universität innsbruck

Architektur und Implementierung von Datenbanksystemen

Task 5

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

Expression representation

Abstract Expression

- For each expression in the AST a class is created
- Abstract expression is super class for all expressions
- Expression classes for binary expressions have right, left
- Operator expressions have unique attributes like conditions for join and selection and columns for projection

```python
class AbstractExpression(ABC):
    def __init__(self):
        pass

    @abstractmethod
    def get_result(self):
        pass

    @abstractmethod
    def __str__(self):
        pass

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

Expression representation

Expression classes

- Arithmetic Operation Expression
- Column Expression
- Comparative Expression
- Conjunctive Expression
- Disjunctive Expression
- Explain
- Hash Distinct
- Literal Expression
- Nested Loops Join
- Ordering Expression
- Projection
- Selection
- Set Expression
- Table Scan

Plan operators interface

Operators implementation

- There is no explicit plan operators interface
- The operators inherit from the abstract expression class
- The `get_schema()` method gets implemented in the `get_result()` methods of the operators
- All operator classes return a new table object with the corresponding relational algebra operation applied

Plan operators interface

Join operator example

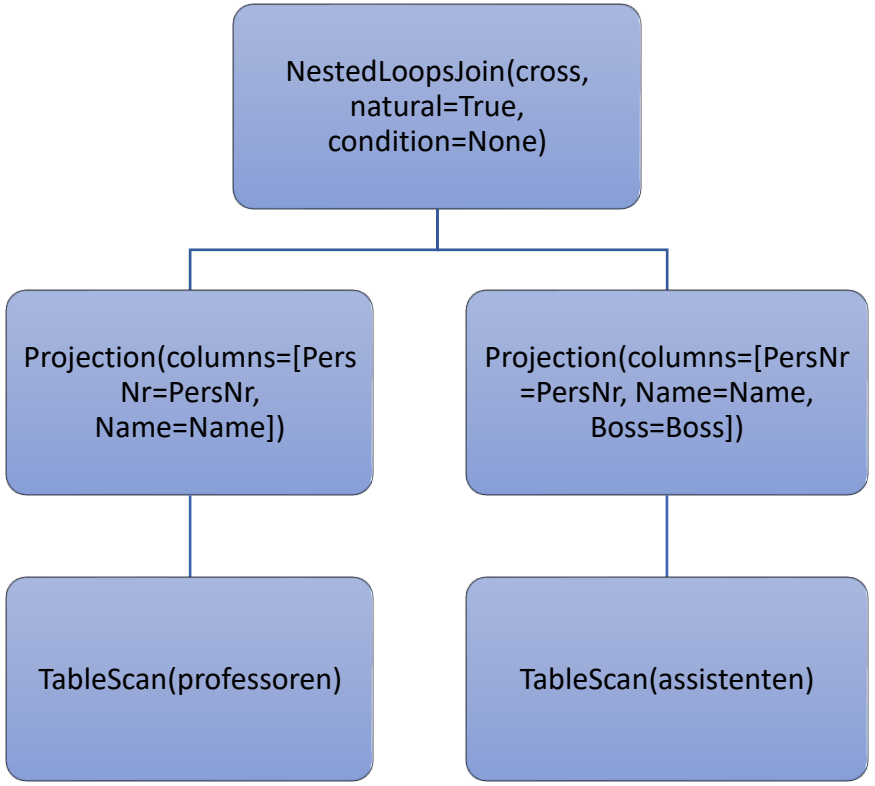
```python
def get_result(self):
    table1 = self.table1_reference.get_result()
    table2 = self.table2_reference.get_result()
    if self.join_type == JoinType.CROSS:
        joined_table_records = []
        for record1 in table1.records:
            for record2 in table2.records:
                joined_table_records.append(record1 + record2)
        joined_table_name = f"{table1.table_name}_cross_join_{table2.table_name}"

        return Table(joined_table_name, table1.schema_names + table2.schema_names,
                     table1.schema_types + table2.schema_types, joined_table_records)
    return None
```

Plan execution

- Materialization is used to execute the plan
- We don‘t use a initialize method
- Only the `get_result()` method from the root of the execution plan gets called recursively and returns the resulting table

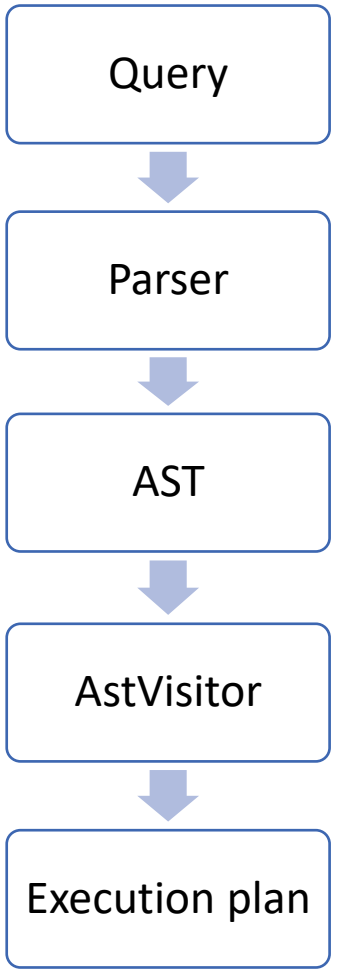
explain pi PersNr, Name professoren cross join pi PersNr, Name, Boss assistenten;



Execution plan mapping

- Query gets parsed by the parser with the defined grammar
- Parser returns AST
- AstVisitor visits nodes from the AST bottom up and builds execution plan

When using the explain query instead of `get_result()` the `explain()` method gets called recursively and builds a visual representation of the execution plan.



www.uibk.ac.at