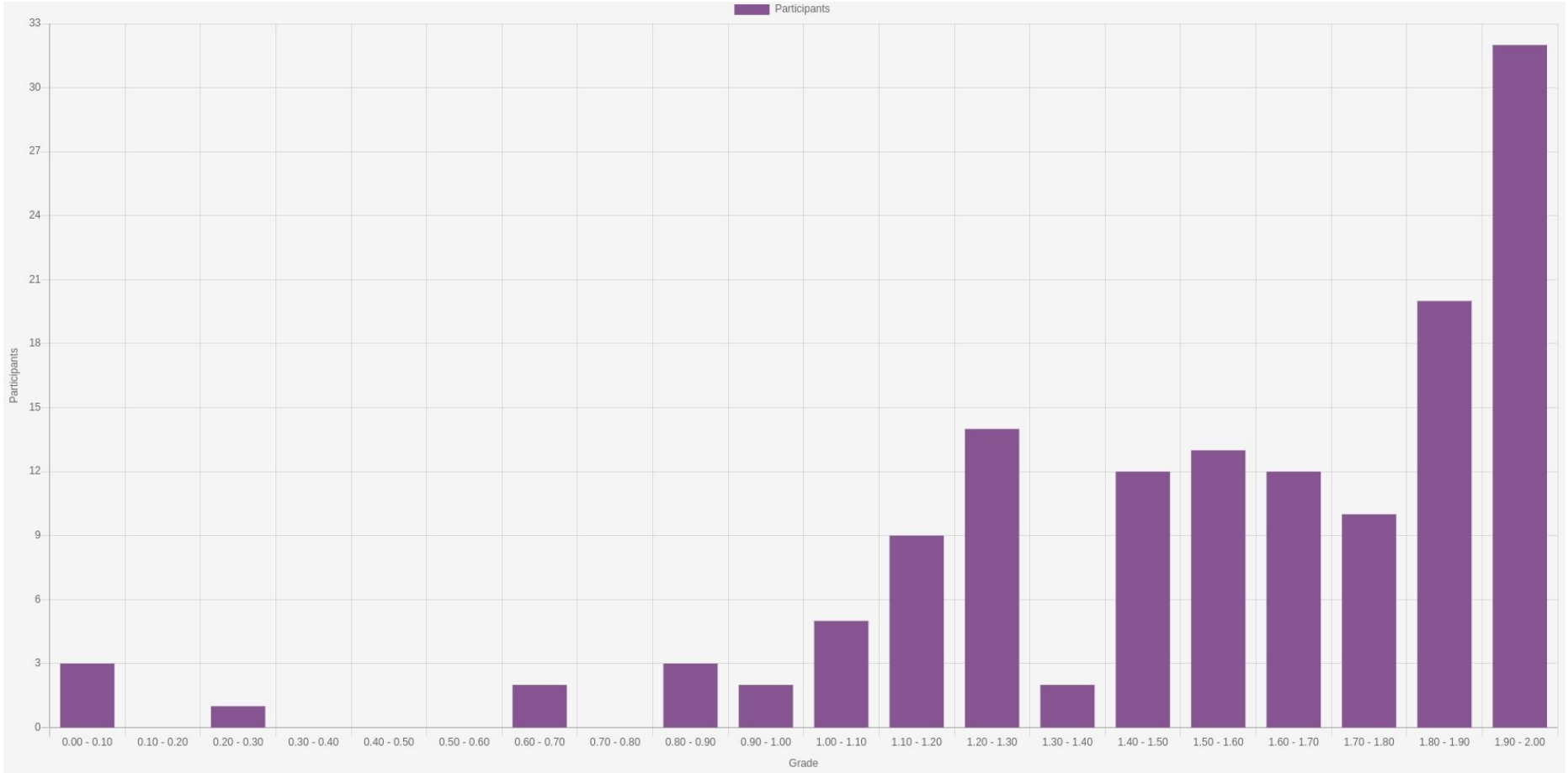
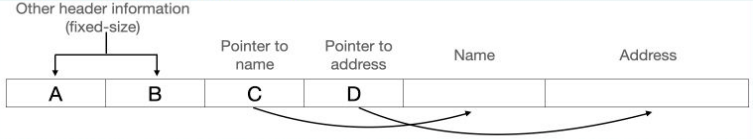
Database Technology

Exercise 2: Review


Participants
33
30
27
24
21
18
15
12
9
6
3
0
Participants
0.00 - 0.10
0.10 - 0.20
0.20 - 0.30
0.30 - 0.40
0.40 - 0.50
0.50 - 0.60
0.60 - 0.70
0.70 - 0.80
0.80 - 0.90
0.90 - 1.00
1.00 - 1.10
1.10 - 1.20
1.20 - 1.30
1.30 - 1.40
1.40 - 1.50
1.50 - 1.60
1.60 - 1.70
1.70 - 1.80
1.80 - 1.90
1.90 - 2.00
Grade

Q1: How many pages are required if the page size is 64 bytes?

Assume the following variable length record structure:



A record requires a fixed number of bytes (A and B in the figure above) for book-keeping tasks. It additionally requires C bytes for a pointer to the (variable-sized) **name** field and D bytes for a pointer to the (variable-sized) **address** field.

The sizes required by the header fields of the record are as follows:

- A = 8 bytes
- B = 4 bytes
- C = 2 bytes
- D = 2 bytes

Consider the following data table. The numbers in parentheses indicate the number of bytes required to store that particular value.

Surname	Address
John Doe (8)	3734 Capitol Avenue (19)
Bruce Wayne (11)	461 Still Street (16)
George Cross (12)	1001 Clover Drive (17)
Johnny Riley (12)	4919 Modoc Alley (16)
Alfred Thaddeus Crane (21)	3895 Owagner Lane (17)
Beryl Hutchinson (16)	4558 Twin Willow Lane (21)

Furthermore, assume that:

- Each variable sized field requires 2 additional bytes (apart from raw storage requirement)
- Each page consists of Page Header and Page Data
- Page Header has a fixed size of 12 bytes
- The tuples are stored as row store or NSM
- The records can span multiple blocks, and do not require additional overhead (for example fragment bits etc) apart from the storage requirement for the record
- The records can be placed one after the other without any additional overhead (for example fragment pointers etc.) to maintain as shown below:

Size of a record?	A+B+C+D+Size(Surname)+2+Size(Address)+2
Size of all records?	Sum of the size of each record in the table
Page requirement?	(Size of all record) / (Size of a single page - 12)

Refer slide 13-21

Q2: Given a 8-byte memory alignment, how many bytes are needed to store an average record of the table created by the following SQL statement?

```
CREATE TABLE Students (
    name CHAR(30),
    lastname CHAR(30),
    Address VARCHAR(256),
    Gender CHAR(1),
    ImmatrNumber INT,
    CurSemester INT,
    birthday DATE);
```

The data types have the following storage requirements in bytes:

- Fixed-length char: number of *reserved* characters
- Variable-length char: number of *actual* characters + 1
- Integers: 4
- Date: 10

You can further assume that:

- Each record has a header of 12 bytes.
- The average length of the values of variable-sized fields is half of the maximum reserved length.

Name	30 + 2 for padding
Lastname	30 + 2 for padding
Address	256/2+1 +7 for padding
Gender	1 + 7 for padding
ImmatrNumber	4 + 4 for padding
CurSemester	4 + 4 for padding
Birthday	10 + 6 for padding
Total bytes	12 + 4 for padding + sum of all

Q3: Pointer swizzling and unswizzling are used to ___ ?

a. map block references from the database address space to the virtual memory space
b. save offsets inside blocks

a. After loading the pages into memory the database address space is converted to virtual memory space.

Q4: Which insertion strategy provides better IO performance during a scan of the table?

Suppose we have a table that is stored sequentially in sorted order on disk. We want to insert a fixed-length record *R* into the table. The block *B* in which we want to insert the record is already full but the next block *B′* has enough empty space.

a. Move a record *R′* from *B* into an overflow block and insert the record *R* into *B*.
b. Move a record *R′* from *B* into *B′* and insert the record *R* into *B*.

b. Move out a record from R’ from B and insert it into B’ and insert the new record R into B.

Q5: Row stores have high tuple reconstruction costs?

No

Q6: Column stores have better performance for inserting tuples than traditional row stores?

No

Q7: How many blocks are retrieved from disk for the following query?

```
select avg(amount) from R
```

Consider the following relation R:

employee_id	payment_id	amount	dates
1	2	50	12/05/2020
2	4	30	21/04/2021
3	3	20	15/07/2020
4	6	100	30/06/2020
5	4	75	11/02/2020
6	5	90	21/03/2020
7	7	310	10/01/2021
8th	7	145	25/10/2020

employee_id	payment_id	amount	dates
1	2	50	12/05/2020
2	4	30	21/04/2021
3	3	20	15/07/2020
4	6	100	30/06/2020
5	4	75	11/02/2020
6	5	90	21/03/2020
7	7	310	10/01/2021
8th	7	145	25/10/2020

Assume that:

- Each field in a record consumes 4 bytes.
- Block size as well as page size is 32 bytes.
- No extra storage overhead is required for any storage model (ie, page and block are same sized and contain purely data.)
- For each query in the questions below, the disk head is above the first block to be read.

NSM (N-arry Storage Model)	4
DSM (Decomposition Storage Model)	1

Q8: Which of the following statement is true for the Vertical Partitioning technique?

✗	The data is mirrored in fractured disks in row layout (NSM) and column layout (DSM).
✗	The data is replicated, twice in row layout (NSM) and twice in column layout (DSM) format, in order to handle disk failures and only OLTP workloads.
✓	The data is stored once but is divided, such that some data is stored in row store layout (NSM) and the remaining is stored in column store layout (DSM).
✗	The data is replicated and enhanced with the right shoes to help elephants be aggressive elephants.
✗	The data is stored twice, once in row layout (NSM) and once in column layout (DSM) format, in order to handle mixed query workloads.

Q9: The main idea of fractured mirrors is to partition a relation across several partitions?

No (Fracture mirror maintains data both in row and column layout.)

Q10: Vertical partitioning aims at reducing the I/O cost for every single incoming query?

No (Vertical partitioning splits a table into multiple tables with limited columns. They work very well for aggregations or queries accessing attributes residing together in a partition. However, when attributes residing in different partition are accessed they perform poor.)