QUERY PROCESSING

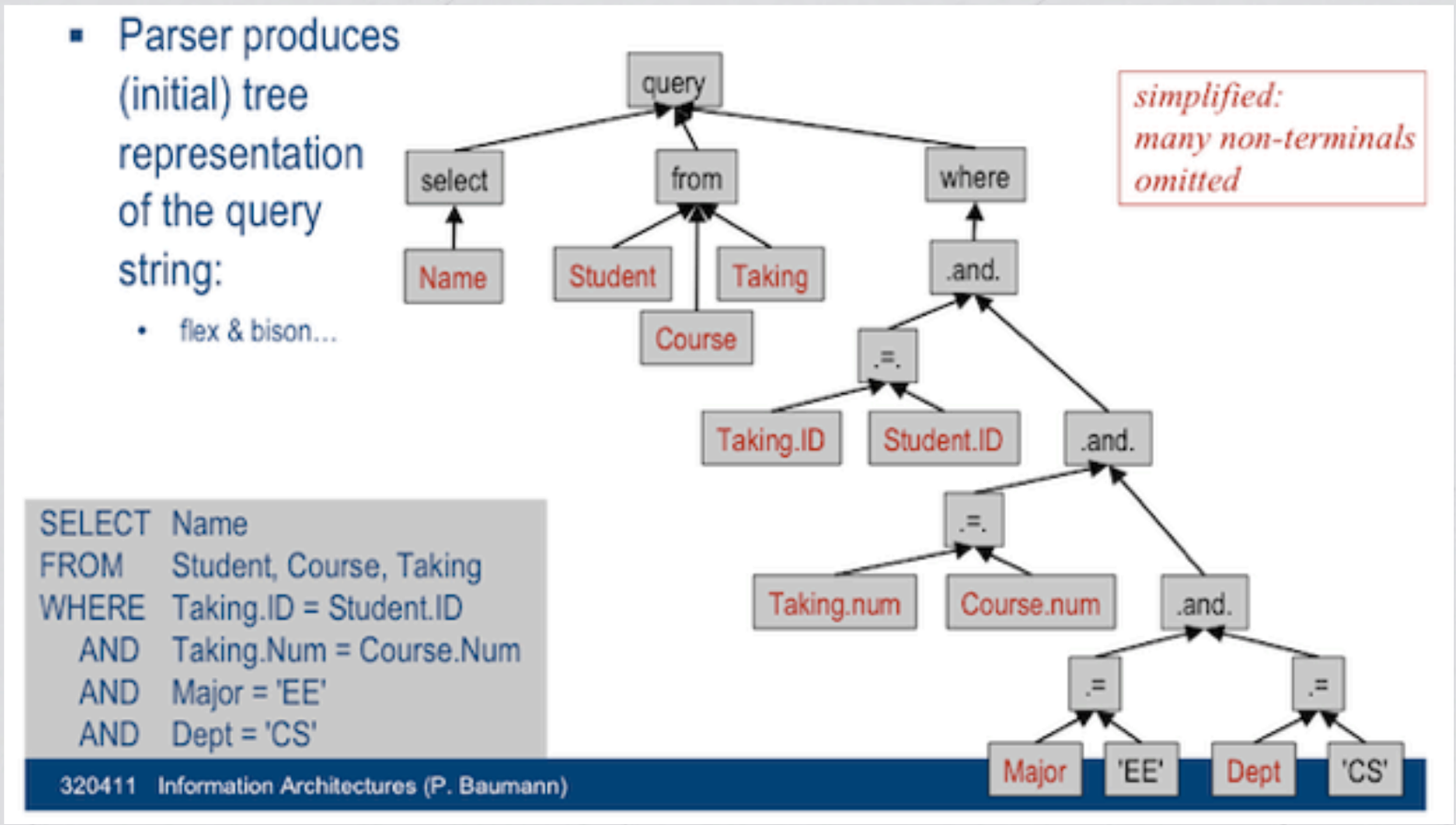
Vaibhav Bajpai

Contents

Overview

* What is Query Processing?
* What is Query Optimization?
* What is Query Compilation?
* What is Query Execution?

Parser



* Produces a Tree Representation of the Query String

* Using *flex* and *bison*

Checker

* Validates the Parse Tree from System Catalog
* All tables in FROM clause exist
* All columns of tables exist
* All comparisons | aggregations are type-compatible.

View Expander

- Suppose Student is actually a view:
  - StudName(ID, Name)
  - StudMajor(ID, Major)

```
CREATE VIEW Student AS
    SELECT StudName.ID, StudName.Name, StudMajor.Major
    FROM   StudName, StudMajor
    WHERE  StudName.ID = StudMajor.ID
```

- Via the view expander, original query becomes:

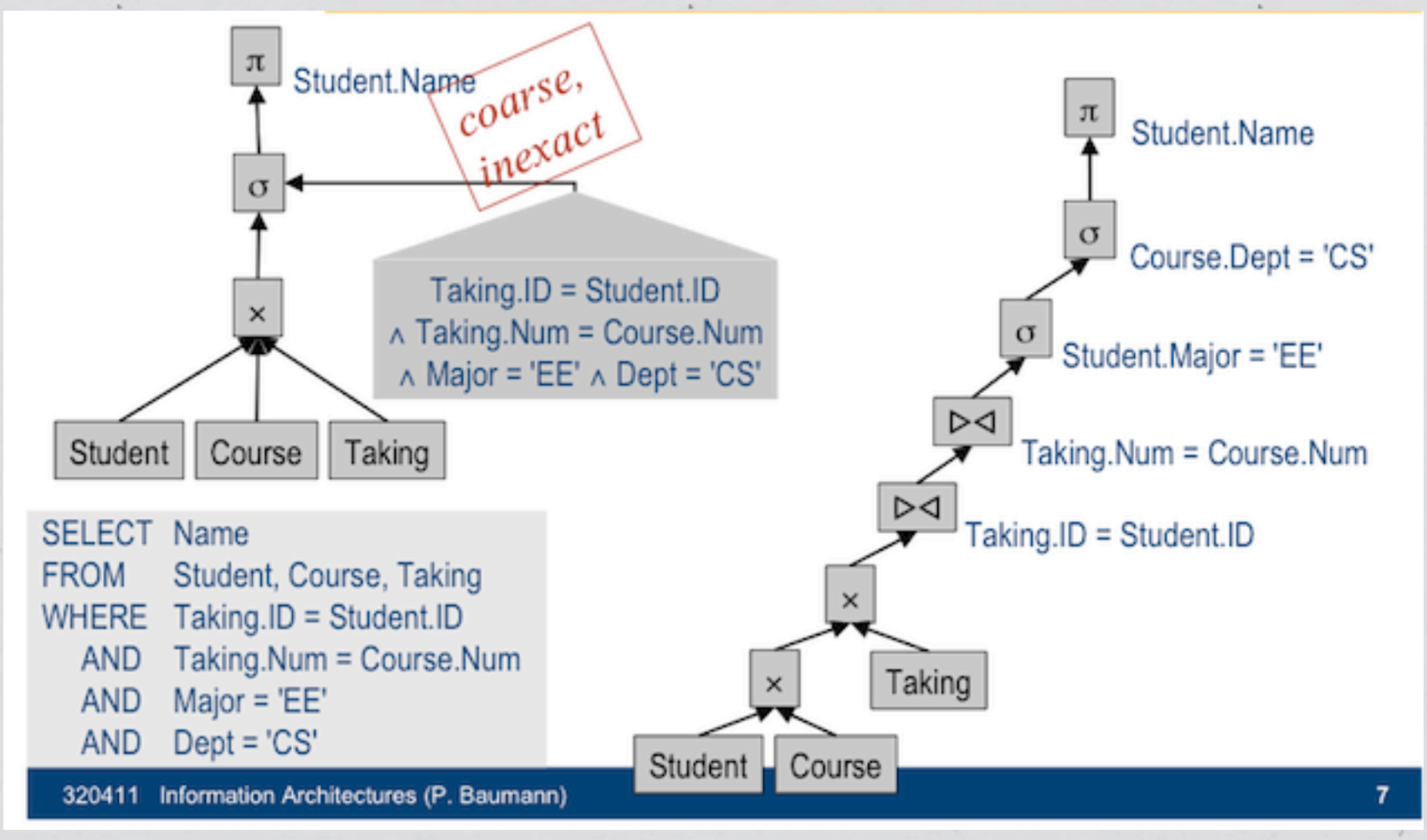
```
SELECT Name
FROM   Course, Taking, Student AS ( SELECT StudName.ID, StudName.Name,
StudMajor.Major FROM StudName, StudMajor WHERE  StudName.ID = StudMajor.ID )
WHERE  Taking.ID = Student.ID AND Taking.Num = Course.Num AND
Student.Major = 'EE' AND Course.Dept = 'CS' AND Student.ID = StudMajor.ID
```

- Or "flattened":

```
SELECT Name
FROM   Course, Taking, StudName, StudMajor
WHERE  Taking.ID = StudName.ID AND Taking.Num = Course.Num AND
StudMajor.Major = 'EE' AND Course.Dept = 'CS' AND StudName.ID = StudMajor.ID
```

*

Logical Query Plan Generator



* Leaves of Logical Plan are Table Names.
* Uses Extended Relational Algebra Operators.

Logical Query Plan Generator

* Basic Operators: Select | Project | Cross-Product | Union | Difference

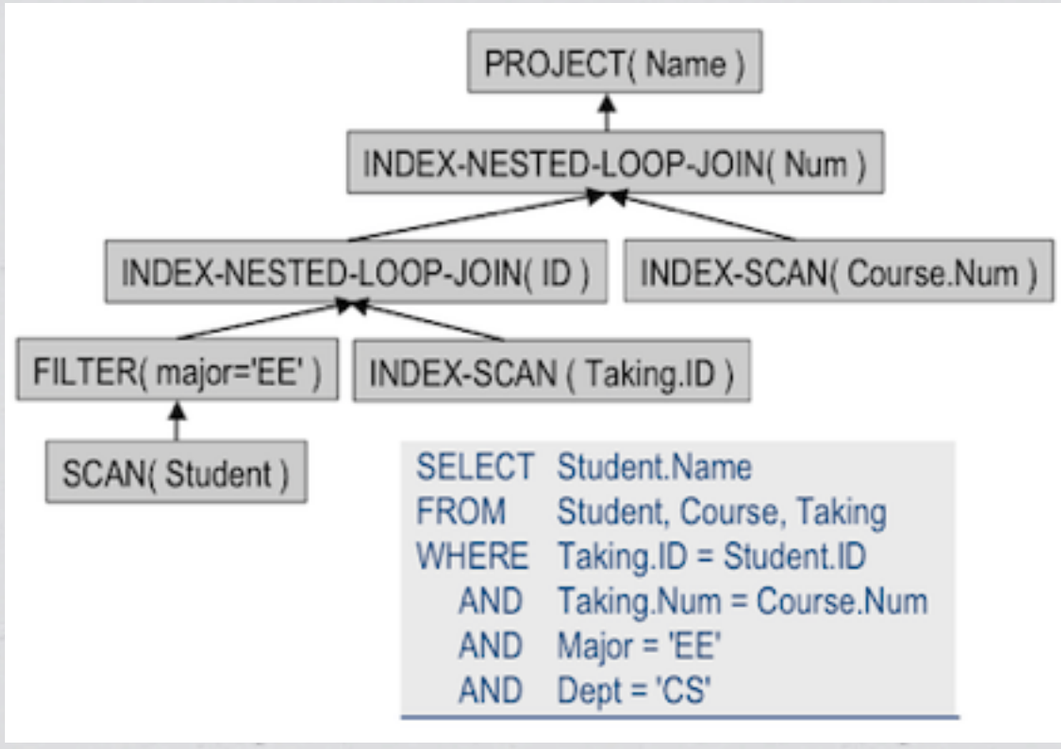
* Abbreviations: Natural-Join | Theta-Join | Intersect

* Extensions: Rename | Aggregate-GroupBy | Distinct

Query Rewriter

* Rewrites the Logical Query to produce better Physical Plan
* Rewrites are based on heuristic rules (not on DB statistics)
* Generate alternatives and send some/all to Physical Plan Generator
* Rewrites are NOT guaranteed to be improvements

Physical Plan Generator



* Leaf Nodes are Tables or Indexes.
* Lots of possible Physical Query Plans over a Logical Query plan.
* *PQ Enumerator* tries to select the optimal one
  * using response time and throughput parameters

Physical Plan Generator

* Access Methods: Scan | Index-Scan | Index-Scan-P
* Join Methods: Nested-Loop-Join | Sort-Merge-Join | Hash-Join
* Operators: Sort | Group | Aggregate | Union | Except | Intersect
* Distributed Systems: Ship
* Parallel Systems: Split | Merge | Exchange

Code Generator

* Translates Physical Query Plan into Executable Code
* Very System-Specific
* May instead use a Query-Plan-Interpretor

Code Generator

* Temporary Table Approach
* Iterator Approach

Temporary Table Approach

* Children write intermediate results to temporary table
* Parents read the temporary table
* [+] conceptually easy
* [-] harder to implement
* [-] less efficient

Iterator Approach

* open(): setup the iterator
* getNext(): get the next record
* close(): destroy the iterator
* inner nodes call the iterator for children.
* root iterator produces the query result.
  * blocking operators: getNext() exhaustive call on children.
  * non-blocking operators: getNext() non-exhaustive call on children.