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
Winter Semester 2024/25

Exam
29.01.2025

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

Consider a slotted page with the following properties:

- Size: $2^{13} = 8192$ bytes
- Addressing mode: **Word addressing** (only every second byte can be addressed)

The following tuples O, P, Q are stored in the slotted page:

- d_1: $|O| = 64$ Bytes
- d_2: $|P| = 127$ Bytes
- d_3: $|Q| = 255$ Bytes

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
									Q	P	O

Exercise 2 - *Index Structures*. 1 Point

For the given table, **draw** a 3-level secondary ISAM index on attribute `Stadt`. The inner level of the index should be **dense** while both outer levels should be **sparse**. An **index block** stores **3 entries**.

`Stadt`	`KFZ`
Rom	I
London	GBM
Prag	CZ
Kiew	UA
Berlin	D
Athen	GR
Krakau	PL
Oslo	N
Dublin	IRL
Wien	A

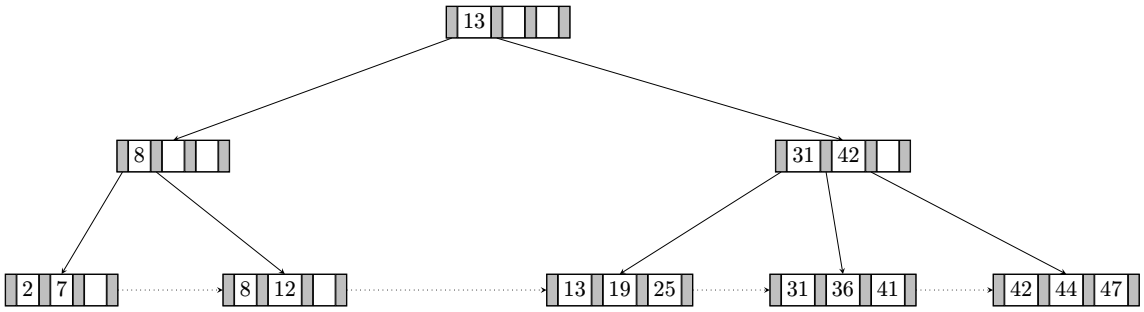
Exercise 3 - *Static Hashing*. 1 Point

Construct a hash index based on attribute *Account Nr* in the following table. The hash value is **first digit** of the attribute value mod 5. The index consists of **5 buckets** and 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 4 - B^+-*Tree Deletion*. 1 Point

Given the following B^+-tree with $m = 4$. Draw the B^+-tree after deleting the key **8**.



Exercise 5 - *External Merge Sort*. 1 Point

Conduct **external merge sort** on the given relation $R[A]$.
A **block can hold 2 tuples**. The size of the buffer consists of **3 blocks**.

90
60
10
0
40
30
45
20
25
5
50
35
45
15
65
100
95
80
85
70

Exercise 6 - *Join Algorithms.* 1 Point

Given **two relations** R and S with the following properties:

$R[A, B, C]$:

- $|R| = 10^7$ tuples stored on $b_R = 20 \cdot 10^3$ blocks
- **dense** B$^+$-tree index on attribute A, $m = 2^8$
- **sparse** B$^+$-tree index on attribute B, $m = 2^7$
- all B$^+$-trees have **maximal height**

$S[B, D, F]$:

- $|S| = 5 \cdot 10^6$ tuples stored on $b_S = 4 \cdot 10^3$ blocks
- **single-level dense** index (ISAM) on attribute B with $5 \cdot 10^4$ blocks
- **single-level sparse** index (ISAM) on attribute D with 40 blocks

Compute the **natural join** $R \bowtie S$ based on an **index nested loop join**.

Compute the most efficient join order ($R \bowtie S$ or $S \bowtie R$) and the according **costs (number of block accesses)**. Accessing one node of the B$^+$-tree is equivalent to one block access. Duplicates do not have to be considered.

Exercise 7 - *Efficient Query Processing*. 1 Point

Given a relation $R[A, B, C]$ with the following properties:

- $|R| = 2.000.000$ tuples.
- Each block of data holds 400 tuples.
- Attribute A holds integer values in the range $[1; 2.000.000]$.
- Attribute B holds integer values in the range $[400.001; 500.000]$.
- Attribute C holds integer values in the range $[100.001; 1.000.000]$.
- The following indexes exist:
  - Sparse B$^+$-tree index on attribute A, $m = 2^8 = 256$, minimal height,
  - dense B$^+$-tree index on attribute B, $m = 2^8 = 256$, minimal height,
  - dense B$^+$-tree index on attribute D, $m = 2^8 = 256$, minimal height.

The following query is to be processed:

$$\sigma_{A>1.800.000 \wedge B=450.000}(R)$$

State the used **strategy (0.5 points)** and calculate the **number of block accesses (0.5 points)** to process the query **in the most efficient way** (accessing 1 node in the B$^+$-tree corresponds to 1 block access).

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

Find a schedule of at least two transactions that is **conflict serializable**, but **not recoverable**. Describe how your example of a non-recoverable schedule can lead to an inconsistent database state.

Exercise 10 — 1 Point

Consider the following schedule and the **two-phase locking** scheduler.

(a) Assume that all transactions want to issue a `write` operation on item `A` in the following order: `T1`, `T2`, `T3`. Complete the schedule by inserting lock requests (e.g., `R:lock-S(A)`) and granted locks (e.g., `G:lock-S(A)`) for all remaining `read` and `write` operations. Denote in the schedule if a transaction has to wait since another transaction holds an incompatible lock.

(b) Draw the wait-for graph. Does the schedule result in a deadlock? Why?

```
T1:         T2:         T3:
_____________________________________
R:lock-S(A)
G:lock-S(A)
read(A)
_____________________________________


                        read(A)
_____________________________________


_____________________________________


_____________________________________


_____________________________________
```