universität innsbruck

Architektur und Implementierung von Datenbanksystemen

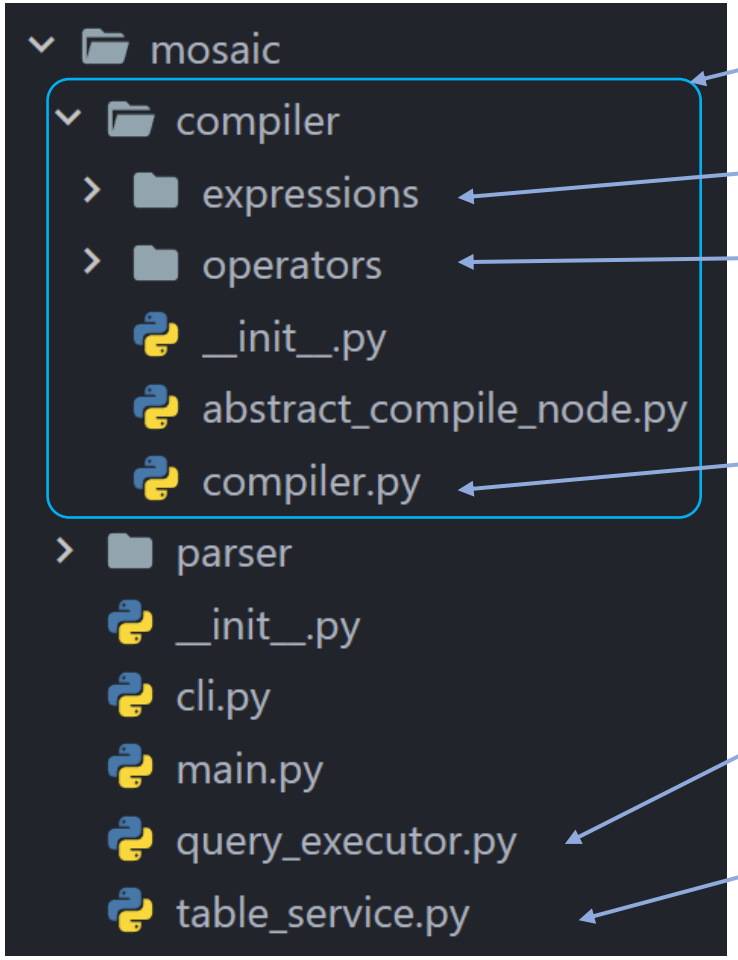
Milestone 2

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

Anforderungen

- Queries kompilieren & ausführen:
  - Projection (pi)
  - Selection ($sigma$)
  - Union / Intersect / Difference ($union$, $intersect$, $except$)
  - Cross Join ($cross\ join$)
  - Sorting (tau)
  - Explain ($explain$)
- Combo-Queries

Projektstruktur



Compiler-Modul: beinhaltet alle Klassen zum Bauen des Execution-Plans anhand eines gegebenen AST‘s

Expressions-Modul: beinhaltet alle Expressions

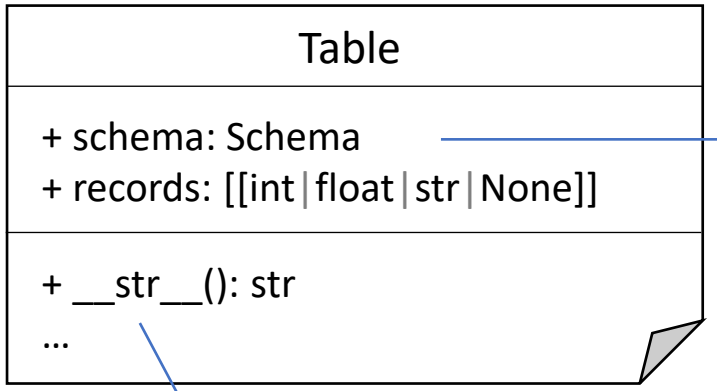
Operators-Modul: beinhaltet alle Operators

Compiler: beinhaltet den Compiler zur Generierung des AST-Visitors → Execution-Plan

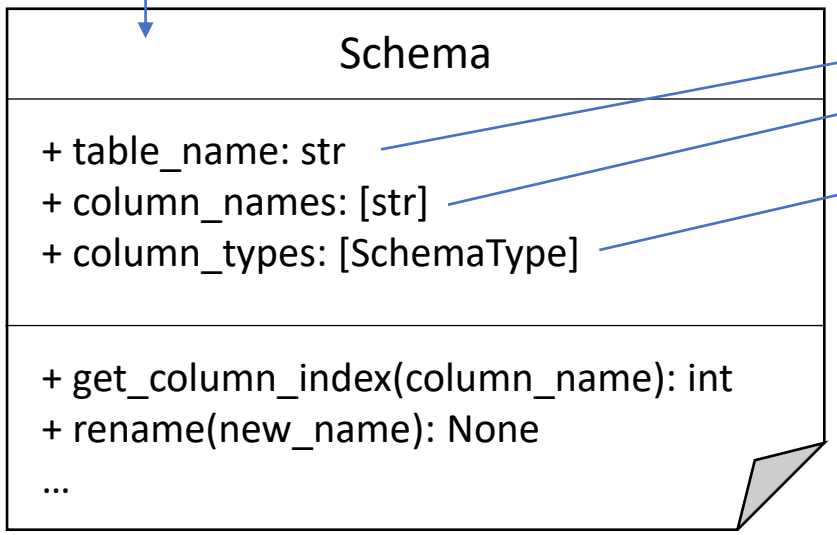
Query-Executor: Bietet Methoden zur Ausführung von Queries, welche durch den Compiler ausgeführt werden

Table-Service: Enthält die Implementierung der Tabellen-Klasse und Methoden um Tabellen zu Laden.

Tabellenrepräsentation



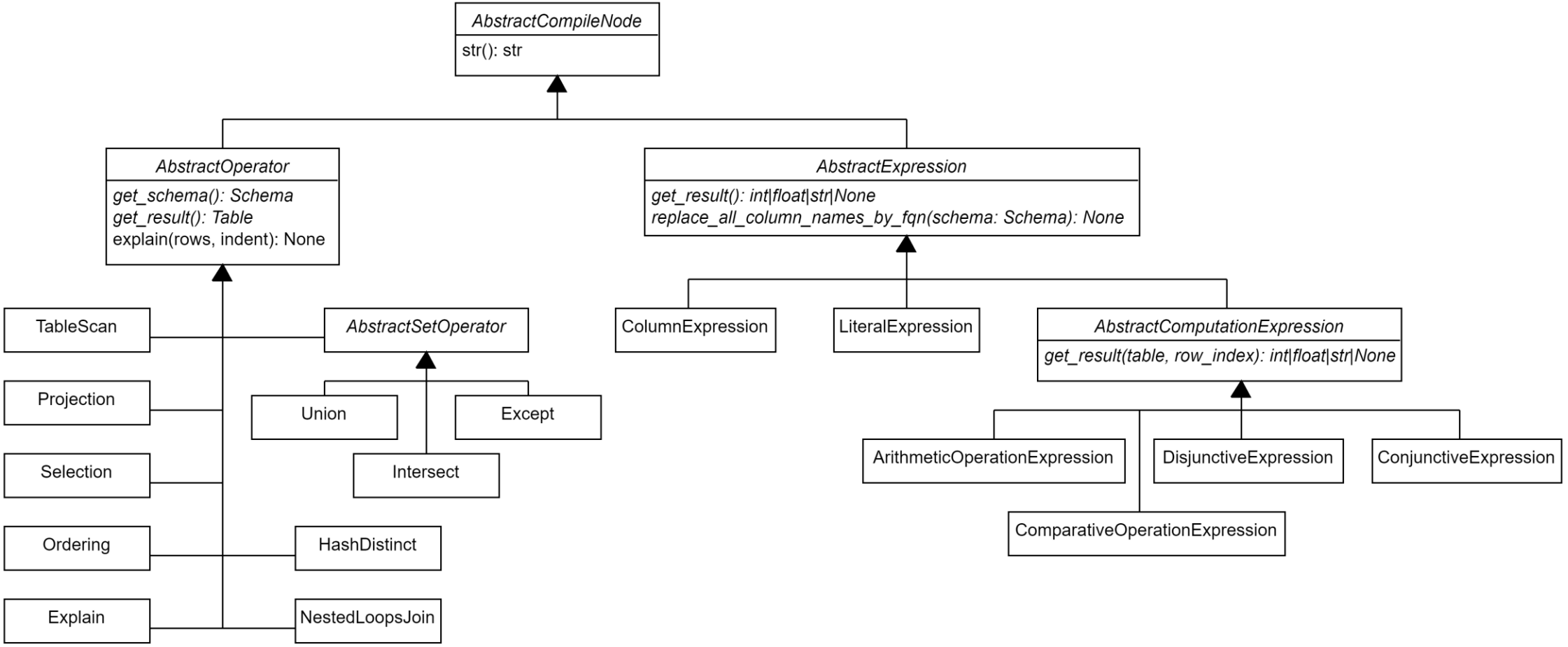
- Spaltennamen werden Fully-Qualified gespeichert
- Aliases werden dementsprechend ohne Tabellen-Name gespeichert



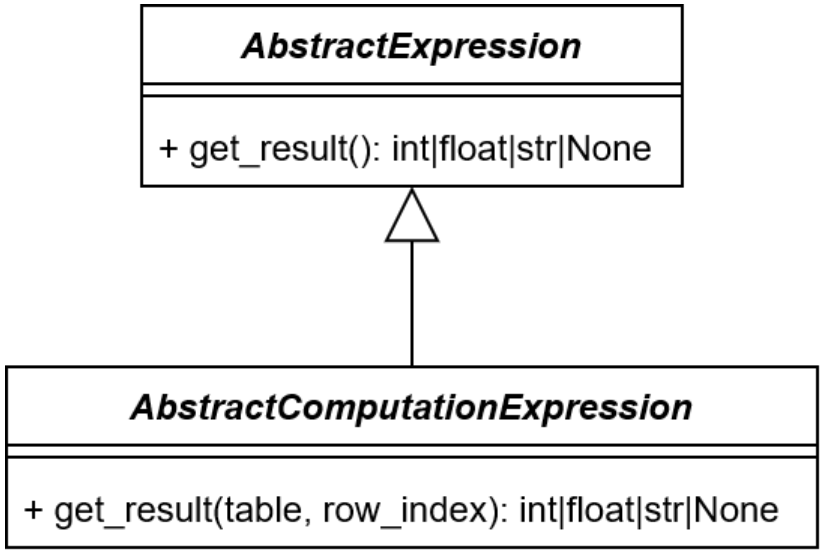

```
+---------------+---------------+
| hoeren.MatrNr | hoeren.VorlNr |
+---------------+---------------+
|         26120 |          5001 |
|         27550 |          5001 |
|         27550 |          4052 |
|         28106 |          5041 |
|         28106 |          5052 |
|         28106 |          5216 |
|         28106 |          5259 |
|         29120 |          5001 |
|         29120 |          5041 |
|         29120 |          5049 |
+---------------+---------------+
```

hoeren	
hoeren.MatrNr (int)	hoeren.VorlNr (int)
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
29120	5215

Expressions & Operators – Klassenhierarchie



Expressions



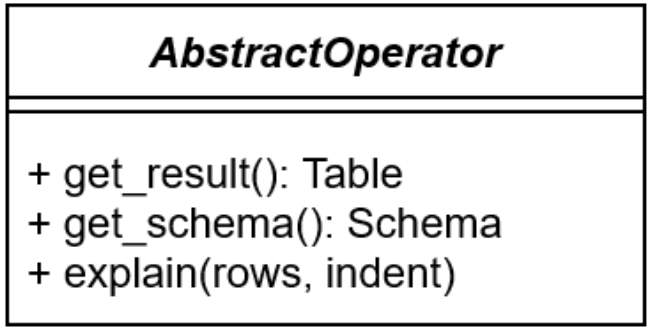
AbstractExpression

- Berechnet Ergebnis für Kinder und gibt es zurück

AbstractComputationExpression

- Berechnet Ergebnis für eine spezifische Zeile einer Tabelle und gibt es zurück

Operators



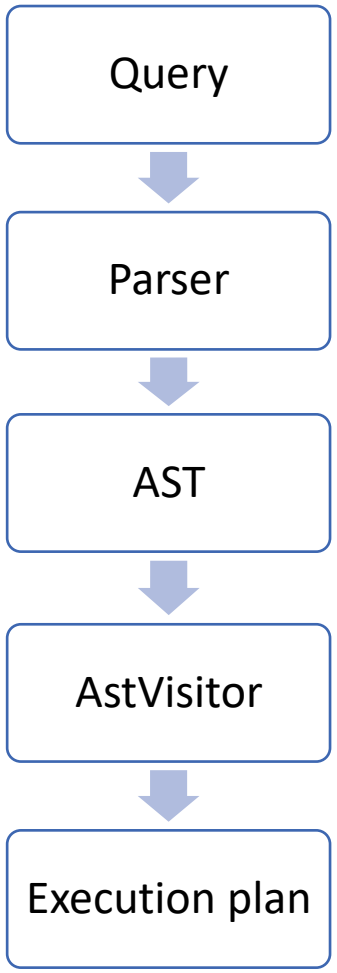
AbstractOperator

- Verwendet die Ergebnisse der Kinder um eine resultierende Tabelle zu bauen

Query Execution

- Der Query-Executor bekommt die auszuführende Query
- Der Parser parsed diese mittels der definierten Grammatik und gibt den AST zurück
- Der Compiler erhält diesen AST und baut einen AST-Visitor
  - Der AST-Visitor wird mithilfe der Operatoren und Expressions gebaut
- Der Query-Executor erhält den ausführbaren Execution-Plan zurück
- Der Query-Executor ruft die **get_result** Methode auf den Plan aus
- Das Ergebnis ist eine Tabelle, welche zurückgegeben wird

- Der Query-Executor misst zudem die Ausführungszeit einer Query und gibt sie zusammen mit dem Ergebnis als Tupel zurück



Explain

- Jede Operator-Klasse enthält eine **explain** Methode, welche beim Aufruf des Explain-Operators rekursiv aufgerufen wird
- Es werden die **__str__** Methoden verwendet um den aktuellen Knoten auszugeben
- Die **explain** Methode ruft rekursiv **explain** auf die Kinder auf
- Um die Fully-Qualified-Name im explain zu erhalten, wird die Methode **replace_all_column_names_by_fqn** rekursiv aufgerufen, um alle vorkommenden ColumnExpressions durch ihre Fully-Qualified-Namen auszutauschen

```
def explain(self, rows, indent):
    rows.append([indent * "-" + ">" + self.__str__()])
```

```
def explain(self, rows, indent):
    super().explain(rows, indent)
    self.table_reference.explain(rows, indent + 2)
```

www.uibk.ac.at