Tutorial 06
Query Optimisation
Big Data Engineering

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

June 4, 2025

Repetition - Question 1

Question

What are the steps the query optimisation process consists of?

Repetition - Question 1

Question

What are the steps the query optimisation process consists of?

Solution

1. SQL
   ↓ canonical translation
2. annotated relational algebra/logical plan
   ↓ heuristic/rule-based optimisation
3. transformed logical plan
   ↓ cost-based optimisation
4. physical plan
   ↓ code generation
5. executable code

Strictly speaking, code generation is **not** a part of query optimisation, but an independent problem.

Repetition - Question 2

Question

If a predicate p of a selection σ_p has a high selectivity, then ...

(A): many tuples fulfil p

(B): it holds $|\text{output relation}| < |\text{input relation}|$

(C): a few tuples fulfil p

(D): it holds $|\text{output relation}| > |\text{input relation}|$

Repetition - Question 2

Question

If a predicate p of a selection σ_p has a high selectivity, then ...

Solution

The correct answers are (B) and (C):

A high selectivity means, that the output relation is very small compared to the input relation. Note that we speak of **high** selectivity when having **small** numerical values.

(A): Wrong, in this case input and output relation would have similiar size.

(B): Correct, this holds for every selective predicate.

(C): Correct.

(D): Wrong, this never holds as there is no way to obtain more tuples than available in the input relation.

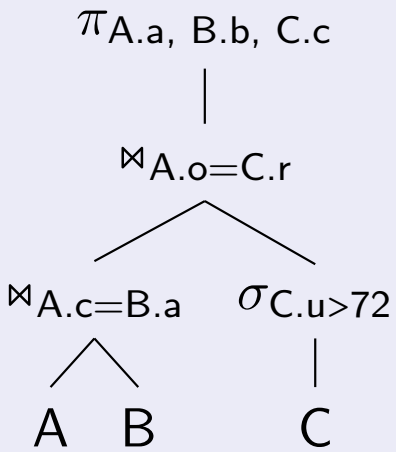
Repetition - Question 3

Question

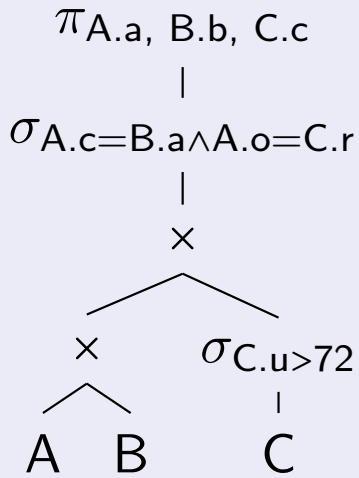
What is the logical plan of the following SQL query according the to canonical translation?

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.c=B.a AND A.o=C.r AND C.u > 72;
```

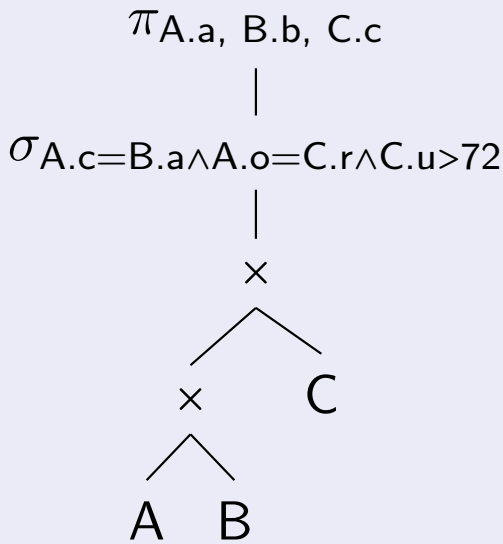
(A):



(B):



(C):



Repetition - Question 3

Question

What is the logical plan of the following SQL query according the to canonical translation?

```
SELECT A.a, B.b, C.c
FROM   A, B, C
WHERE  A.c=B.a AND A.o=C.r AND C.u>72;
```

Solution

The correct answer is (C):

No further optimisations (like e.g. Predicate Pushdown) are performed in the canonical translation.

Repetition - Question 4

Question

How does the following query look like after performing the predicate pushdown?

$\sigma_{S.i=T.v \wedge S.b=30 \wedge T.x \geq 19}\ (S \bowtie_{S.v=T.i}\ T)$

(A): $\sigma_{S.i=T.v}\ (S \bowtie_{S.v=T.i \wedge S.b=30 \wedge T.x \geq 19}\ T)$

(B): $\sigma_{S.i=T.v}\ ((\sigma_{S.b=30}\ S) \bowtie_{S.v=T.i}\ (\sigma_{T.x \geq 19}\ T))$

(C): $S \bowtie_{S.i=T.v \wedge S.v=T.i \wedge S.b=30 \wedge T.x \geq 19}\ T$

(D): $(\sigma_{S.b=30}\ S) \bowtie_{S.i=T.v \wedge S.v=T.i}\ (\sigma_{T.x \geq 19}\ T)$

Repetition - Question 4

Question

How does the following query look like after performing the predicate pushdown?

$\sigma_{S.i=T.v \wedge S.b=30 \wedge T.x \geq 19}\ (S \bowtie_{S.v=T.i} T)$

Solution

The correct answer is (B):

$\sigma_{S.i=T.v}\ ((\sigma_{S.b=30}\ S) \bowtie_{S.v=T.i} (\sigma_{T.x \geq 19}\ T))$

Repetition - Question 5

Question

How does the following query look like after performing the projection pushdown?

$\pi_{A.y,B.x}\ (A \bowtie_{A.x=B.x} B)$

(A): $\pi_{A.y,B.x}\ ((\pi_{A.x,A.y}\ A) \bowtie_{A.x=B.x} (\pi_{B.x,B.y}\ B))$

(B): $\pi_{A.y,B.x}\ ((\pi_{A.y}\ A) \bowtie_{A.x=B.x} (\pi_{B.x}\ B))$

(C): $(\pi_{A.x,A.y}\ A) \bowtie_{A.x=B.x} (\pi_{B.x}\ B)$

(D): $\pi_{A.y,B.x}\ ((\pi_{A.x,A.y}\ A) \bowtie_{A.x=B.x} (\pi_{B.x}\ B))$

Repetition - Question 5

Question

How does the following query look like after performing the projection pushdown?

$\pi_{A.y,B.x}\ (A \bowtie_{A.x=B.x} B)$

Solution

The correct answer is (D):

$\pi_{A.y,B.x}\ ((\pi_{A.x,A.y}\ A) \bowtie_{A.x=B.x} (\pi_{B.x}\ B))$

Exercise 1

Question

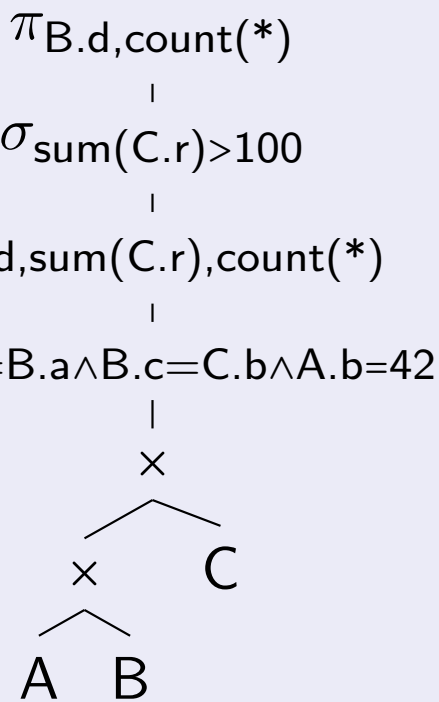
Consider the following SQL query.

```
SELECT    B.d, COUNT(*)
FROM      A, B, C
WHERE     A.b = B.a AND B.c = C.b AND A.b = 42
GROUP BY  B.d
HAVING    SUM(C.r) > 100;
```

1. Translate the SQL query into a logical plan using the canonical translation.
2. Now, break up all conjunctions in the logical plan into predicates.
3. Following that, push down all predicates as far as possible.
4. Combine all selections and Cartesian products into joins.
5. The plan offers more optimisation strategies than those demonstrated in the lecture. Explain and apply this optimisation.

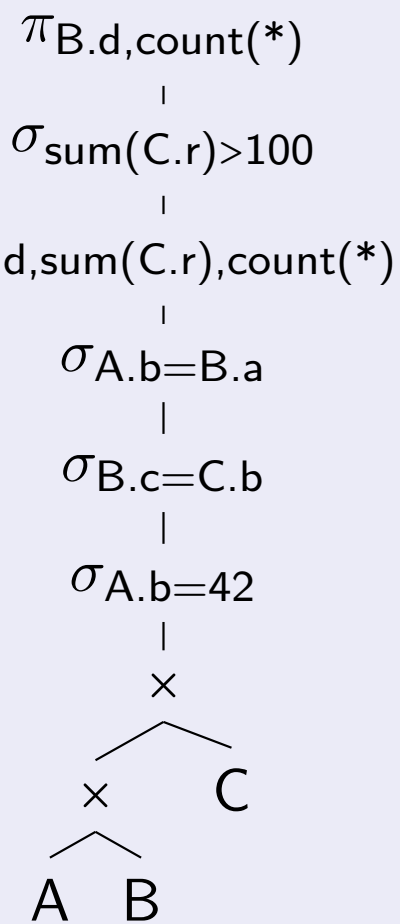
Exercise 1.1

Solution



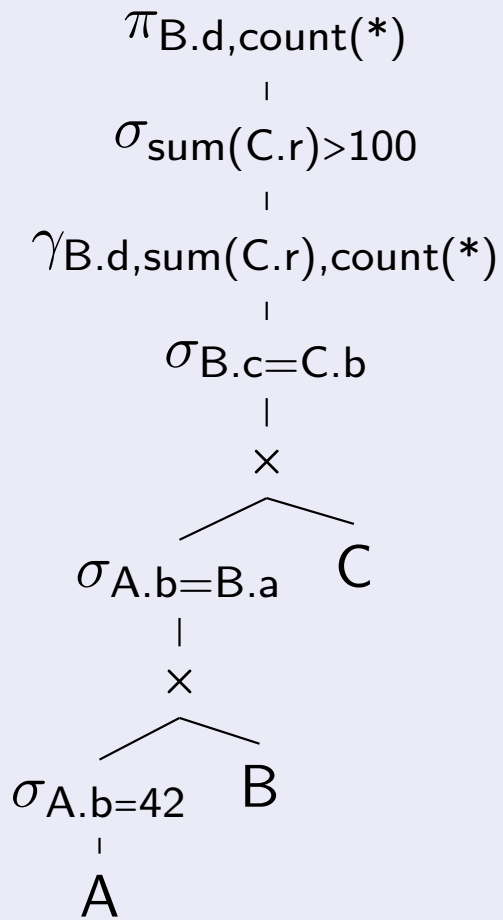
Exercise 1.2

Solution



Exercise 1.3

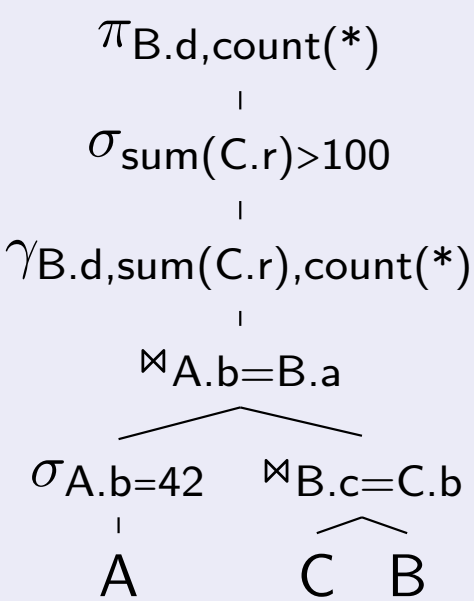
Solution

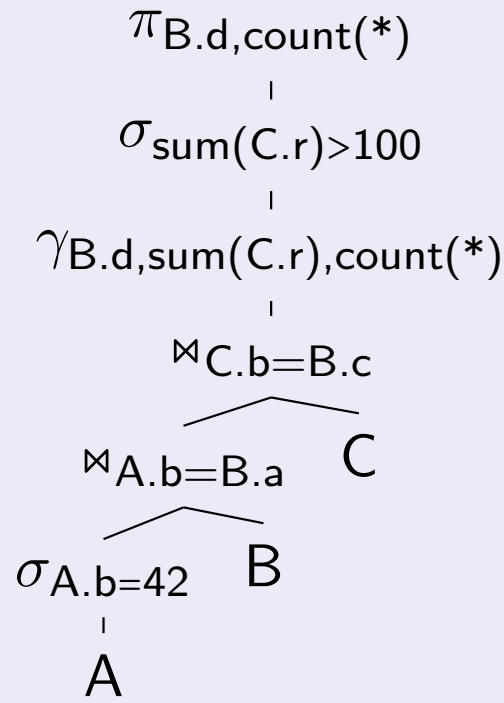


Exercise 1.4

Solution

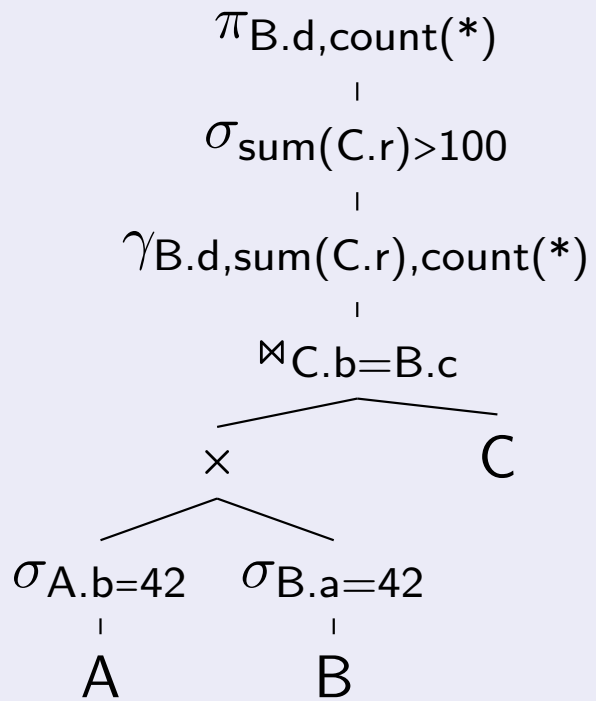
Determining a join order is not part of heuristic optimisation. As a result, multiple solutions are possible.





Exercise 1.5

Solution

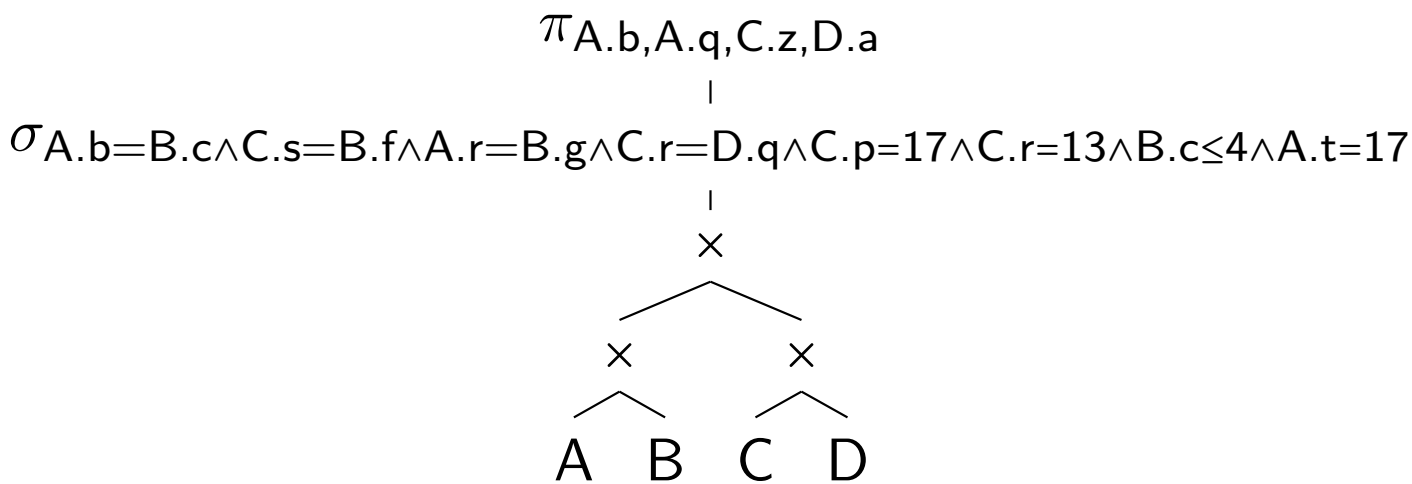


Rule: Transitivity
Let A and B be attributes of any table, c a constant and op a comparison operator. If $A = B$ holds and $A\ op\ c$, then $B\ op\ c$ also holds. Following that, one can add another selection, reducing the amount of data to be considered when checking $A = B$. Further, if $A = c$ holds (and thus also $B = c$), the conditions $A = B$ can be removed completely at the join and be replaced by a Cartesian product.

Exercise 2.1

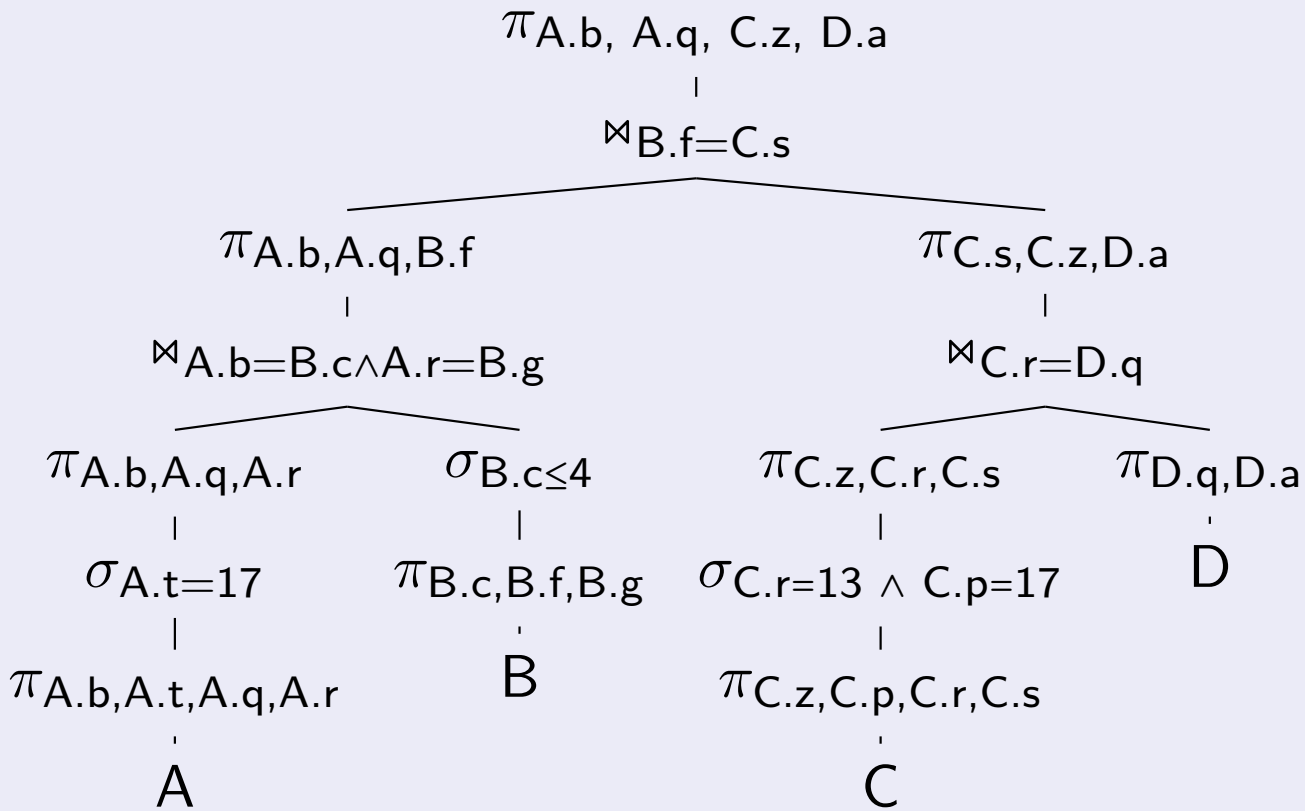
Question

Optimise the logical plan below using the rules demonstrated in the lecture. Use the join order $(A \bowtie B) \bowtie (C \bowtie D)$.



Exercise 2.1

Solution



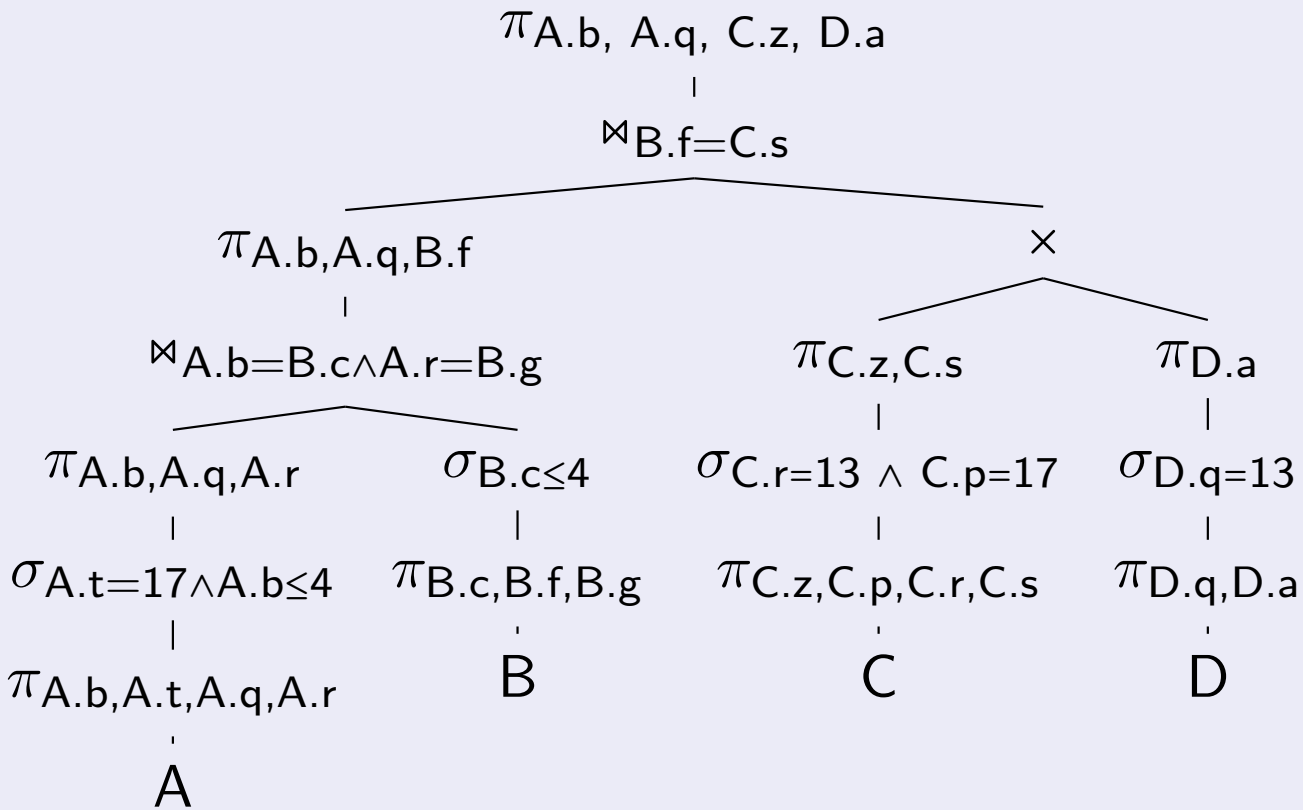
Exercise 2.2

Question

The plan can be optimised further using the rule established in Exercise 2.5 (transitivity). Apply those optimisations.

Exercise 2.2

Solution



Exercise 2.3

Question

Why do we try to execute selections and projections as early as possible? Compare projections and selections in terms of their similarities and differences.

Exercise 2.3

Solution

We try to execute them as early as possible as they reduce the amount of data ($\rightarrow$ similarity). That means that operations, joins in particular, have to process less data and are thus faster. Here, a selection reduces the data horizontally by removing tuples/cells, while a projection reduces the data vertically by removing attributes/columns ($\rightarrow$ difference).

Exercise 3.1

Question

Assume you have access to a perfect cost model. Why is it still difficult to determine the optimal plan?

Exercise 3.1

Solution

A cost model only assigns the costs to a plan. In order to find the cheapest plan, you first have to enumerate it and pass it to the cost model. The bigger the query, the bigger the amount of plans that have to be enumerated. Especially the amount of join orderings increases exponentially with the number of tables. Accordingly, if we have many tables, not all join orderings can potentially be enumerated, which might lead to missing the best plan.

Exercise 3.2

Question

Why do we have multiple cost models? What does it mean to call a plan *optimal*?

Exercise 3.2

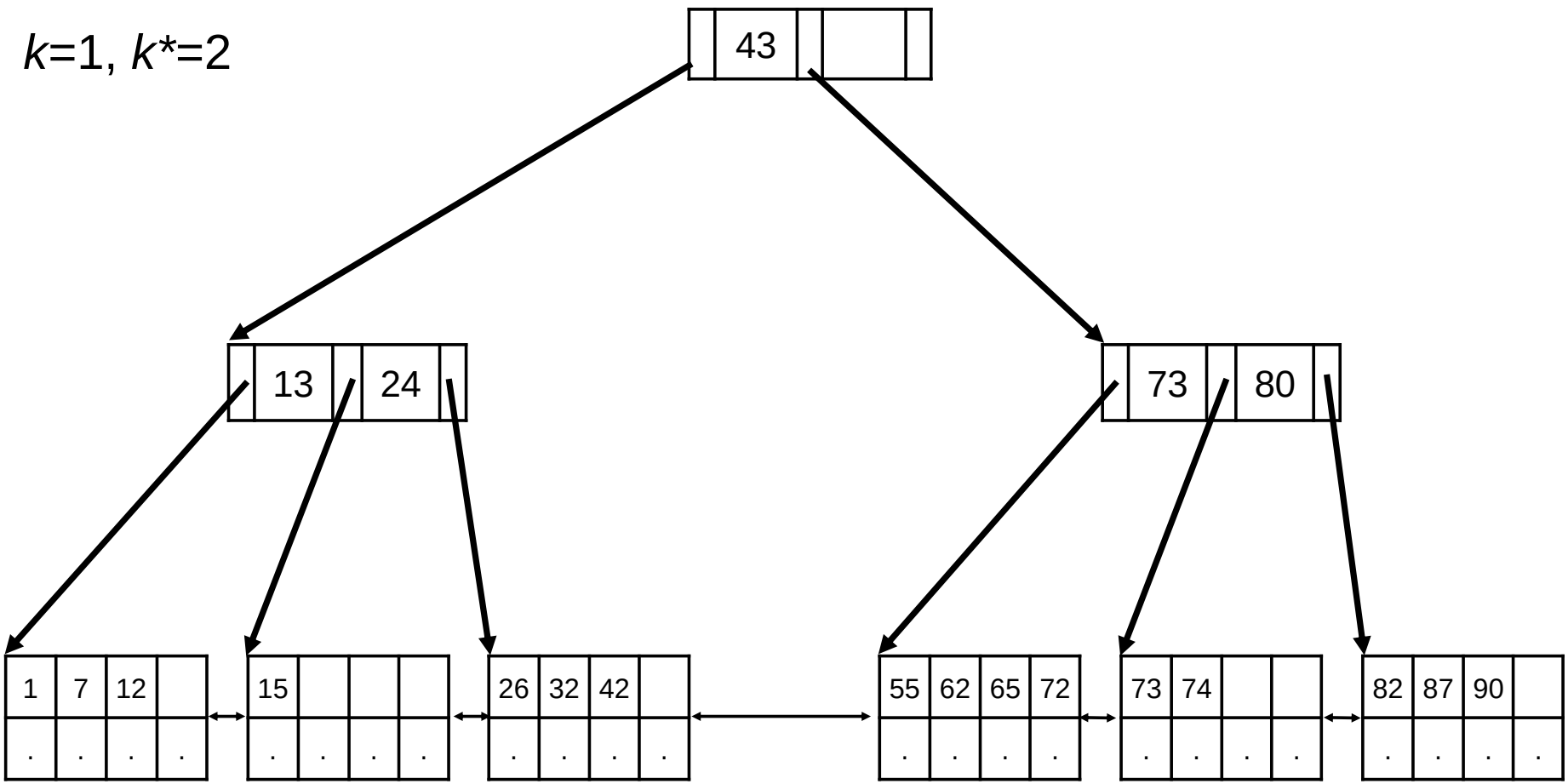
Solution

- We follow different goals with different cost models. For example, on a mobile device, it could be more important to reduce the energy consumption. On the other hand, in a big data center, it might be more important to reduce the query times.
- If we find an optimal plan using a cost model, it is only optimal for this particular cost model. If we change the cost model, the plan could be suboptimal.

Exercise 4.1

Question

For the following B-Tree, explain which invariant(s) do(es) not hold.



Exercise 4.1

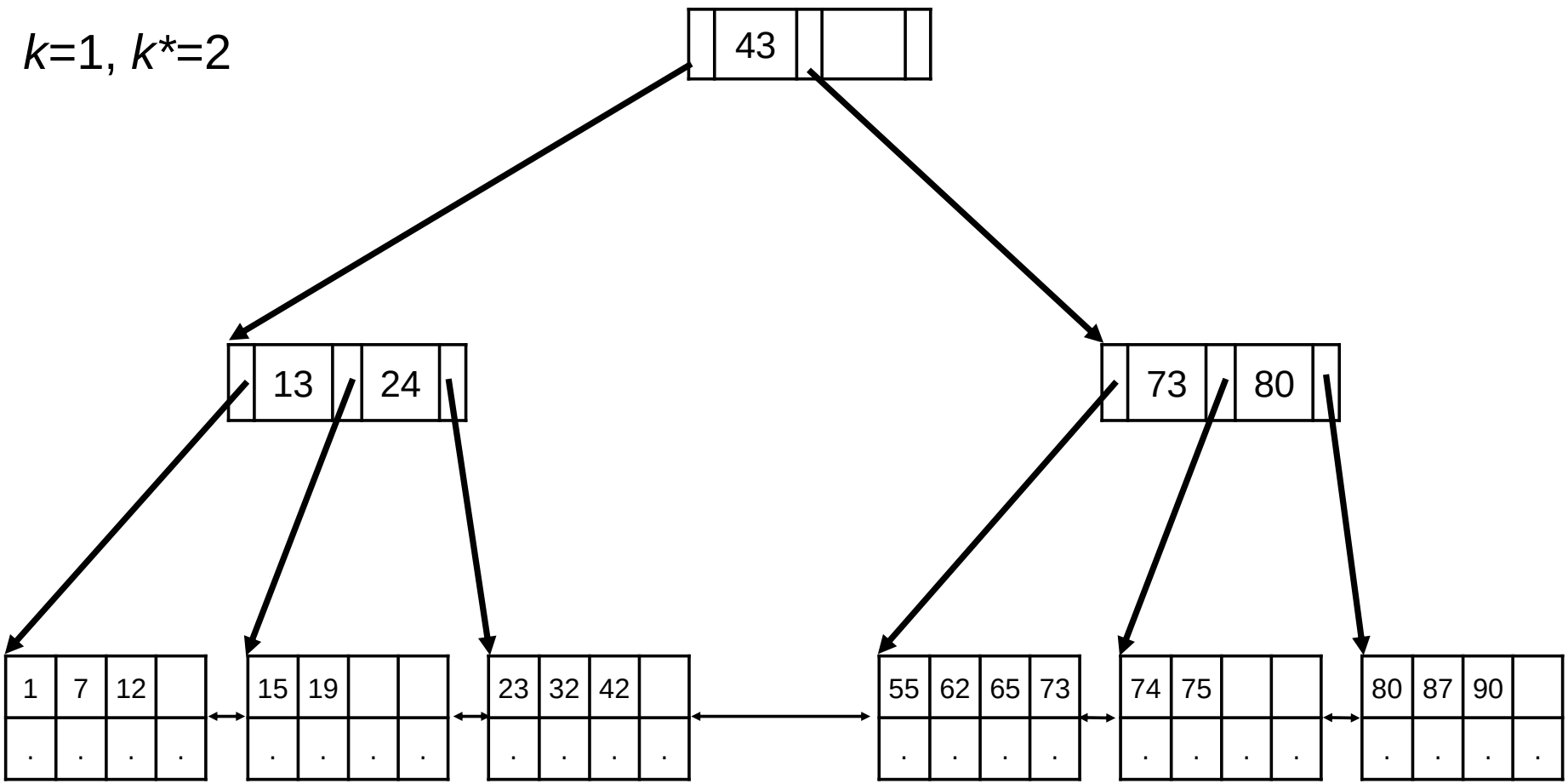
Solution

The second leaf node does not fulfill the invariant that all nodes must be at least half full, as it only contains 1 key.

Exercise 4.2

Question

For the following B-Tree, explain which invariant(s) do(es) not hold.



Exercise 4.2

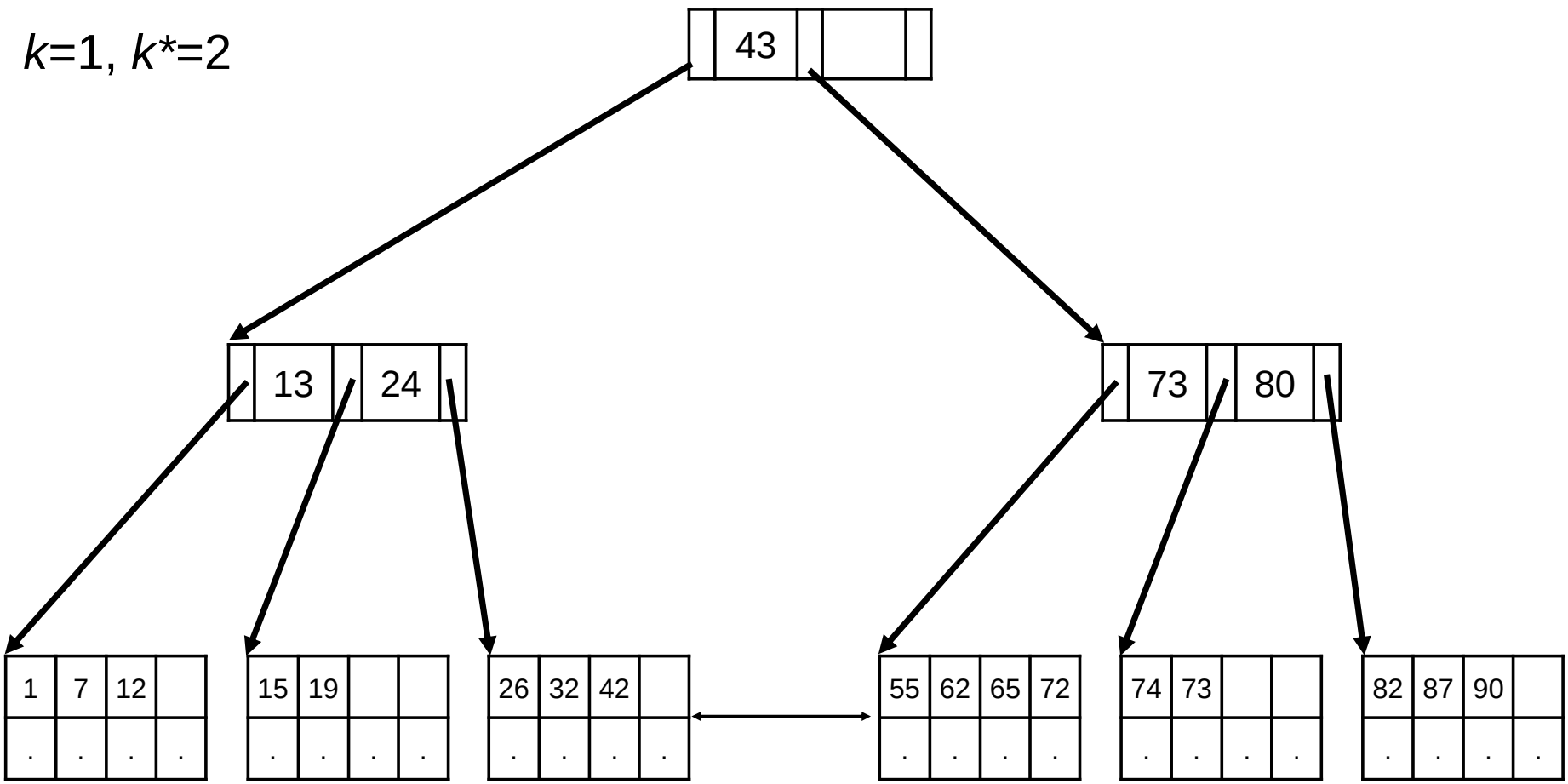
Solution

The keys contained in the left subtree are partially wrong. Either the pivot element 24 in the left inner node is wrong as its right child contains a 23 or the 23 has to be moved to the "second" leaf node (containing the keys 15 and 19). Further, assuming the partitioning strategy from the lecture, the left key 73 in the right inner node is wrong, as its left child contains a 73.

Exercise 4.3

Question

For the following B-Tree, explain which invariant(s) do(es) not hold.



Exercise 4.3

Solution

The pointers between the children of the individual nodes are missing. Further, the second child of the right inner node is not sorted.

Correction Mini-Test 05 - Common Mistakes

Exercise 1:

- 1.: Using joins or predicate pushdowns. Both of these are not part of the canonical translation.
- 2.: Not using joins or projection pushdowns.
- 2.: Forgetting necessary projections after predicates or joins.
- 3.: Not using transitivity to add the selection 'C.u=4'.
- 3.: Finding the optimisation 'C.u=4' but missing to adapt projections and to replace the join by a Cartesian product.

Exercise 2:

- Not explicitly stating which invariant is violated.
- Not identifying the specific location in the B-tree where the invariants are violated.