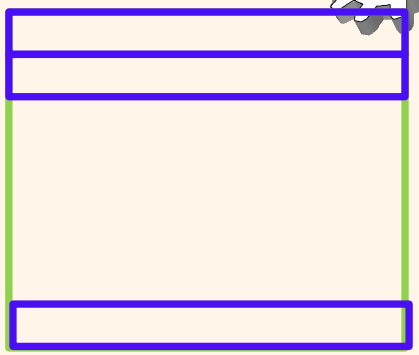
Transcript File Stored as a Heap File



StudId	CrsCode	Sem	Grade
666666	MGT123	F1994	4.0
123456	CS305	S1996	4.0
987654	CS305	F1995	2.0

Page 0

B = 3
R = 4



717171	CS315	S1997	4.0
666666	EE101	S1998	3.0
765432	MAT123	S1996	2.0
515151	EE101	F1995	3.0

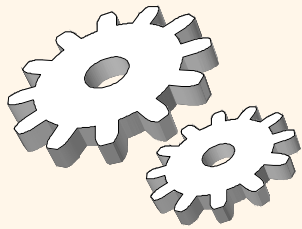
Page 1

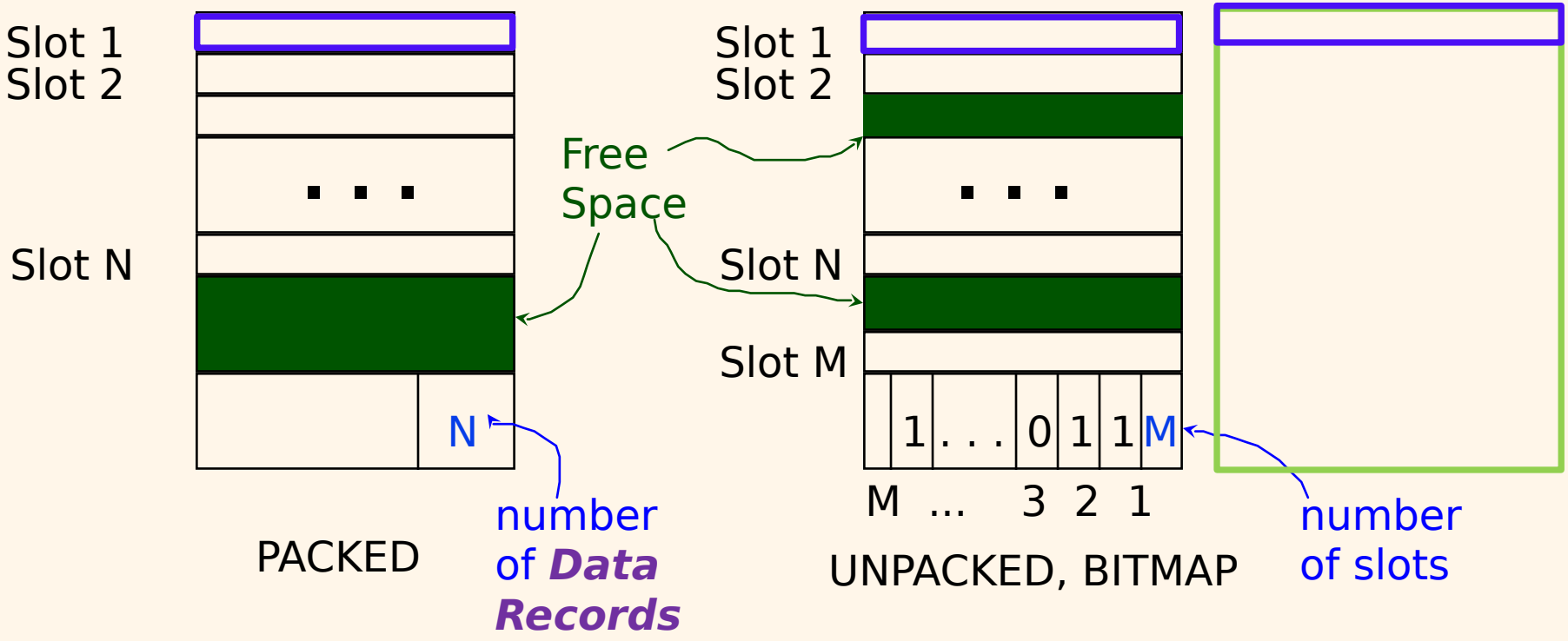
234567	CS305	S1999	4.0
878787	MGT123	S1996	3.0

Page 2

Page Formats?

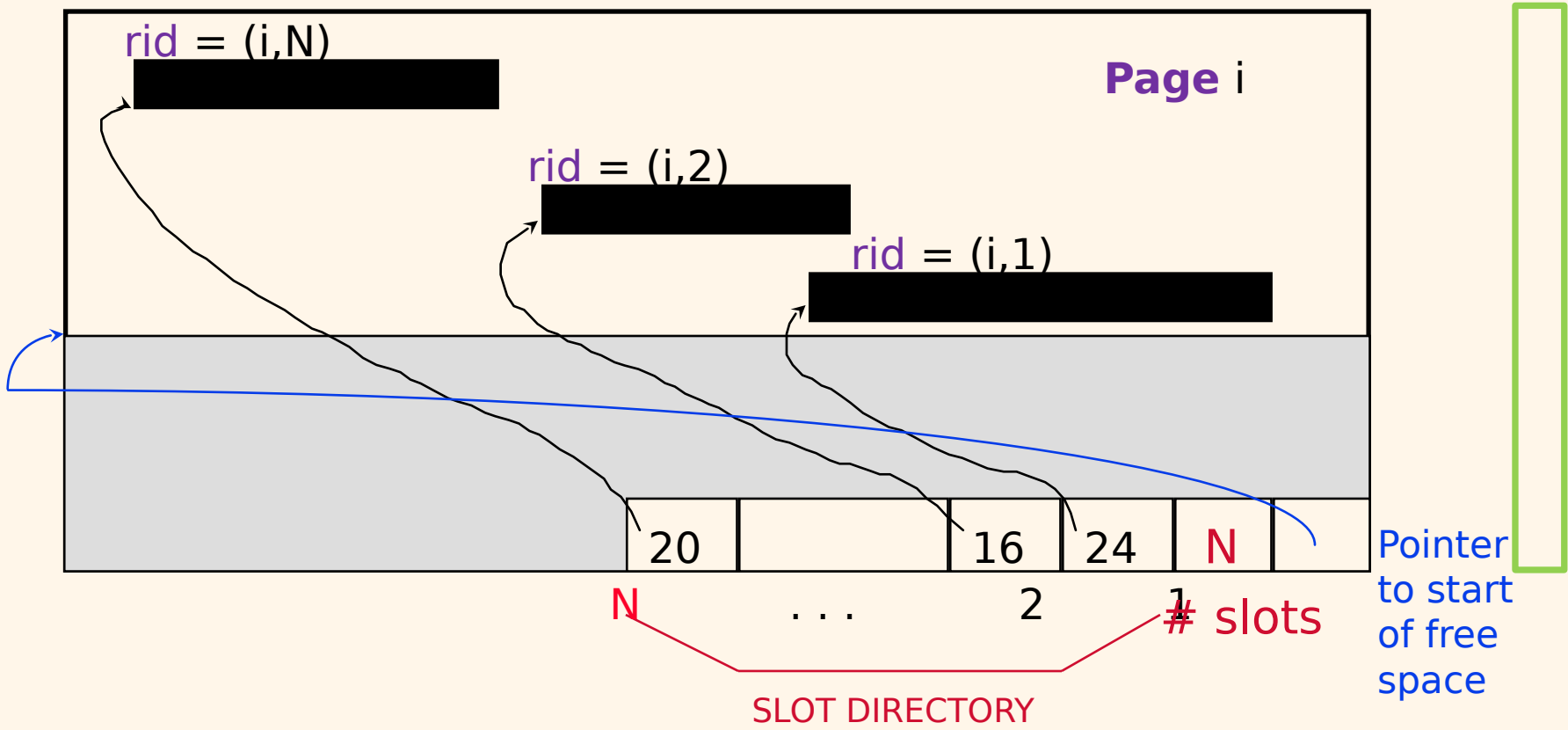
Page Formats: Fixed Length Records





- *Record id (**rid**) = <**Page id**, slot #>. In first alternative, moving **Data Records** for free space management changes **rid**; may not be ke

Page Formats: Variable Length Records



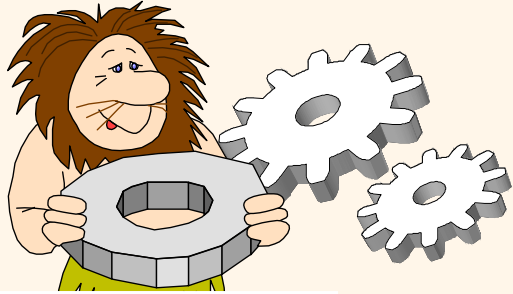
- *Can move **Data Records** on **Page** without changing **rid**; so, attractive for **fixed-length***

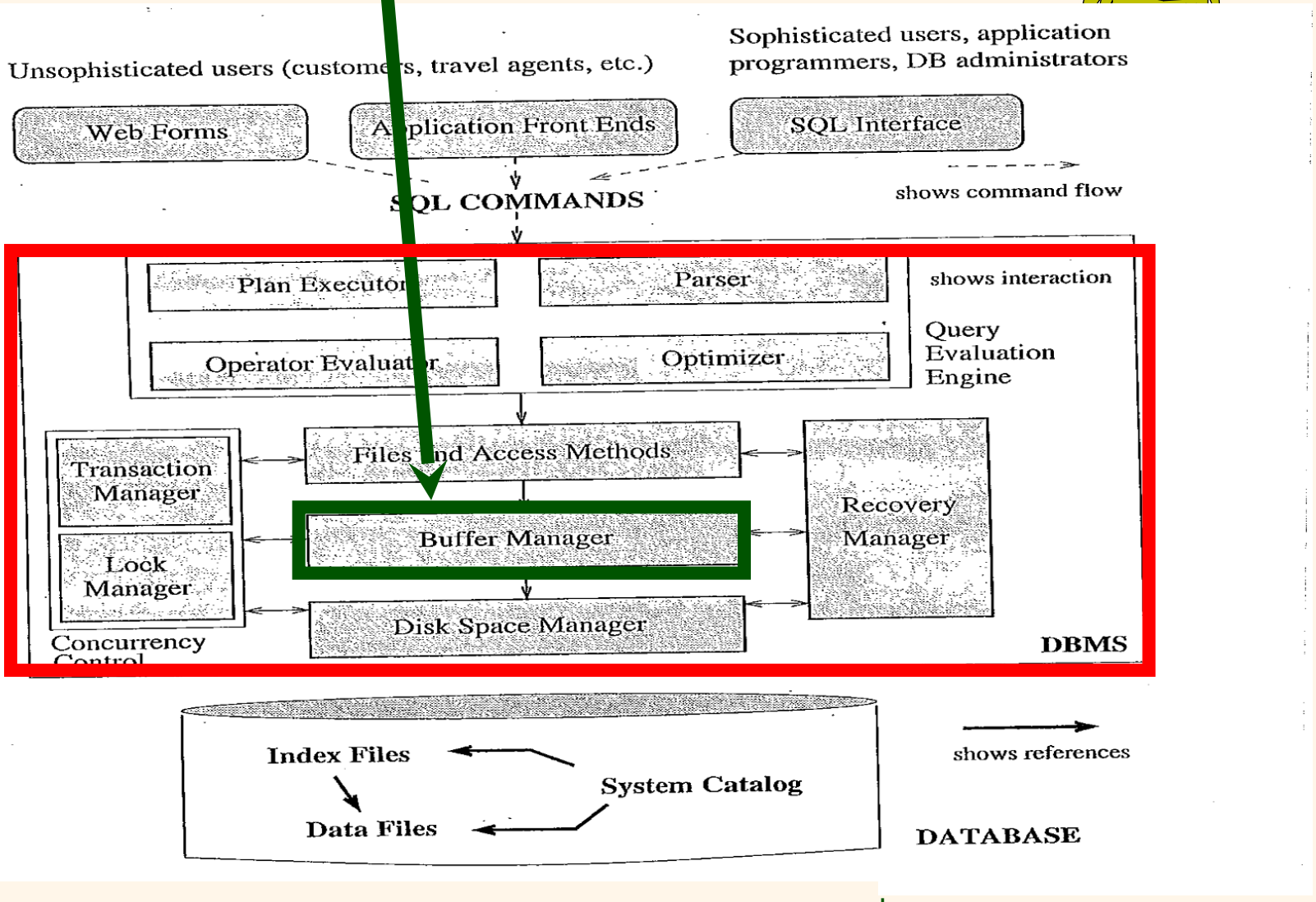
TA, Jordan (A – L).

TA, Fernando (M – Z).

Please compare CANVAS vs. TEAMS Attendance.
Print screens of students in CANVAS but not in the TEAMS meeting.
(3.20.2024 Attendance X missing LastName.docx)

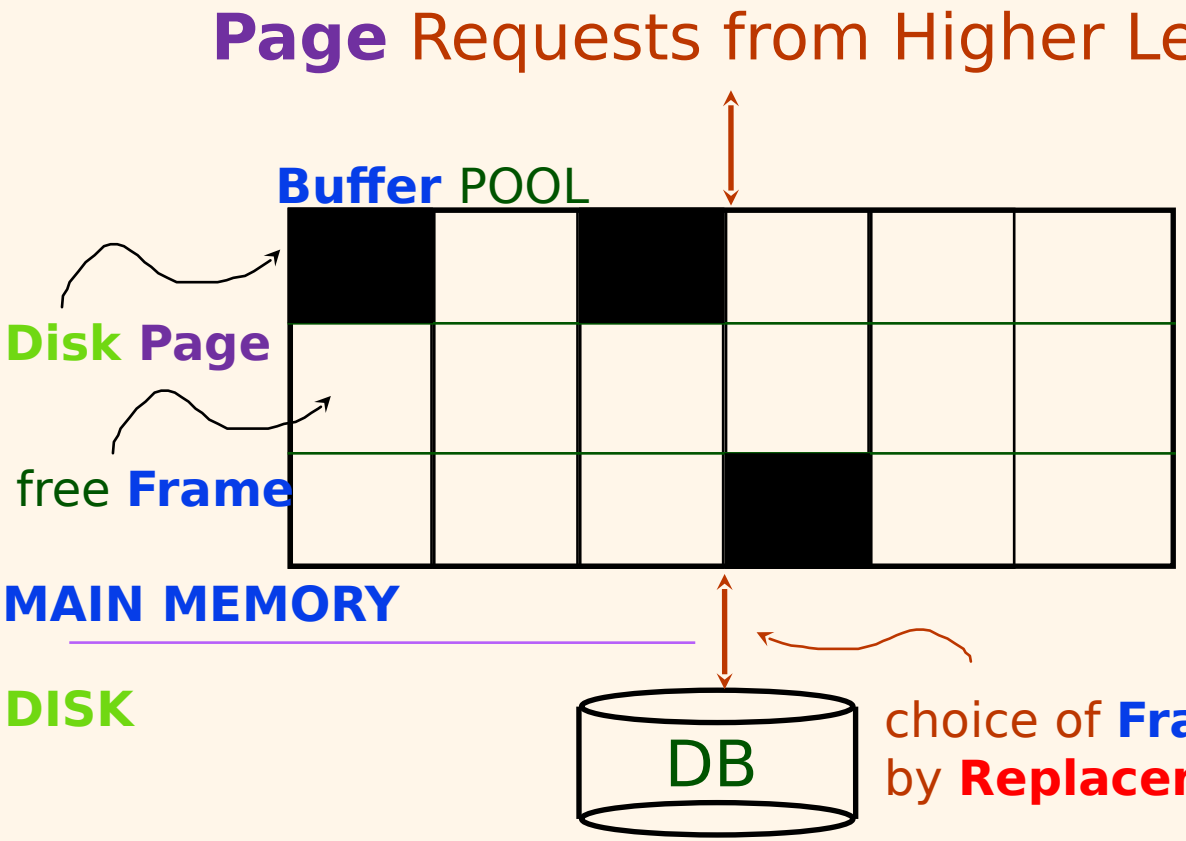
A DBMS

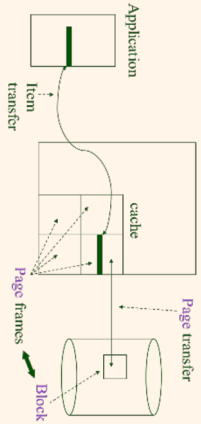



Unsophisticated users (customers, travel agents, etc.)
Sophisticated users, application programmers, DB administrators
Web Forms
Application Front Ends
SQL Interface
SQL COMMANDS
shows command flow
Plan Executor
Parser
Operator Evaluator
Optimizer
shows interaction
Query Evaluation Engine
Transaction Manager
Lock Manager
Files and Access Methods
Buffer Manager
Disk Space Manager
Recovery Manager
Concurrency Control
DBMS
Index Files
System Catalog
Data Files
shows references
DATABASE

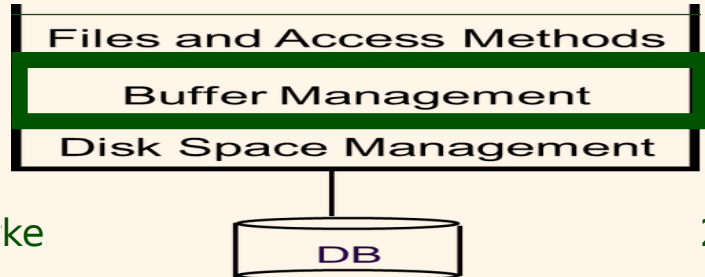
ke

Buffer (Frame) Management in a DBMS

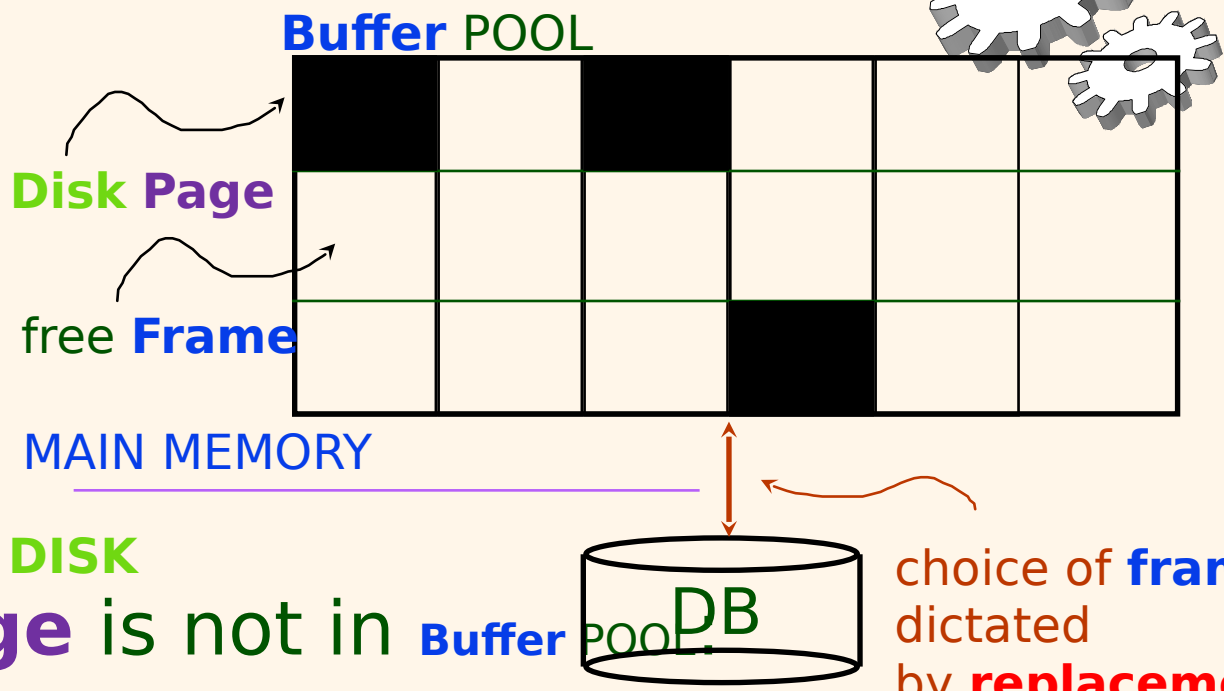




- *Data must be in RAM for DBMS to operate on it!*
- *Table of <frame#, Pageid> pairs is*



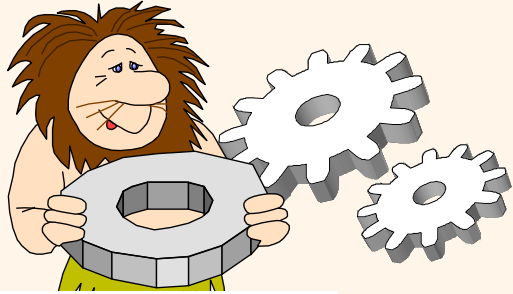
*When a **Page** is Requested ...*

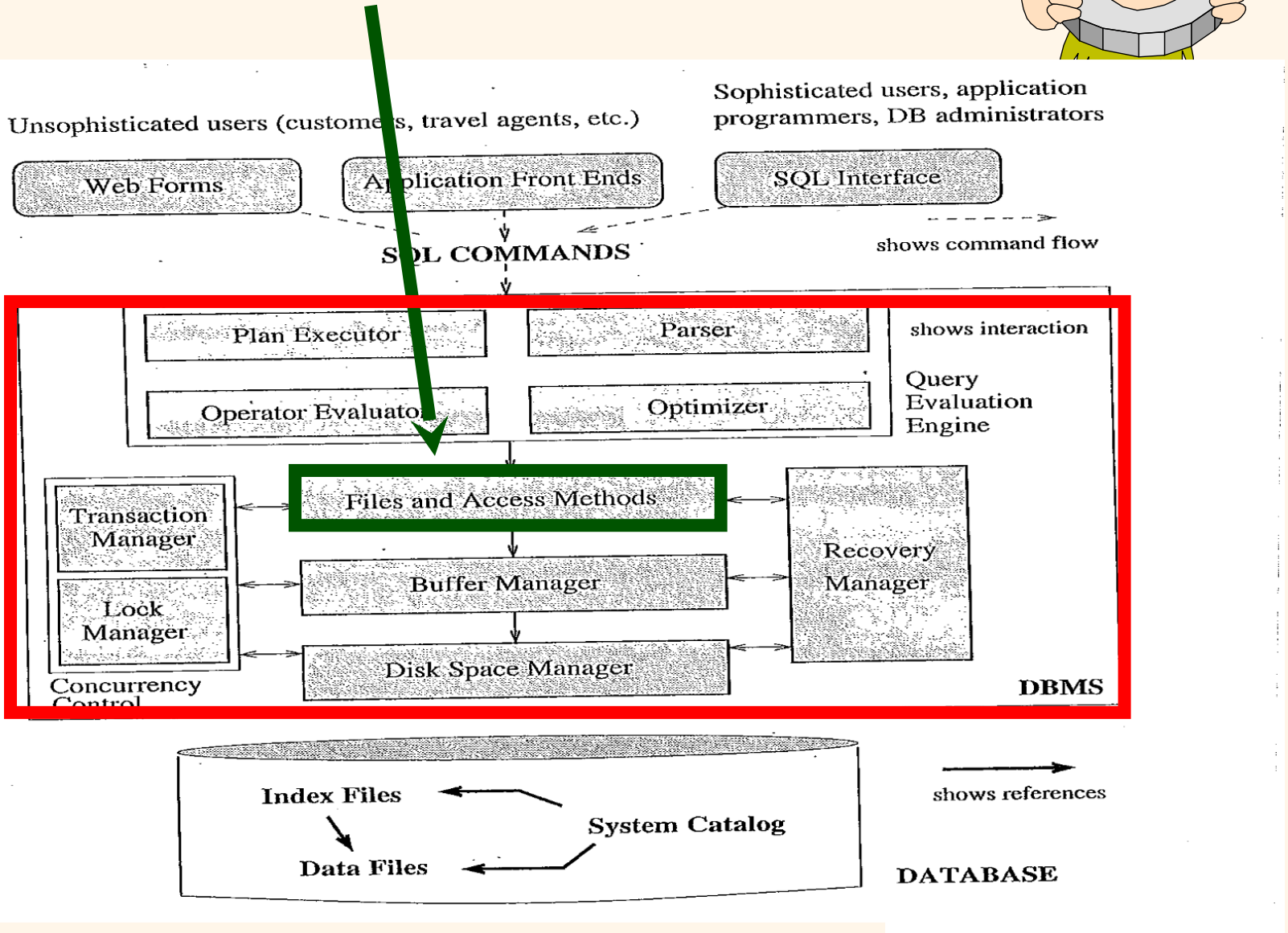


- If requested **Page** is not in **Buffer** POOL
  - Choose a **Frame** for *replacement*
  - If **Frame** is dirty, Write it to **Disk**
  - Read requested **Page** into chosen **Frame**
- *Pin* the **Page** and return its address.

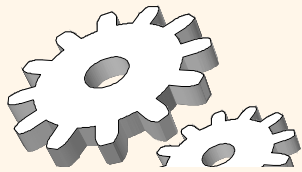
☛ *If requests can be predicted (e.g., **sequential scans**) **Pages** can be pre-fetched several **Pages** at a time!*

A DBMS




Unsophisticated users (customers, travel agents, etc.)
Sophisticated users, application programmers, DB administrators
Web Forms
Application Front Ends
SQL Interface
SQL COMMANDS
shows command flow
Plan Executor
Parser
shows interaction
Operator Evaluator
Optimizer
Query Evaluation Engine
Transaction Manager
Files and Access Methods
Recovery Manager
Lock Manager
Buffer Manager
Disk Space Manager
Concurrency Control
DBMS
Index Files
System Catalog
Data Files
shows references
DATABASE

Files of Data Records (Tuples/Rows)



Page or Block is OK when doing **I/O**, but higher levels of **DBMS** operate on ***Data Records*** (**Tuples**), and ***Files of Data Records*** (**Tuples**).

FILE: A collection of **Pages** (**B**), each containing a **collection** of **Data Records** (**Tuples**) (**R**). Must support:

- insert/delete/modify **Data Record**
- read a particular **Data Record** (specified using *record id* ***rid***)
- scan all **Data Records** (possibly with **some** to be retrieved)

sid	*name*	*login*	*age*	*gpa*	
53831	Madayan	madayan@music	11	1.8	<1,1>
53832	Guldu	guldu@music	12	2.0	<1,2>
53666	Jones	jones@cs	18	3.4	<1,3>
53688	Smith	smith@ee	19	3.2	<2,1>
53650	Smith	smith@math	19	3.8	<2,2>

Unordered (Heap) Files of Data Records

Simplest **File** structure contains **Data Records** in no particular order.

As **File** grows and shrinks, Disk **Pages** are allocated and de allocated.

To support **Data Record** level operations, we must:

- keep track of **the *Pages*** in a **File** (**B**)
- keep track of *free space* on **Pages**
- keep track of the ***Data Records*** on a **Page** (**R**)

Th

sid	name	login	age	gpa
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	19	3.2
53650	Smith	smith@math	19	3.8

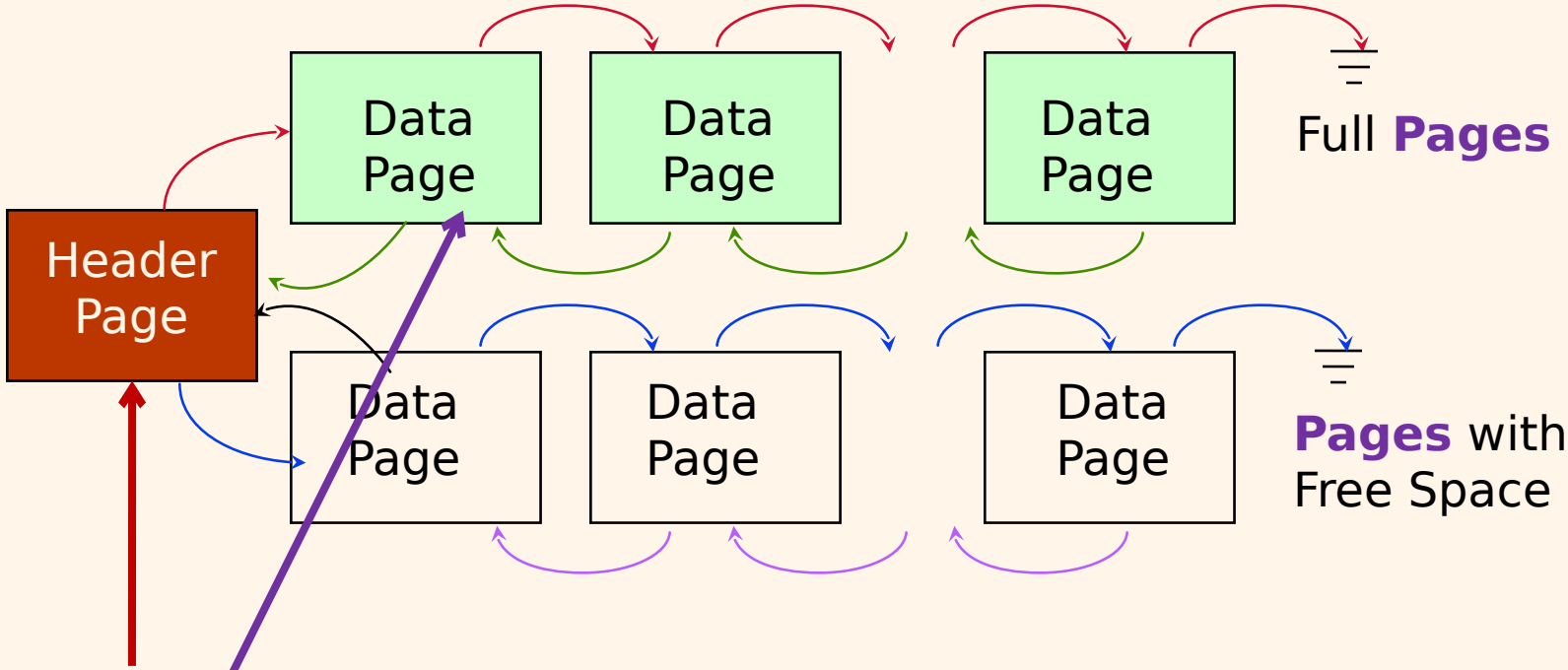
<1,1>
<1,2>
<1,3>
<2,1>
<2,2>

or keeping track of **Pages** (**B**).

ta

Heap File *Implemented* ***as a Linked*** *based* ***LIST***

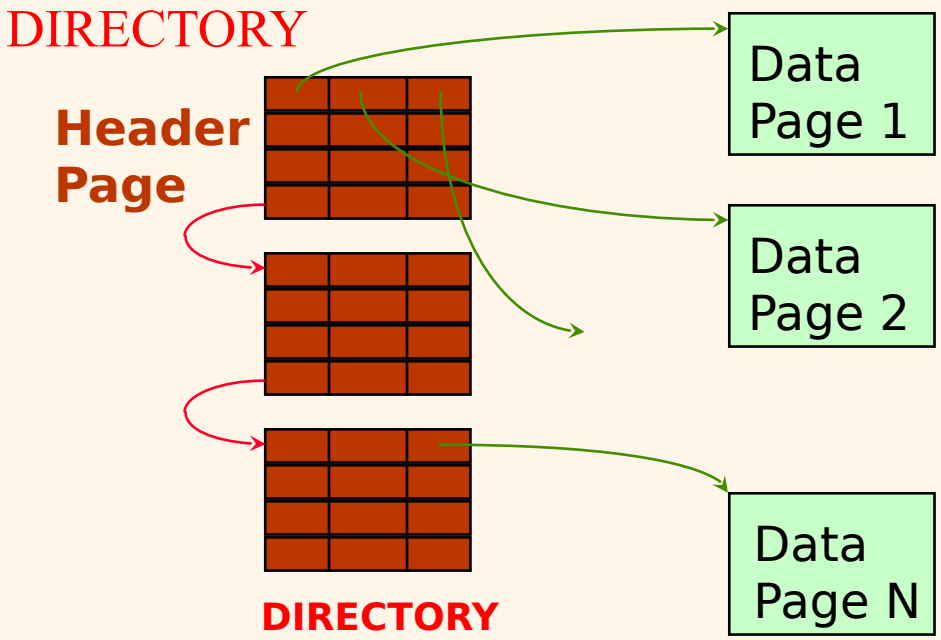
(I loved my Data Structures Class!)



The **Header Page id** and **Heap File** name must be stored someplace.

Each **Data Page** contains 2 `pointers' plus **Data Records**

Heap File Using a Page DIRECTORY

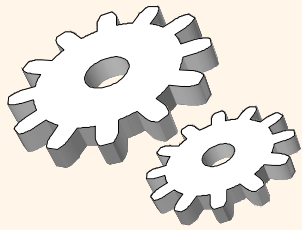


The **Header** entry for a **Data Page** can include the **number of free bytes** on the **Page**.

DIRECTORY is a collection of **Pages**; Linked **LIST** implementation is just one alternative.

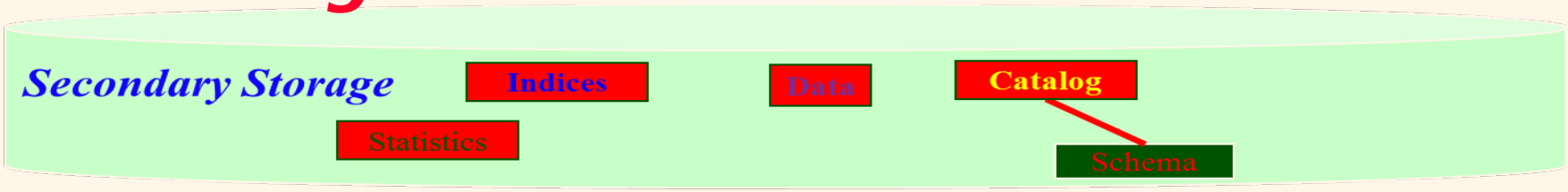
- *Much smaller than **Linked based List** of all **Heap File***

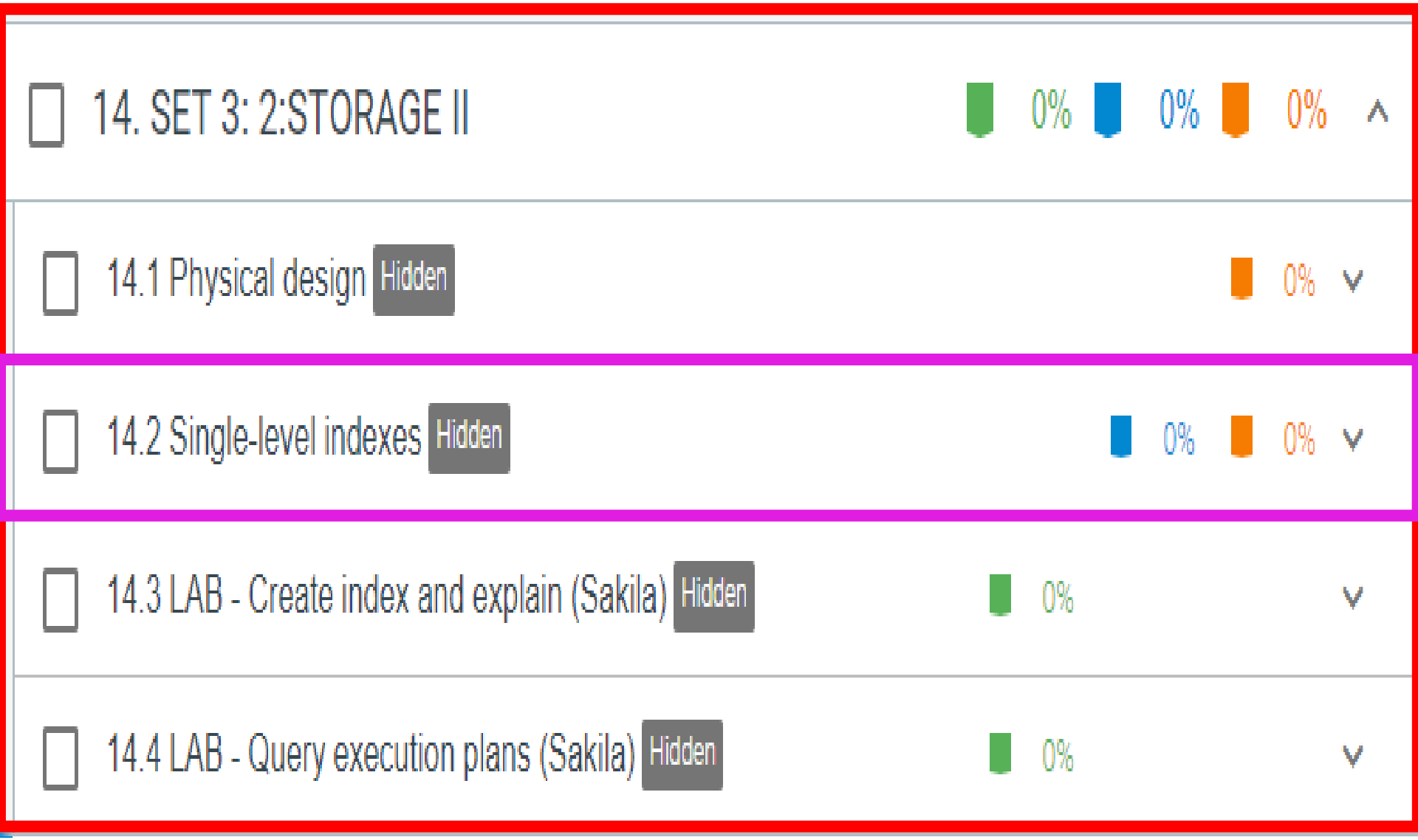
System **Catalogs**



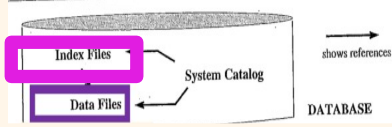
- For each Index:
  - structure (e.g., **B+ tree**) and **search key** fields
- For each **Relation/Table**:
  - **name**, **File** name, **File** structure (e.g., Heap **File**)
  - attribute name and type, for each attribute
  - index name, for each Index
  - integrity constraints
- For each **View**:
  - view name and definition
- Plus Statistics, Authorization, **Buffer** Pool Size, etc.

☛ ***Catalogs*** *are themselves stored as* ***Relations***!




14. SET 3: 2:STORAGE II
0%
0%
0%
14.1 Physical design
Hidden
0%
14.2 Single-level indexes
Hidden
0%
0%
14.3 LAB - Create index and explain (Sakila)
Hidden
0%
14.4 LAB - Query execution plans (Sakila)
Hidden
0%

Indexes – Data Structures



key,

r_{id}

- **Ordered** set of values that contains **Index** search key and *pointers*
- **More efficient** to use **Index** to access **Table** than to **scan all Rows** in **Table** sequentially

key, r_{id}

STATE_NDX INDEX

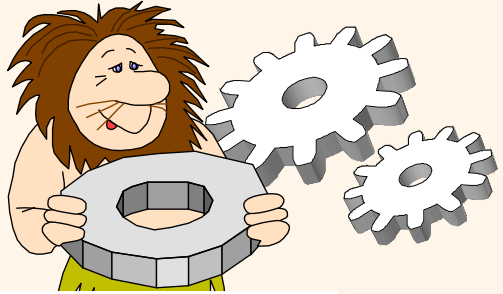
Key	Row
AZ	2
....	....
....	....
FL	1
FL	7
FL	8
FL	13245
FL	14786
....	
....	

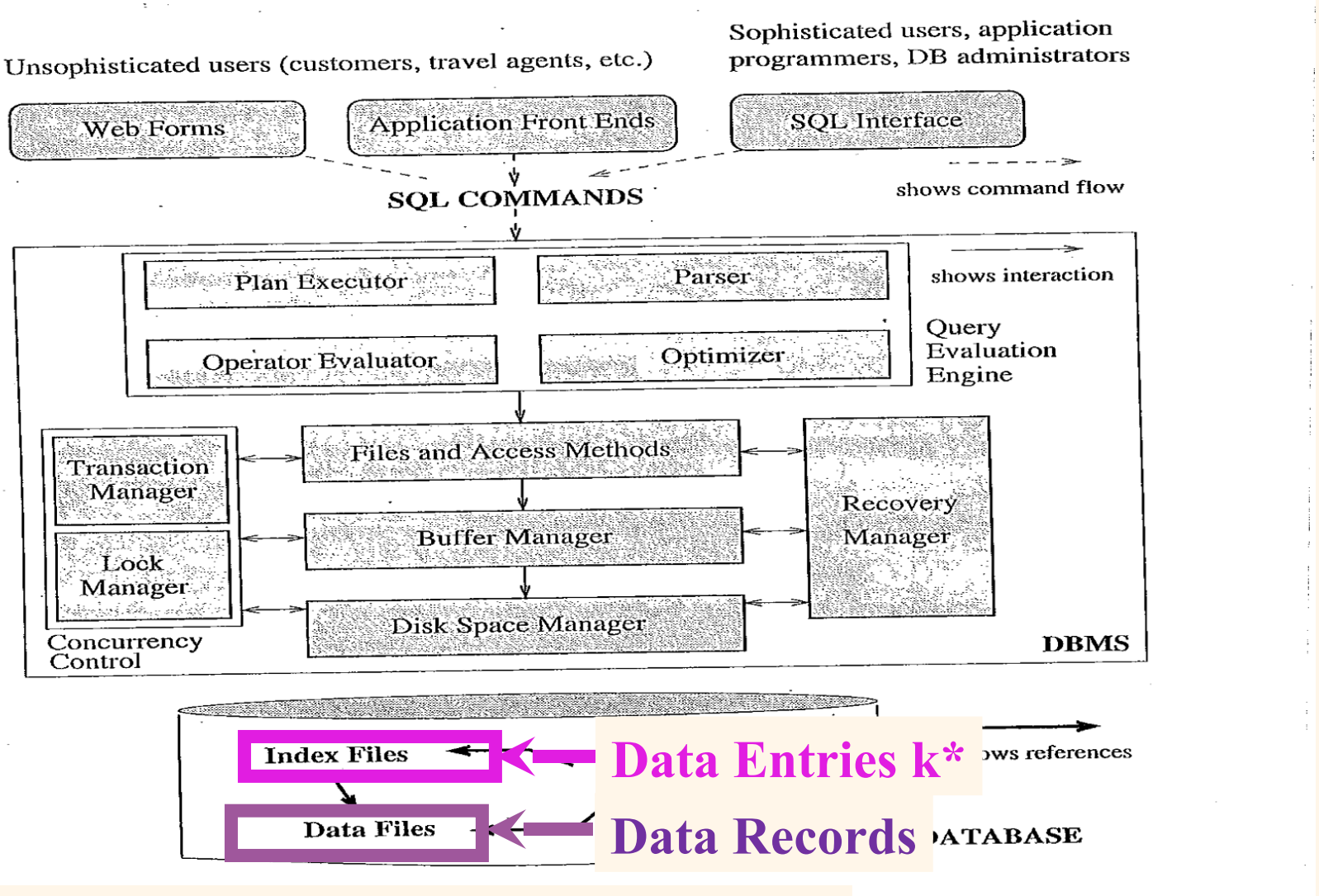
CUSTOMER TABLE
(14,786 rows)

Row #	CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_STATE	CUS_BALANCE
1	10010	Ramas	Alfred	A	615	844-2573	FL	$0.00
2	10011	Dunne	Leona	K	713	894-1238	AZ	$0.00
3	10012	Smith	Kathy	W	615	894-2285	TX	$345.86
4	10013	Olowski	Paul	F				
5	10014	Orlando	Myron					
6	10015	O'Brian	Amy	B				
7	10016	Brown	James	C				
8	10017	Williams	George					
9	10018	Farriss	Anne	C				
10	10019	Smith	Olette	K				
....	.....	.....	.....	.				
....	.....	.....	.....	.				
13245	23120	Veron	George	D				
....	.....	.....	.....	.				
....	.....	.....	.....	.				
14786	24560	Suarez	Victor					

- Indexes: Data structures to organize records via trees or hashing.
  - Like Sorted Files, they speed up searches for a subset of records, based on values in certain ("search key") fields

A DBMS




Unsophisticated users (customers, travel agents, etc.)
Sophisticated users, application programmers, DB administrators
Web Forms
Application Front Ends
SQL Interface
SQL COMMANDS
shows command flow
Plan Executor
Parser
Operator Evaluator
Optimizer
shows interaction
Query Evaluation Engine
Transaction Manager
Lock Manager
Concurrency Control
Files and Access Methods
Buffer Manager
Disk Space Manager
Recovery Manager
DBMS
Index Files
Data Files
Data Entries k*
Data Records
ows references
ATABASE

Dense Index

\# index entries = # of data records

14. SET 3: 2:STORAGE II 0% 0% 0%

14.1 Physical design Hidden 0%

14.2 Single-level indexes Hidden 0% 0%

14.3 LAB - Create index and explain (Sakila) Hidden 0%

14.4 LAB - Query execution plans (Sakila) Hidden 0%

14.1 Physical design

MySQL storage engines

Logical design specifies tables, columns, and keys. The logical design process is described elsewhere in this material. ***Physical design*** specifies indexes, table structures, and partitions. Physical design affects query performance but never affects query results.

A ***storage engine*** or ***storage manager*** translates instructions generated by a query processor into low-level commands that access data on storage media. Storage engines support different index and table structures, so physical design is dependent on a specific storage engine.

MySQL can be configured with several different storage engines, including:

- InnoDB is the default storage engine installed with the MySQL download. InnoDB has full support for transaction management, foreign keys, referential integrity, and locking.
- MyISAM has limited transaction management and locking capabilities. MyISAM is commonly used for analytic applications with limited data updates.
- MEMORY stores all data in main memory. MEMORY is used for fast access with databases small enough to fit in main memory.

Different databases and storage engines support different table structures and index types. Ex:

- Table structure. Oracle Database supports heap, sorted, hash, and cluster tables. MySQL with InnoDB supports only heap and sorted tables.
- Index type. MySQL with InnoDB or MyISAM supports only B+tree indexes. MySQL with MEMORY supports both B+tree and hash indexes.

This section describes the physical design process and statements for MySQL with InnoDB. The process and statements can be adapted to other databases and storage engines, but details depend on supported index and table structures.

PARTICIPATION ACTIVITY | 14.1.1: Logical and physical design.

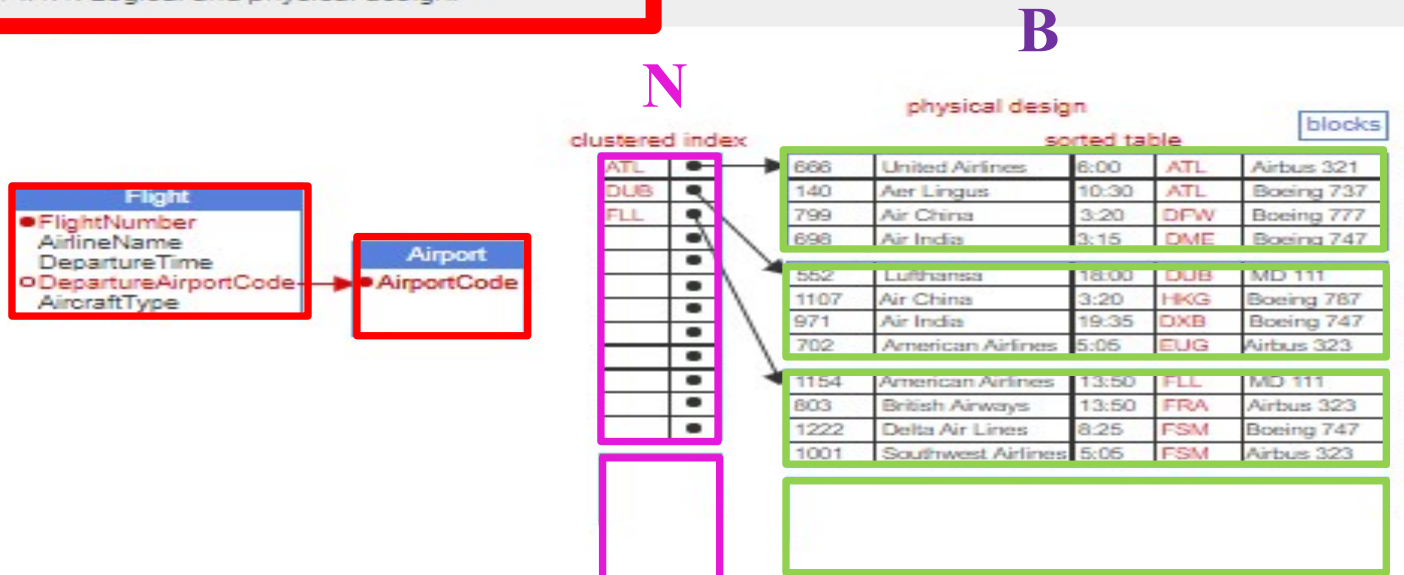
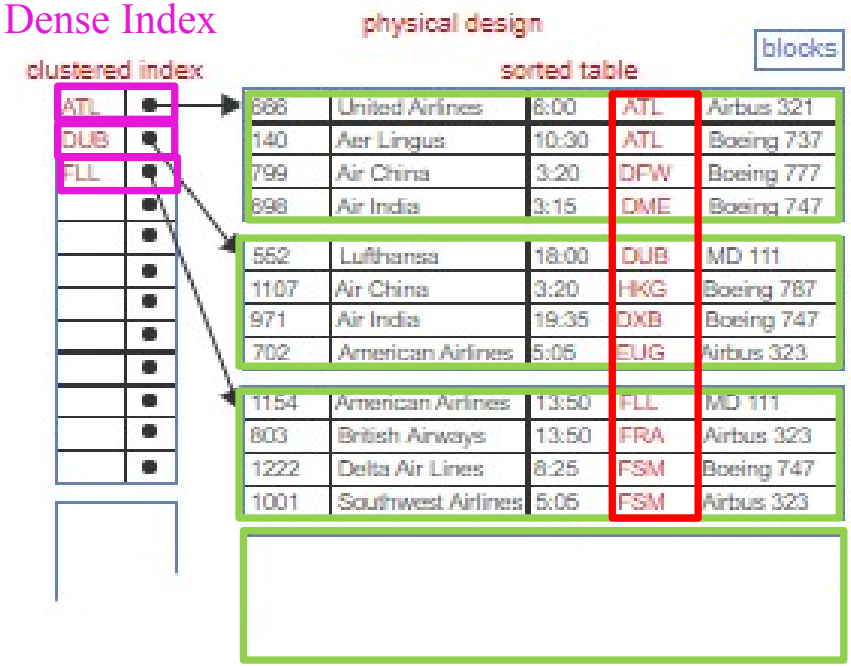


Table Sorted on search key

index entries = # of blocks

Dense Index



B

N

CREATE INDEX, DROP INDEX, and SHOW INDEX statements

Next two classes

The ***CREATE INDEX*** statement creates an index by specifying the index name and table columns that compose the index. Most indexes specify just one column, but a composite index specifies multiple columns.

The ***DROP INDEX*** statement deletes a table's index.

The ***SHOW INDEX*** statement displays a table's index. SHOW INDEX generates a result table with one row for each column of each index. A multi-column index has multiple rows in the result table.

The SQL standard includes logical design statements such as CREATE TABLE but not physical design statements such as CREATE INDEX. Nevertheless, CREATE INDEX and many other physical design statements are similar in most relational databases.

Table 14.1.1: INDEX statements.

Statement	Description	Syntax
CREATE INDEX	Create an index	`CREATE INDEX IndexName ON TableName (Column1, Column2, ..., ColumnN);`
DROP INDEX	Delete an index	`DROP INDEX IndexName ON TableName;`
SHOW INDEX	Show an index	`SHOW INDEX FROM TableName;`

Feedback?

14. SET 3: 2:STORAGE II

0% 0% 0%

14.1 Physical design Hidden

0%

14.2 Single-level indexes Hidden

0% 0%

Participation activities

14.2.1: Single-level index.	0%
14.2.2: Single-level indexes.	0%
14.2.3: Query processing with single-level index.	0%
14.2.4: Query processing with single-level index...	0%
14.2.5: Binary search on a sorted index.	0%
14.2.6: Binary search.	0%
14.2.7: Dense and sparse indexes.	0%
14.2.8: Primary, clustering, and secondary index...	0%
14.2.9: Insert with dense sorted index.	0%
14.2.10: Inserts, updates, and deletes.	0%

Challenge activities

14.2.1: Single-level indexes.	0%

14.3 LAB - Create index and explain (Sakila) Hidden

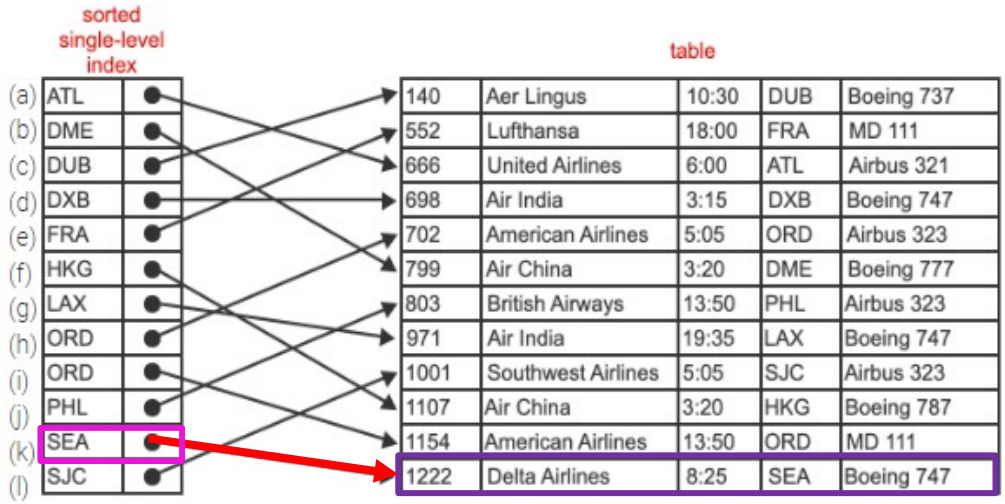
0%

14.4 LAB - Query execution plans (Sakila) Hidden

0%

VH

CHALLENGE ACTIVITY | 14.2.1: Single-level indexes.



Select the index entry that refers to the row for Delta Airlines flight 1222.

Pick k

1

1 2 3 4 5 6

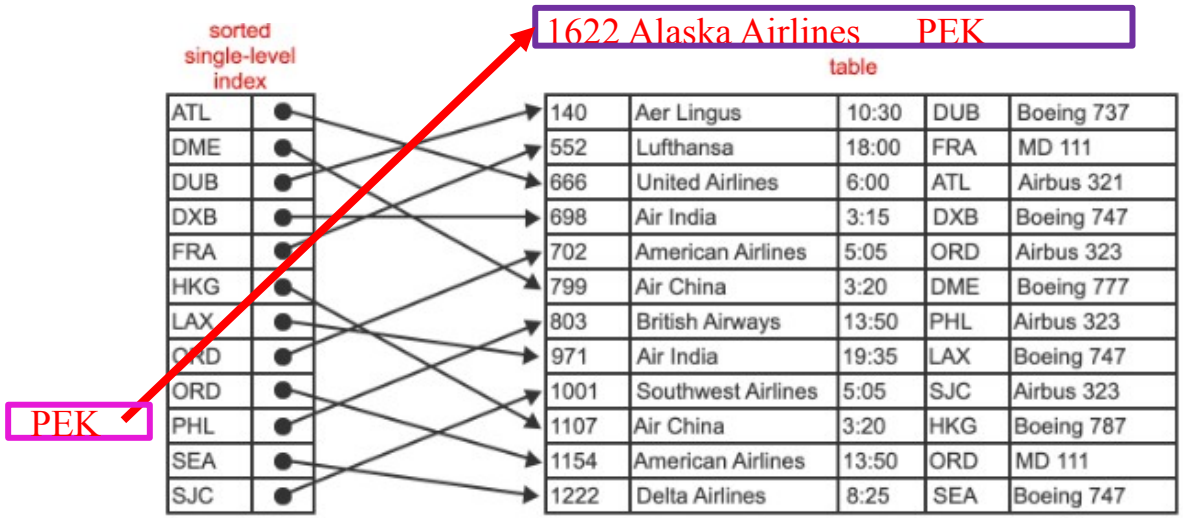
ta

?????

VH

CHALLENGE ACTIVITY | 14.2.1: Single-level indexes.

Index



Alaska Airlines flight 1622 departing from PEK is inserted into the table. Where does the new index entry go?

Pick Between ORD and PHL

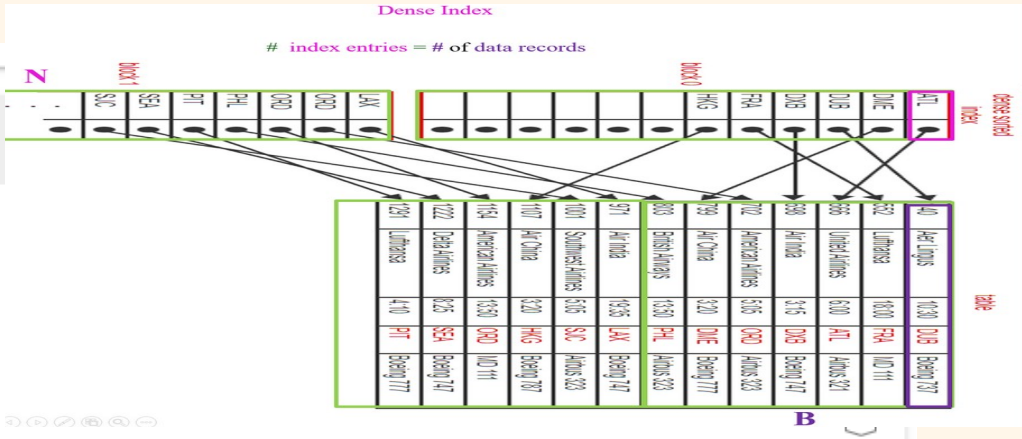
2

1 2 3 4 5 6

ta

?????

VH



CHALLENGE ACTIVITY 14.2.1: Single-level indexes.

B = 5,000 blocks

A table occupies 5000 blocks. FlightNumber is the primary key. A single-level index on FlightNumber occupies 500 blocks. The WHERE clause of a SELECT specifies "FlightNumber = 2780". N = 500 blocks

500 + 1

What is the maximum number of blocks necessary to process the SELECT?

Ex: 5 501 blocks

Assume the table has no index. What is the maximum number of blocks necessary to process the SELECT?

Ex: 5,000 blocks B

3

3
4
5
6

?????

VH

CHALLENGE ACTIVITY

14.2.1: Single-level indexes.

Consider the following scenario:

- A table has 400,000,000 rows.
- Each row is 100 bytes
- Magnetic disk transfer rate is 0.1 gigabytes per second
- Assume 1 gigabyte is approximately 1,000,000,000 bytes

Assuming no free space, a table scan requires approximately how many seconds?

Ex: 1234 seconds

400,000,000 rows

100 bytes / row

0.1 gigabytes / sec

1 gigabyte = **1,000,0000,000** bytes

Table scan

400,000,000 rows * 100 bytes / row = 40000000000 bytes = 40 gigabytes

40 gigabytes * 0.1 gigabytes / sec = 400 sec

4

?????

VH

3000000000 ÷ 4000 =

750,000

3000000000 ÷ 4000 =
750,000

10000000 × 300 =
3,000,000,000

B –blocks of **data records**

Consider the following scenario:

- A table is sorted on indexed column and has 10,000,000 rows — **10,000,000 rows**
- Each row is 300 bytes — 300 bytes / row
- Table and index blocks are 4 kilobytes — 1 block = **4,000** bytes / block
- Assume 1 kilobyte is approximately 1,000 bytes

Assuming the index is sparse how many entries are in the index?

Ex: 1234567 entries

\# index entries = **B** blocks of data records/ rows for the Table

Total number of bytes = **10,000,000 rows** * 300 bytes / row = 3,000,000,000 bytes

B –blocks of **data records** = 3,000,000,000 bytes / **4,000** bytes / block

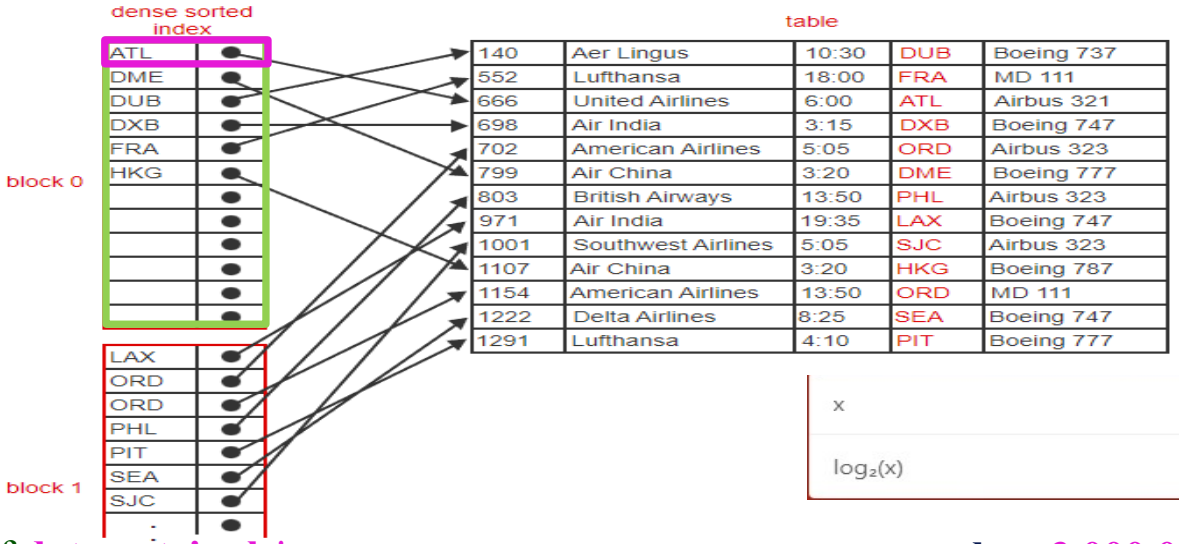
= 750,000 blocks

\# index entries = **750,000**

5

?????

VH



24000000000 ÷ 8000 = 3,000,000

600000000 × 40 = 24,000,000,000

x	3,000,000
$\log_2(x)$	21.517

N –blocks of **data entries k*** $\log_2$ 3,000,000 = **22** blocks

Consider the following scenario:

- A table has 600,000,000 rows. **600,000,000 rows**
- Each index block is 8 kilobytes. 1 block = **8,000** bytes / block
- Each index entry is 40 bytes. 1 index entry = **40** bytes
- Assume 1 kilobyte is approximately 1,000 bytes.

Assuming the index is dense and sorted, a binary search for one row reads approximately how many blocks?

$\log_2$ Ex: 123456 blocks

\# index entries = # of data records = **600,000,000 index entries**

?????

Total number of bytes = 600,000,000 index entries * 1 index entry = **40** bytes =

24,000,000,000 index bytes

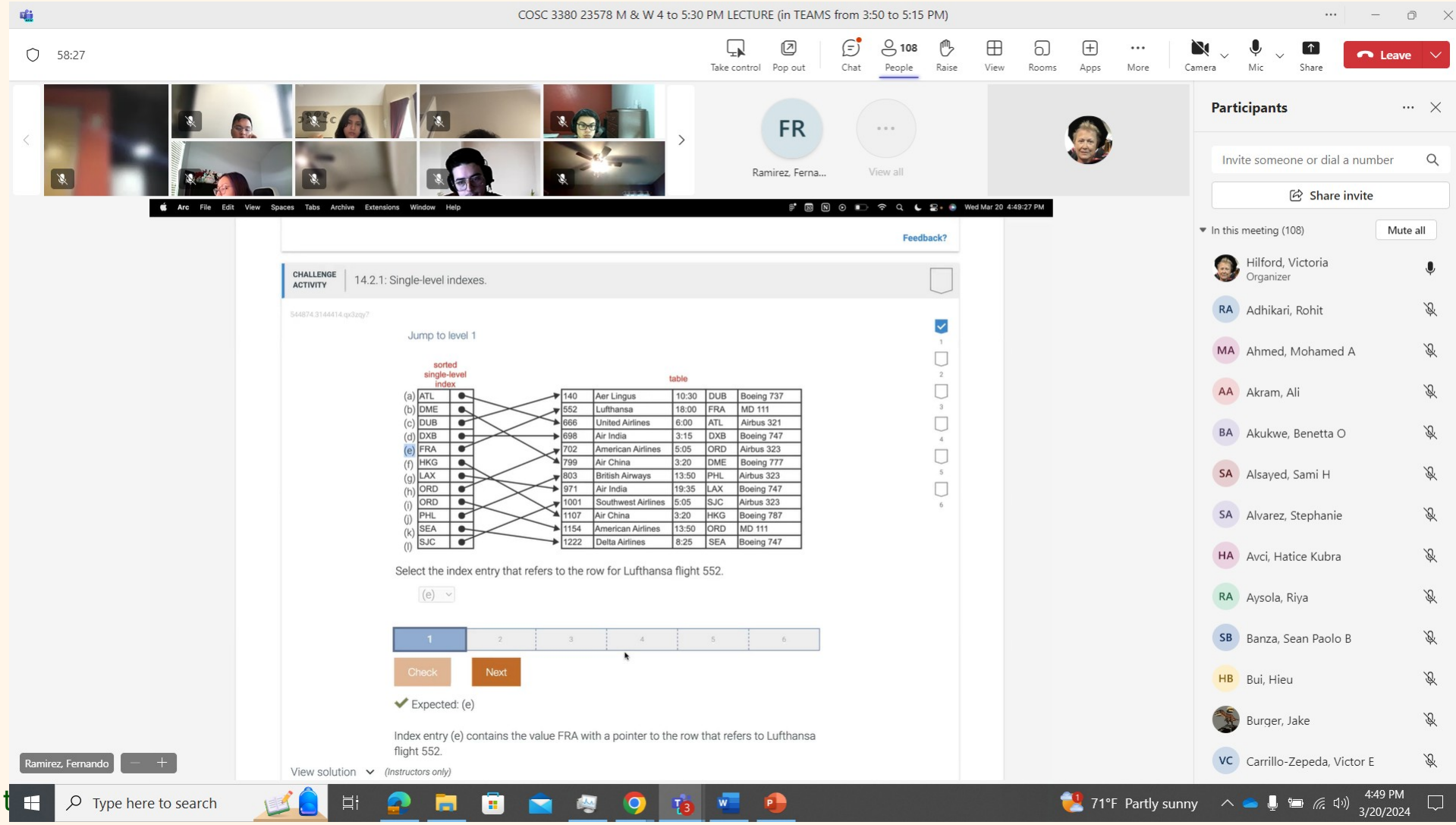
6

N –blocks of **data entries k*** = **24,000,000,000 index** bytes / 1 block = **8,000** bytes / block

= **3,000,000** blocks

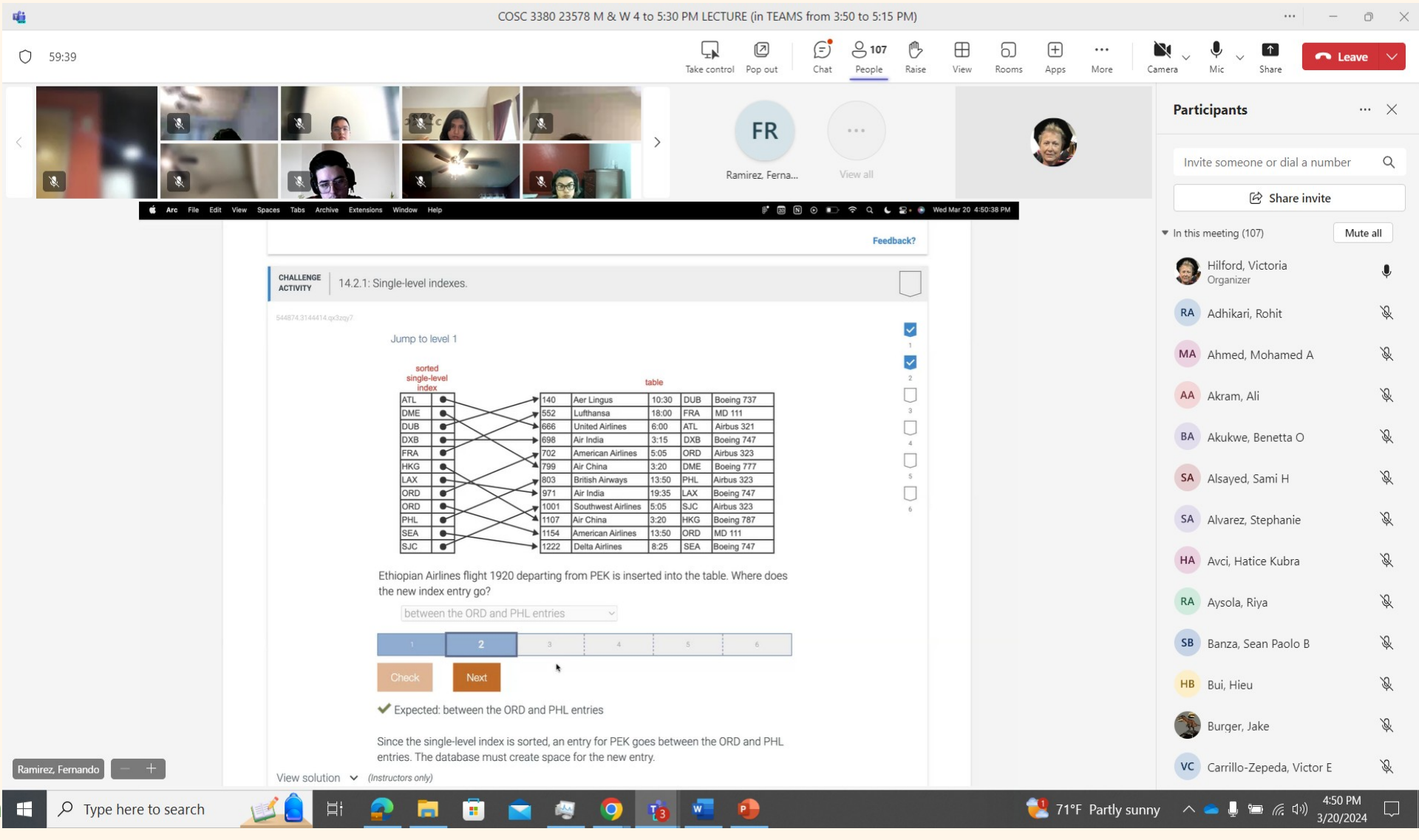
TA time (Fernando)

(CA 14.2.1 Step 1 – Single level index)



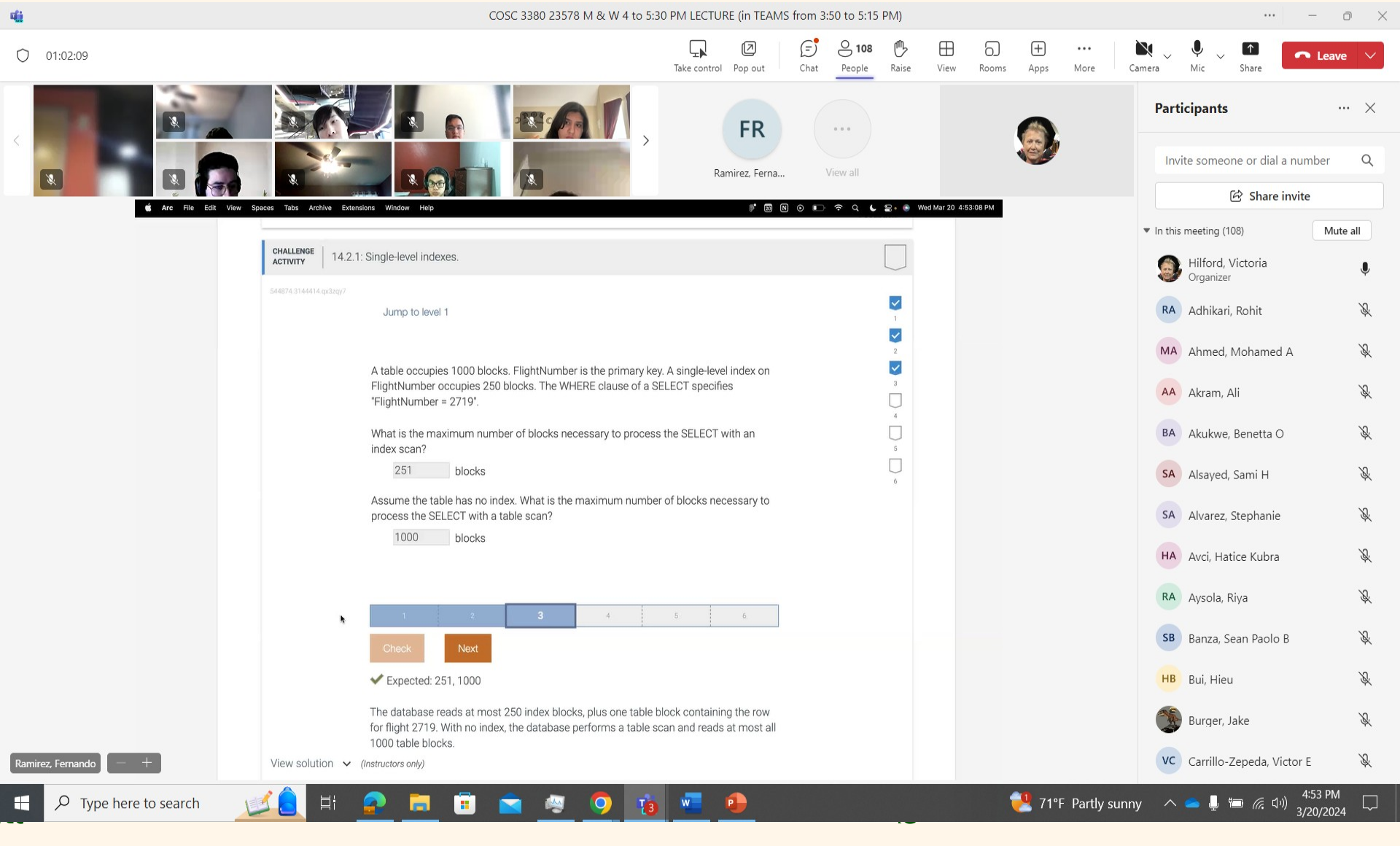
TA time (Fernando)

(CA 14.2.1 Step 2 – Single level index)



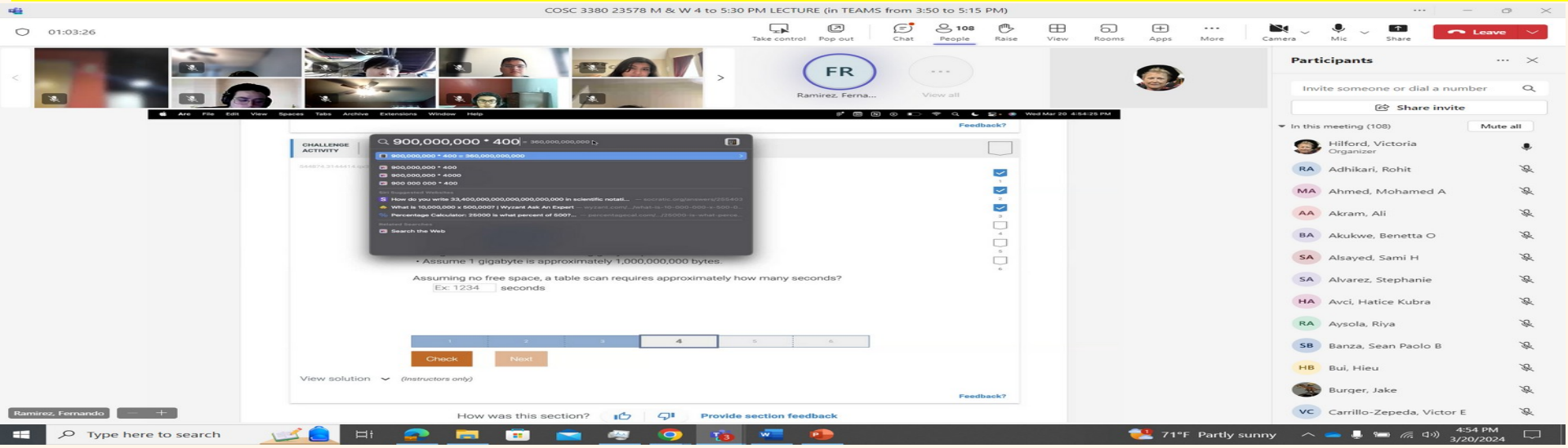
TA time (Fernando)

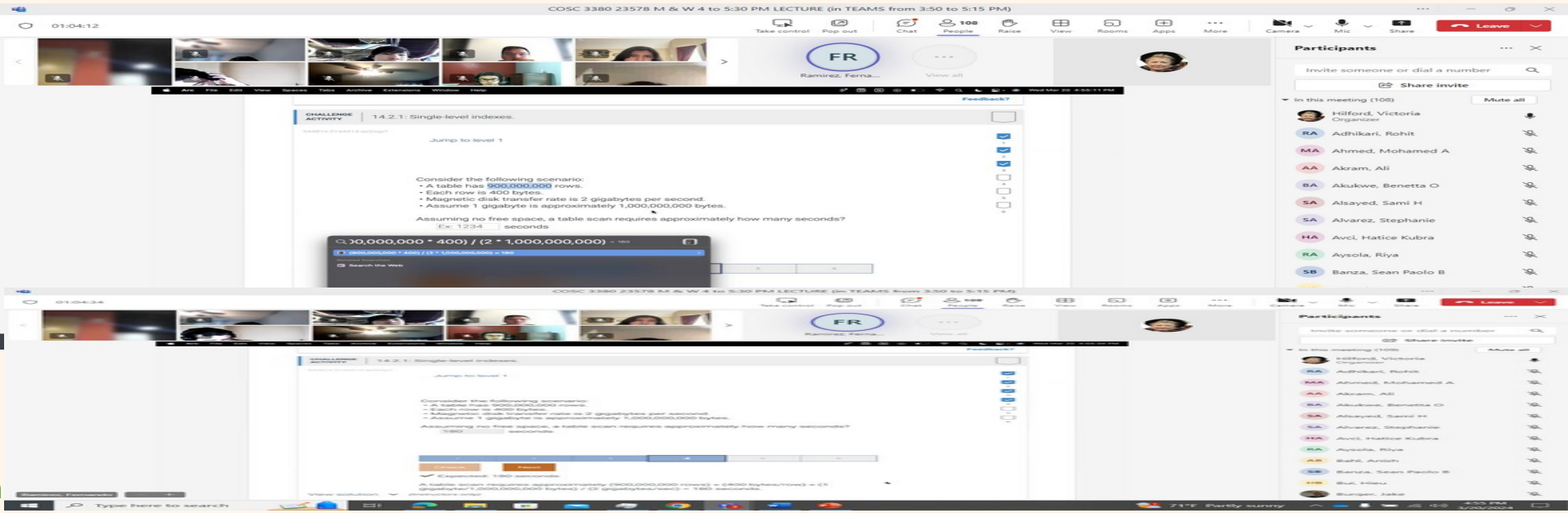
(CA 14.2.1 Step 3 – Single level index)



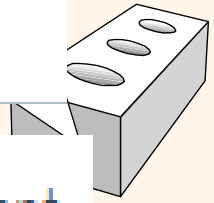
TA time (Fernando)

(CA 14.2.1 Step 4 – Single level index)





Storage II Practice Questions



Which storage engine provides full support for transaction management and locking?

a. MyISAM

b. InnoDB

c. MEMORY

d. MySQL

MySQL storage engines

Logical design specifies tables, columns, and keys. The logical design process is described elsewhere in this material. ***Physical design*** specifies indexes, table structures, and partitions. Physical design affects query performance but never affects query results.

A ***storage engine*** or ***storage manager*** translates instructions generated by a query processor into low-level commands that access data on storage media. Storage engines support different index and table structures, so physical design is dependent on a specific storage engine.

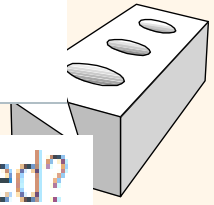
MySQL can be configured with several different storage engines, including:

- InnoDB is the default storage engine installed with the MySQL download. InnoDB has full support for transaction management, foreign keys, referential integrity, and locking.
- MyISAM has limited transaction management and locking capabilities. MyISAM is commonly used for analytic applications with limited data updates.
- MEMORY stores all data in main memory. MEMORY is used for fast access with databases small enough to fit in main memory.

?????

hrke

14. SET 3: 2:STORAGE II



In MySQL with InnoDB, how are primary and secondary indexes created?

a. Primary indexes are automatically created for each primary key. Secondary indexes are created manually for all foreign keys.

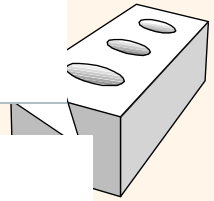
b. Primary indexes are manually created for each primary key. All secondary indexes are created automatically.

c. Primary indexes are automatically created for each primary key. Secondary indexes are created automatically for all foreign keys.

d. Primary indexes are manually created for each primary key. Secondary indexes are created manually for all foreign keys.

?????

hrke



In MySQL with InnoDB, tables with a primary key have a ____ structure, while those without a primary key have a ____ structure.

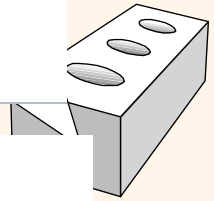
a. heap, sorted

b. sorted, heap

c. sorted, hash

d. cluster, heap

14. SET 3: 2:STORAGE II



What approach can a database administrator use to assess the effectiveness of indexes when investigating slow queries?

a. Run EXPLAIN on queries

b. Change the primary key

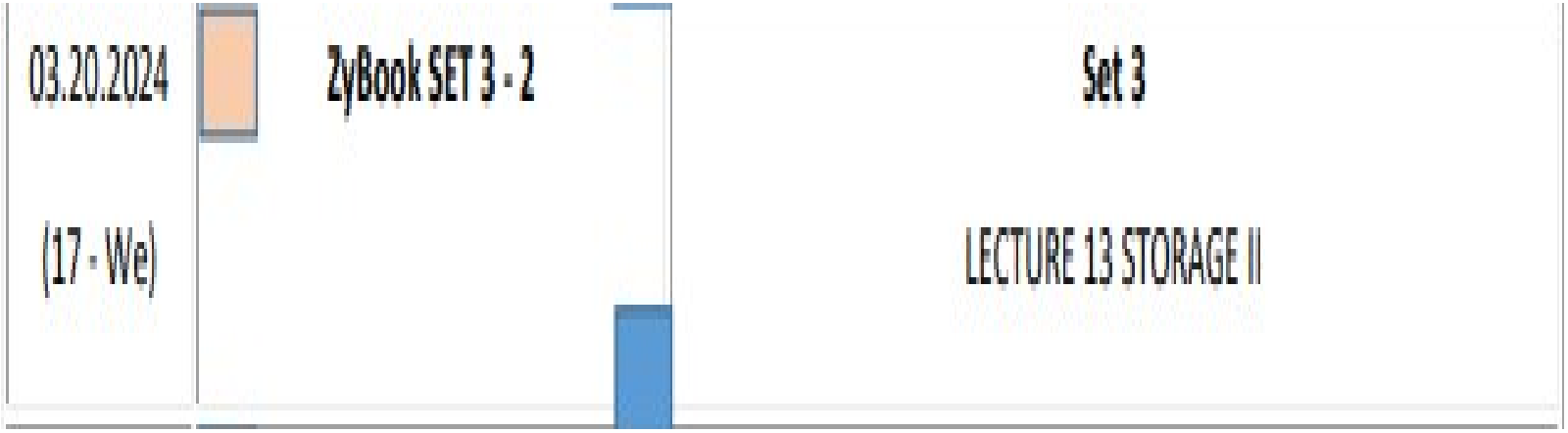
c. Inspect query logs

d. Partition large tables

?????

hrke

At 5:00 PM .



12. SET 3 Empty

14. SET 3: 2:STORAGE II Hidden 0%

VH work on
SET 3 – 2: STORAGE II

Next

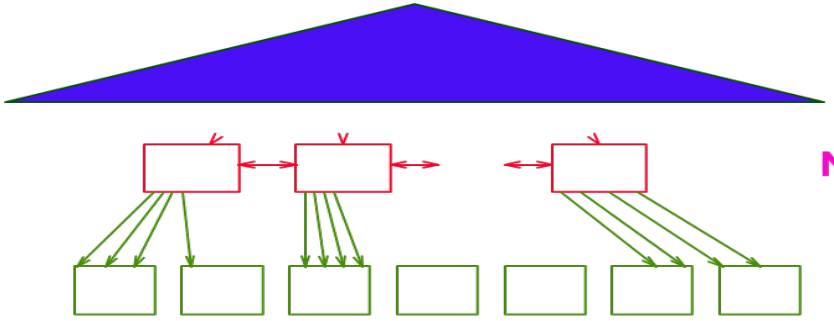
03.25.2024

(18 - Mo)

ZyBook SET 3- 3

Set 3

LECTURE 14 STORAGE III B TREE INDEX



$$\text{Cost} = \lceil \log_F N \rceil + 1 = \text{Tree Level}$$

F = # pointers/ Index Page, N = # Leaf (Data Entry k*) blocks

N blocks of Data Entries k*

B blocks of Data Records

☐ 12. SET 3 Empty

☐ 15. SET 3 - 3:STORAGE III B TREE INDEX Hidden 0% 0%

VH, unHIDE

From 5:05 to 5:15 PM – 5 minutes.

03.20.2024 (17 - We)	ZyBook SET 3 - 2	Set 3 LECTURE 13 STORAGE II

CLASS PARTICIPATION 20 points — 20% of Total

STORAGE II

Class 17 END PARTICIPATION
Not available until Mar 20 at 5:05pm | Due Mar 20 at 5:15pm | 100 pts

VH, publish

This is a synchronous online class.

Attendance is required.

Recording or distribution of class materials is prohibited.

1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**

At 5:15 PM.

End Class 17

VH, unhide ZyBook Section 15.

VH, Download Attendance Report
Rename it:
3.20.2024 Attendance Report FINAL

VH, upload Class 17 to CANVAS.