Architektur und Implementierung von Datenbanksystemen
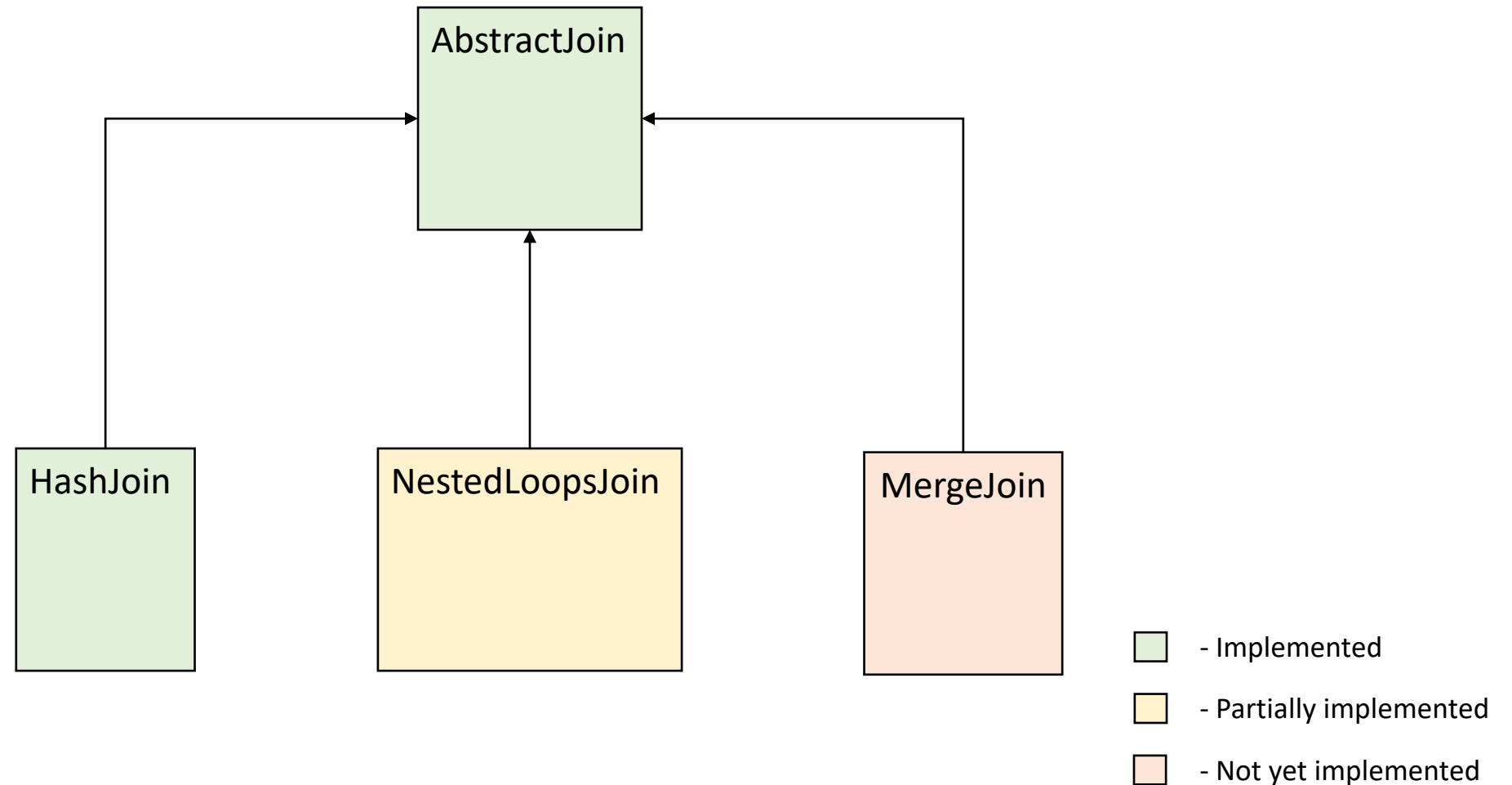
Task 7

Team 3 - Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

Requirements

- Join algorithm(s)

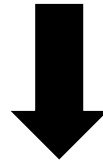- Grouping/Aggregation

- Complexity

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

universität
innsbruck

Join

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

Hashjoin

```
explain pruefen join pruefen.VorlNr = hoeren.VorlNr hoeren;
+-----------------------------------------------------------------------------------+
| Operator                                                                          |
|-----------------------------------------------------------------------------------|
| -->NestedLoopsJoin(type=inner, natural=False, condition=(pruefen.VorlNr = hoeren.VorlNr)) |
| ---->TableScan(pruefen)                                                           |
| ---->TableScan(hoeren)                                                            |
+-----------------------------------------------------------------------------------+
```

Optimizer:

If condition is equality or conjunction of equalities

```
+----------------------------------------------------------------------------------+
| Operator                                                                         |
|----------------------------------------------------------------------------------|
| -->HashJoin(type=inner, natural=False, condition=(pruefen.VorlNr = hoeren.VorlNr)) |
| ---->TableScan(pruefen)                                                          |
| ---->TableScan(hoeren)                                                           |
+----------------------------------------------------------------------------------+
```

Hashjoin: Implementation

- Build Phase:

pruefen join pruefen.VorlNr = hoeren.VorlNr and pruefen.MatrNr = hoeren.MatrNr hoeren;

```
hashtable = dict()

references = get_references(table1, condition) #(pruefen.VorlNr, pruefen.MatrNr)

for record in table1:
    key = record.values_at_references(references) #(3004, 203042)
    hashtable[key].append(record)

return hashtable
```

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

universität
innsbruck

Hashjoin: Implementation

- Probe Phase:

```
pruefen join pruefen.VorlNr = hoeren.VorlNr and pruefen.MatrNr = hoeren.MatrNr hoeren;
```
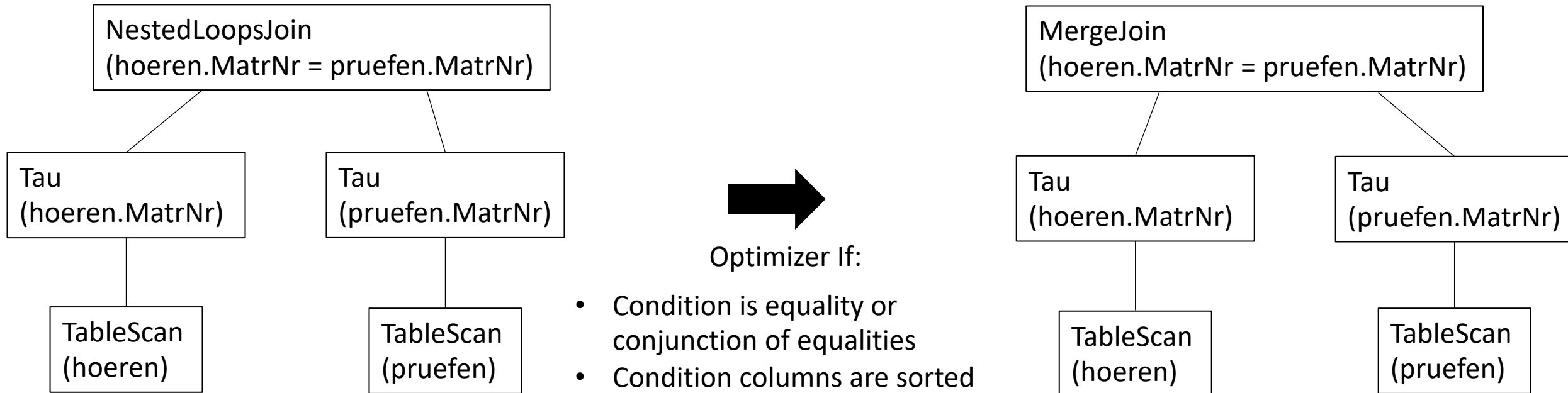
```
references = get_references(table2, condition) #(hoeren.VorlNr, hoeren.MatrNr)
hashtable = build_hash(table1)
result_records = []

for record in talbe2:
        key = record.values_at_references(references) #(3004, 203042)
        if key in hashtable:
                result_records.extend(build_records(hashtable[key], record))

# build null records for left joins
...

return result_records
```

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

Mergejoin

NestedLoopsJoin
(hoeren.MatrNr = pruefen.MatrNr)

Tau
(hoeren.MatrNr)

Tau
(pruefen.MatrNr)

TableScan
(hoeren)

TableScan
(pruefen)

Optimizer If:

- Condition is equality or conjunction of equalities
- Condition columns are sorted

MergeJoin
(hoeren.MatrNr = pruefen.MatrNr)

Tau
(hoeren.MatrNr)

Tau
(pruefen.MatrNr)

TableScan
(hoeren)

TableScan
(pruefen)

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

universität
innsbruck

Hash based Grouping/Aggregation

- Grouping Phase

gamma Rang, Raum aggregate Anzahl as count(Name) professoren;

```
references = get_grouping_references(group_names) #(professoren.Rang, professoren.Raum)
groups = dict()

for record in table.records:
        key = record.values_at_references(references) #(C4, 234)
        groups[key].append(record)

return groups
```

universität
innsbruck

Hash based Grouping/Aggregation

- Aggregate Phase

gamma Rang, Raum aggregate Anzahl as count(Name) professoren;

```python
groups = get_groups()
result = []

for group_key, group in groups.items():
        agg_row = list(group_key)
        for aggregate in aggregates:
                aggregate_col = get_column(group, aggregate.column)
                agg_row.append(aggregate(aggregate.function, aggregate_col)
        result.append(agg_row)

return result
```

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias

Complexity

Nested Loops Join: $\mathcal{O}(n \cdot m)$

m ... number of rows in outer relation

n ... number of rows in inner relation

Merge Join: $\mathcal{O}(n + m)$ (sorting is excluded)

m ... number of rows in outer relation

n ... number of rows in inner relation

Hash Join: $\mathcal{O}(n + m)$

m ... number of rows in outer relation

n ... number of rows in inner relation

Hash based Grouping/Aggregate: $\mathcal{O}(a \cdot n)$

a ... number of aggregates

n ... number of rows

universität
innsbruck

Team 3
Gründlinger Diana, Huber Marcel, Klotz Thomas, Targa Aaron, Thalmann Matthias