Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Databases

Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

January 8, 2026

Version from January 7, 2026

Topics for Today

- Execution plan generation
- Assessment of execution plans

Acknowledgement

- Some of the upcoming content is taken from the following lecture (and corresponding exercises) and has been translated and partially modified:
  - PD Dr. habil. Özgür L. Özçep: »Datenbanken«

Query Optimization – Comprehension Questions

Are the following statements correct?

- »The number of possible join execution plans increases polynomially with an increase in the number of relations.«

- »When rewriting a query, WHERE clauses are never extended.«

Query Optimization – Comprehension Questions

Are the following statements correct?

- »The number of possible join execution plans increases polynomially with an increase in the number of relations.«
  - Incorrect. The number of possible join execution plans increases *exponentially* with an increase in the number of relations.
- »When rewriting a query, WHERE clauses are never extended.«

Query Optimization – Comprehension Questions

Are the following statements correct?

- »The number of possible join execution plans increases polynomially with an increase in the number of relations.«
  - Incorrect. The number of possible join execution plans increases *exponentially* with an increase in the number of relations.
- »When rewriting a query, WHERE clauses are never extended.«
  - Incorrect. During rewriting, the WHERE clause can be extended by adding additional join conditions. This allows additional join plans to be included in the optimization.

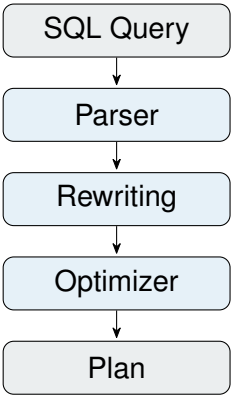
Generation of an Execution Plan for an SQL Query

What are the individual steps to generate an execution plan for an SQL query?

Generation of an Execution Plan for an SQL Query

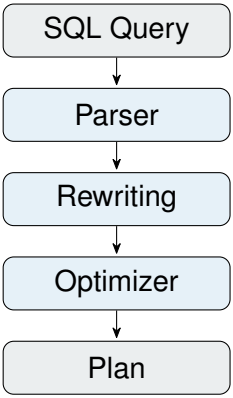
What are the individual steps to generate an execution plan for an SQL query?

- Parser: Syntactic and semantic analysis of the query
- Rewriting: Data-independent optimization
  - Adjust the order of operations
  - Add additional join predicates
- Optimizer: Data-dependent optimization
  - Consider various plans based on the data and estimate the costs



Generation of an Execution Plan for an SQL Query

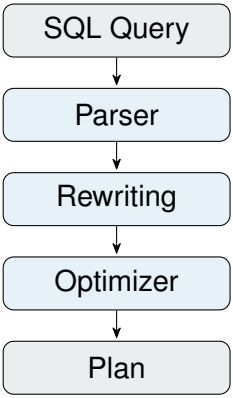
Is it possible to swap the order of the rewriter and optimizer?



Generation of an Execution Plan for an SQL Query

Is it possible to swap the order of the rewriter and optimizer?

- The rewriter works independently of the data
  - Prepares most efficient processing of a general query with corresponding operations
- Due to preparation of the rewriter, the optimizer works with the optimal structure of operations
- Placing the rewriter after the optimizer would mean that potential plans obtained by extending WHERE clauses are not considered during optimization



Assessment of Execution Plans

Consider the following schema of a database:

- Student(StudentId, FirstName, LastName)
- Course(CourseId, Title)
- Enrolled(CourseId, StudentId)

Further, assume the following cardinalities:

$$|Student| = 900, \quad |Course| = 50, \quad |Enrolled| = 4500$$

Formulate a query in SQL to obtain the first names of all students that are enrolled in the course »Databases«.

Assessment of Execution Plans

Formulate a query in SQL to obtain the first names of all students that are enrolled in the course »Databases«.

```sql
SELECT FirstName
FROM Student S, Course C, Enrolled E
WHERE C.CourseId = E.CourseId
  AND S.StudentId = E.StudentId
  AND C.Title = 'Databases';
```

Assessment of Execution Plans

```
1 SELECT FirstName
2 FROM Student S, Course C,
     Enrolled E
3 WHERE C.CourseId = E.CourseId
4   AND S.StudentId = E.StudentId
5   AND C.Title = 'Databases';
```

Consider the following schema of a database:

- Student(StudentId, FirstName, LastName)
- Course(CourseId, Title)
- Enrolled(CourseId, StudentId)

Further, assume the following cardinalities:

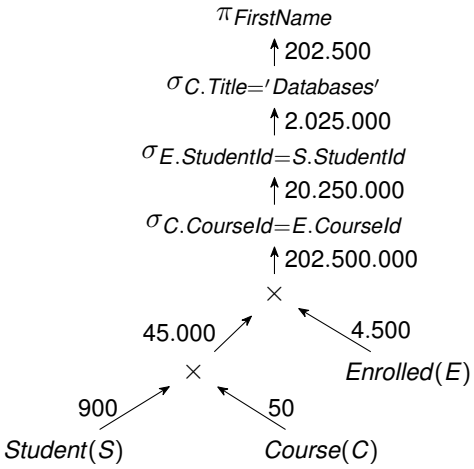
$$|Student| = 900, \quad |Course| = 50, \quad |Enrolled| = 4500$$

Create a naive query plan from the SQL query that uses only the Cartesian product (no joins). Estimate the number of intermediate results.

Assessment of Execution Plans

Estimation without additional information about indices:

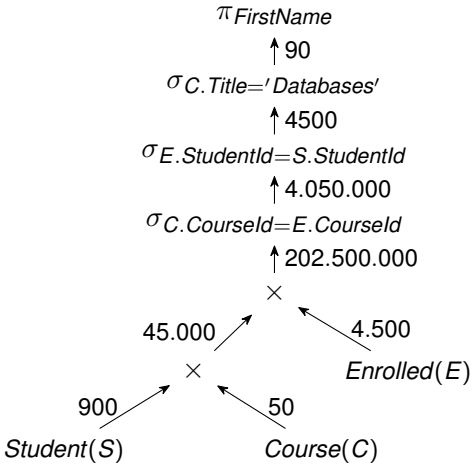
- Each selection reduces the size by a factor of 10



Assessment of Execution Plans

Estimation with additional information about indices:

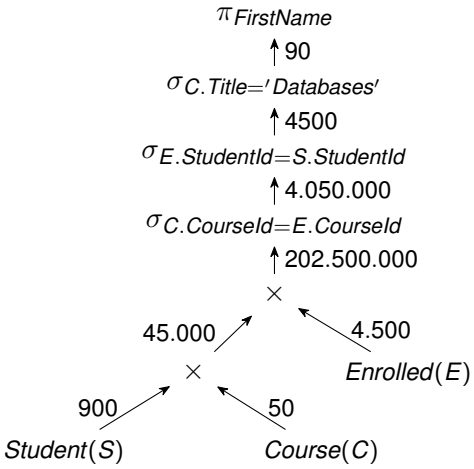
- $\sigma_{C.CourseId=E.CourseId}$
  - All course IDs are compared with enrollments in the respective course. This effectively extends E by a column $C.Title$, whereby there can be up to 900 duplicates of a row with a specific course ID, as there are 900 students, resulting in $900 \cdot 4500 = 4.050.000$ entries



Assessment of Execution Plans

Estimation with additional information about indices:

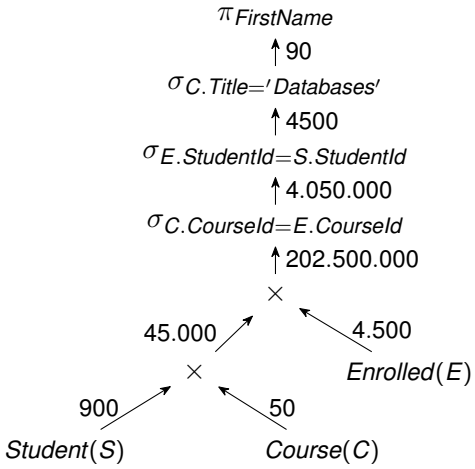
- $\sigma_{E.StudentId=S.StudentId}$
  - Not every entry from the join of E and C is combined with all 900 entries from S anymore, but only with the one entry with a matching key, giving a reduction in size of factor 900:
    $4.050.000 \,/\, 900 = 4.500$



Assessment of Execution Plans

Estimation with additional information about indices:

- $\sigma_{C.Title='Databases'}$
  - Assuming that only one of the 50 courses has the title »Databases« (which requires semantic knowledge and thus, assuming factor 10 instead is also fine), the size is reduced by a factor of 50: $4.500 / 50 = 90$



Assessment of Execution Plans

```
1 SELECT FirstName
2 FROM Student S, Course C,
     Enrolled E
3 WHERE C.CourseId = E.CourseId
4   AND S.StudentId = E.StudentId
5   AND C.Title = 'Databases';
```

Consider the following schema of a database:

- Student(<u>StudentId</u>, FirstName, LastName)
- Course(<u>CourseId</u>, Title)
- Enrolled(<u>CourseId</u>, <u>StudentId</u>)

Further, assume the following cardinalities:

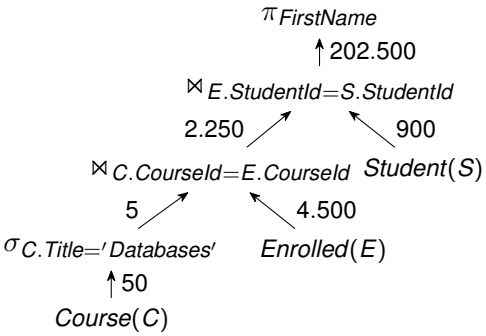
$$|Student| = 900, \quad |Course| = 50, \quad |Enrolled| = 4500$$

Optimise the previous query plan without the restrictions given there, so that the intermediate results are kept small. Estimate the number of intermediate results.

Assessment of Execution Plans

Estimation without additional information about indices:

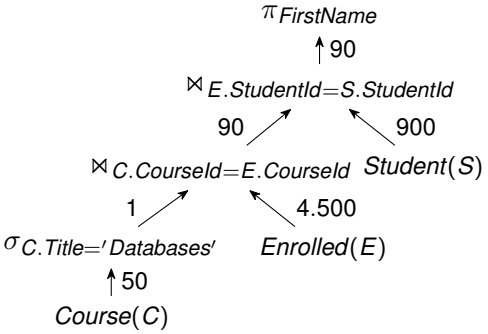
- Each selection reduces the size by a factor of 10
- Each join reduces the size by a factor of 10
  - $\bowtie_{C.CourseId=E.CourseId}$
    - $5 \cdot 4.500 \,/\, 10 = 2.250$
  - $\bowtie_{E.StudentId=S.StudentId}$
    - $2.250 \cdot 900 \,/\, 10 = 202.500$



Assessment of Execution Plans

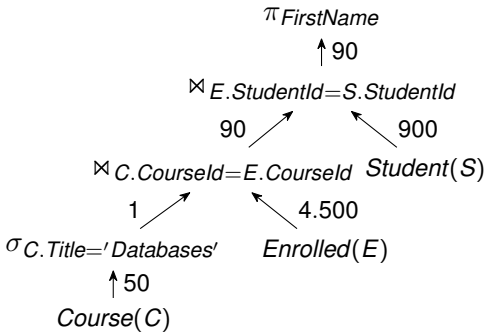
Estimation with additional information about indices:

- $\sigma_{C.Title='Databases'}$
  - Assumption: There is just one course with the title »Databases«
- $\bowtie_{C.CourseId=E.CourseId}$
  - Average number of students enrolled per course: $\frac{|E|}{|C|} = \frac{4500}{50} = 90$
- $\bowtie_{E.StudentId=S.StudentId}$
  - Leaves size unchanged as we only need data from the 90 students that take part in »Databases«



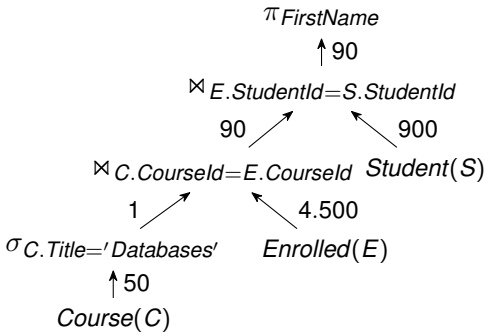
Assessment of Execution Plans

Formulate an expression in relational algebra that corresponds to the previous query tree.



Assessment of Execution Plans

Formulate an expression in relational algebra that corresponds to the previous query tree.



$$\pi_{\text{FirstName}}\Big(((\sigma_{\text{C.Title='Databases'}}\, C) \bowtie_{\text{C.CourseId=E.CourseId}} E) \bowtie_{\text{E.StudentId=S.StudentId}} S\Big)$$