Department of Computer Science

Prof. Dr. Nikolaus Augsten
Jakob-Haringer-Str. 2
5020 Salzburg, Austria
Phone: +43 662 8044 6347
E-Mail: nikolaus.augsten@plus.ac.at

PARIS LODRON UNIVERSITÄT SALZBURG

Databases II — Exam

Winter Semester 2024/25 — 16.05.2025

Name: ______________________ **Student ID:** ______________________

Hints

- Check whether you received all pages of the exam (11 pages).
- Write your name or your student ID on each sheet of the exam and hand in all pages.
- All answers are expected to be written on the exam sheets.
- Clearly highlight and enumerate additional pages that are used for longer answers. Match your text with the according exercise.
- Only use pencils that are permanent and non-red colored.
- Use the notation and techniques discussed in the lecture.
- Exercises with more than one solution are not graded.
- You are allowed to use one A4 sheet with your personal notes (both sides, hand written or printed).
- Exam duration: 90 minutes

Signature ______________________

Grading

Filled by the examiner

Exercise	1	2	3	4	5	6	7	8	9	10	Sum
Total points	1	1	1	1	1	1	1	1	1	1	10
Points reached											

Exercise 1 - *Slotted Page.* 1 Point

We want to store tuples inside the following table:

```
CREATE TABLE books (
  bid INTEGER,
  btitle VARCHAR(20)
);
```

Storing an `INTEGER` requires 4 bytes. Storing strings as a `VARCHAR` requires one byte for each character and an additional byte for termination. As an example, the string `DBMS` requires 5 bytes.

The tupes are stored inside a slotted page with the following properties:

- Size: $2^{13} = 8192$ bytes
- Addressing mode: **Byte addressing** (individual bytes can be addressed)

The following operations are performed in that order:

```
INSERT INTO books VALUES (42, 'Categoriae'); -   Tuple A
INSERT INTO books VALUES (13, 'Analytica-priora'); -   Tuple B
INSERT INTO books VALUES (37, 'De-sophisticis-elenchis'); -   Tuple C
```

Fill in the **missing values/addresses** of the slotted page. Numerical values are expected, using arrows as pointers does not suffice. p_i and g_i correspond to the tuple d_i.

a	f	g_1	p_1	g_2	p_2	g_3	p_3	...	d_3	d_2	d_1
									C	B	A

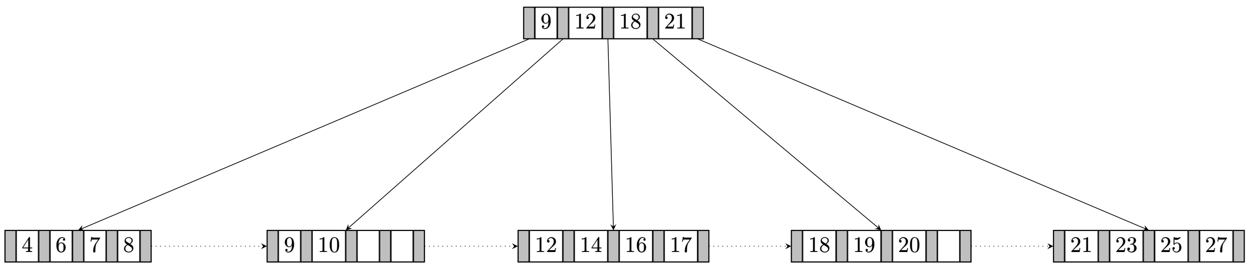
Exercise 2 - *Static Hashing*. 1 Point

Construct a hash index based on attribute *Account Nr* in the following table. The hash value is **first digit** of the attribute value. Each bucket stores up to **3 tuples**. **Overflow chaining** is used to resolve bucket overflows. Note that a pointer to an overflow bucket needs one entry in the bucket. **Illustrate the resulting hash index**.

Owner Name	*Account Nr*	*Balance*
Donovan	579976	2.467
Kermit	585989	7.824
Solomon	489384	6.824
Gavin	579331	3.850
Kelly	630468	8.949
Angelica	676246	6.452
Fredericka	589374	8.888
Caesar	682535	2.776
Chanda	304225	2.014
Patricia	886712	7.726

Exercise 3 - *B⁺-Tree Deletion.* 1 Point

Given the following B$^+$-tree with $m = 5$. Draw the B$^+$-tree after deleting the key **10**.



Exercise 4 - *Indexes with Composite Search Keys.* 1 Point

Consider a relation $R[A, B, C, D]$ and the following four query predicates. State the order of the attributes of (up to) two ordered composite indexes that allow computing all four queries efficiently using those indexes.

1. **WHERE** $B = 42$ **AND** $C < 100$
2. **WHERE** $A < 50$ **AND** $B = 84$
3. **WHERE** $B = 42$ **AND** $C < 100$ **AND** $A = 100$
4. **WHERE** $C = 100$ **AND** $D > 50$ **AND** $B = 42$

Exercise 5 - *External Merge Sort.* 1 Point

Conduct **external merge sort** on the given relation $R[A]$.
A **block can hold 2 tuples**, the memory buffer can hold up to **4 blocks**.

11
25
10
0
13
9
7
22
18
5
16
3
24
20
12
8
6
2
19
14
4
21
23
17
15
1

Exercise 6 - *Join Algorithms.* 1 Point

Which join algorithm (**Hash Join**, **Index Nested Loop Join**) provides the minimal costs in the given scenario? State the **costs** for both **algorithms**.

Compute the **natural join** between relations $R[A, B]$ and $S[B, C]$ with $|R| = 1000$ and $|S| = 12000$ tuples. The relations are stored on $b_R = 250$ and $b_S = 2000$ consecutive blocks. The buffer fits $M = 21$ blocks. There is a **sparse** B$^+$-tree index on $S.B$, where each node in the B$^+$-tree can store up to 20 search keys. The B$^+$-tree index has maximum height.

Exercise 7 - *Efficient Query Processing.* 1 Point

Consider a relation $R[A, B]$. There is a **sparse B$^+$-tree index** on attribute $R.A$ and a **dense hash index** on attribute $R.B$. Values of attribute B are unique. What is the **most efficient strategy** for processing queries of the following type?

$$\sigma_{A<a \,\vee\, B=b}(R)$$

Describe **all necessary steps**.

Exercise 8 - *Query Optimization*. 1 Point

Consider the following relations:

(B)oats(bid, name, color)
(S)ailors(sid, name, rating, age)
(R)eservations(bid, sid, day)

Furthermore, consider the following SQL query:

```
SELECT DISTINCT B.name
  FROM Boats B, Sailors S, Reservations R
 WHERE S.age < 40
   AND B.color = 'blue'
   AND B.bid = R.bid
   AND S.sid = R.sid;
```

a. Write the given SQL query in **algebraic normal form using an operator tree**.
b. Optimize the **operator tree** using **heuristic optimization**.

Exercise 9 — 1 Point

Consider the following schedule S:

T1:	T2:	T3:	T4:
		`write(D)`	
	`read(A)`		
		`write(A)`	
`write(A)`			
	`write(B)`		
	`read(C)`		
		`read(C)`	
	`commit`		
		`read(D)`	
		`commit`	
			`read(A)`
`commit`			
			`read(B)`
			`commit`

Decide for each of the following options whether it is true (**T**) or false (**F**).
Incorrect answers will result in points being deducted.

1. There is only one equivalent serial schedule to S. ☐
2. The precedence graph of S has six edges. ☐
3. S is a recoverable schedule. ☐
4. S is a cascadeless schedule. ☐

Exercise 10 — 1 Point

Can the following schedule be the output of a **strict two-phase locking** scheduler? If yes, add all required lock/unlock instructions. Otherwise, explain why.

T1:	T2:	T3:
	read(A)	
	read(B)	
		read(A)
		write(A)
	write(B)	
	COMMIT	
		read(A)
		read(B)
read(B)		
read(A)		
		COMMIT
COMMIT		