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 2025/26

Exam
28.01.2026

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.
- **Multiple-Choice**: Wrong answers result in point deductions!
- 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 - *Buffer Management.* 1 Point

Consider a system with a (initially empty) buffer of size 4 (i.e., the buffer can store 4 pages). **MRU** (most recently used) is used as the replacement strategy. Consider the following page access pattern of the pages A,B,C,D,E:

A B D D E A E C A B C A E D

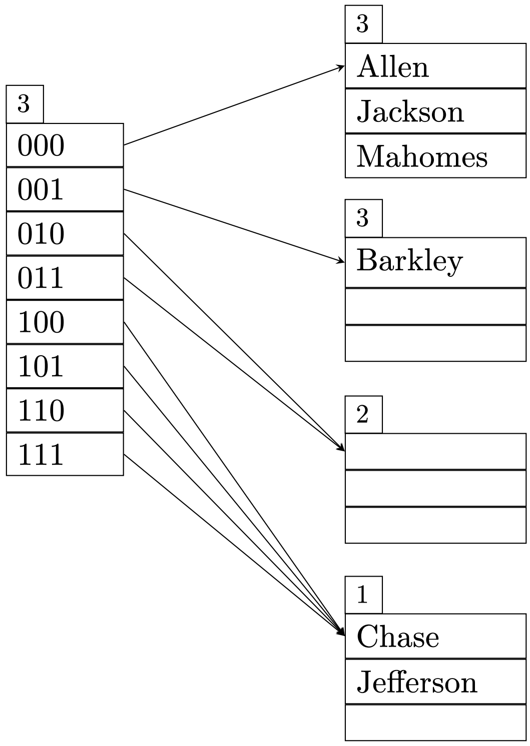
What is the number of disk accesses? Which pages are stored in the buffer after the sequence of page accesses? Complete the following table. Buffer hits are marked with "*".

buffer page 1	A													
buffer page 2		B												
buffer page 3			D	*										
buffer page 4														

Exercise 2 - *Extendible Hashing*. 1 Point

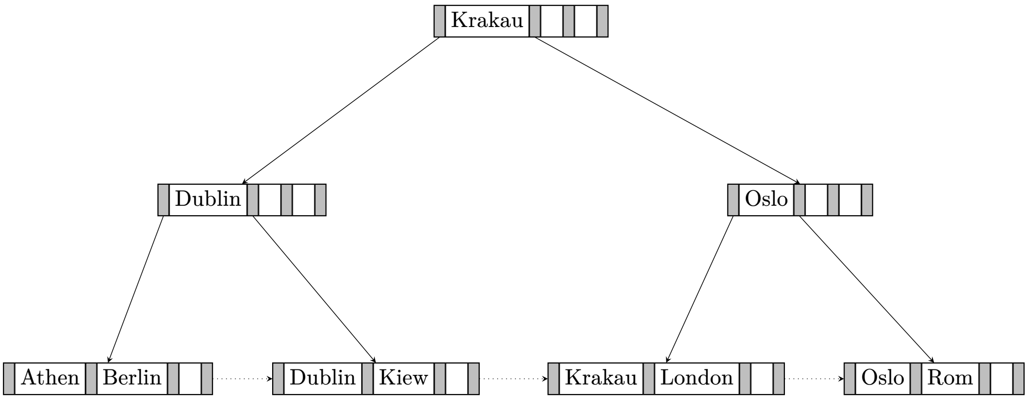
The hash function $h(x)$ returns the binary values as shown in the table. The tuple **Jackson** should be deleted from the hash container. A bucket in the hash container stores up to 3 tuples. The container should be as small as possible. **Illustrate the resulting hash container**.

x	$h(x)$
Allen	00001
Barkley	00101
Chase	10010
Henry	01101
Jackson	00010
Jefferson	10101
Mahomes	00011



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

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



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

Consider an ordered composite index on the attributes *(BrName, AccName, Bal)* in that order. State whether the index structure can be used to efficiently answer the following query predicates. Explain your answers briefly.

(1) **WHERE** AccName="John" **AND** BrName="Salzburg"

(2) **WHERE** AccName="Smith" **AND** 10000<Bal<20000

(3) **WHERE** BrName="Munich" **AND** AccName="Doe" **AND** 10000<Bal<20000

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 **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 - *Pipelining*. 1 Point

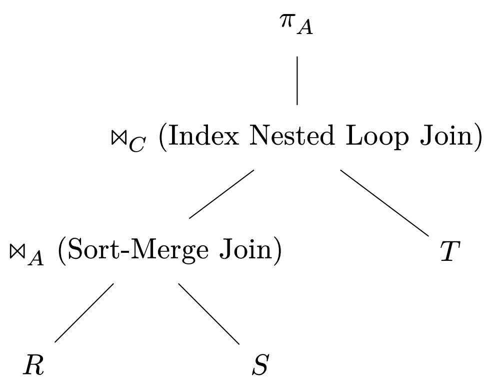
Consider the relations $R[A, B]$, $S[A, C]$, and $T[C, D]$.

- R: 10^6 tuples with a **sparse** B^+-tree index on A.
- S: $5 \cdot 10^5$ tuples.
- T: $2 \cdot 10^6$ tuples with a **dense** B^+-tree index on C.

The logical query is: $\pi_A((R \bowtie_{R.A=S.A} S) \bowtie_{S.C=T.C} T)$.

Note: As in relational algebra (and unlike SQL), the projection π_A removes duplicates (deduplication).

The database optimizer generates the physical query plan including annotations for physical operators shown in the tree below.



(1) In the tree above, **annotate** all edges as either **blocking** or **pipelining**. (For a blocking edge, the consuming operator has to wait for the producing operator to finish before producing the first tuple.)

(2) How can efficient deduplication be implemented in the projection π_A?

(3) Suppose we replace the **Index Nested Loop Join** with a **Hash Join** for the join $S \bowtie_C T$. Explain how this replacement affects your answers to (1) and (2).

Exercise 7 - *Efficient Query Processing.* 1 Point

Consider a relation $R[A]$. There is a **sparse B^+-tree index** of attribute $R.A$ What is the **most efficient strategy** for processing **range queries** of the following type?

$$\sigma_{a<A<b}(R)$$

Describe **all necessary steps**.

Exercise 8 - *Join Size Estimation.* 1 Point

Consider the following 3 relations $R[A, B, C]$, $S[A, D, E]$, $T[D, E, F]$ and their properties:

- $|R[A, B, C]| = 1000$ tuples, $V(R, A) = 100$, $V(R, B) = 200$, $V(R, C) = 300$
- $|S[A, D, E]| = 4000$ tuples, $V(S, A) = 50$, $V(S, D) = 200$, $V(S, E) = 300$
- $|T[D, E, F]| = 2000$ tuples, $V(T, D) = 200$, $V(T, E) = 400$, $V(T, F) = 600$

Attribute values are assumed to be distributed uniformly and independently. Estimate the size of the following query. $(\sigma_{A=100}(R) \neq \emptyset)$.

$$(\sigma_{A=100}(R)) \bowtie S \bowtie T$$

Exercise 9 — 1 Point

Answer the following concisely. For each anomaly provide: (a) a schedule showing operations and commit/rollback, (b) an explanation why it can be a problem, and (c) the lowest SQL isolation level that prevents it (choose one: Read Uncommitted, Read Committed, Repeatable Read, Serializable).

(1) Dirty read

(2) Non-repeatable read

Exercise 10 — 1 Point

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

T1:	T2:	T3:
`read(X)`		
	`write(X)`	
		`read(X)`
`write(X)`		
		`write(X)`

Does the schedule result in a deadlock? If yes, what would happen to the transactions according to the wound-wait deadlock prevention protocol (assume TS(T1) < TS(T2) < TS(T3))?