CS 564: Database Management Systems

Lecture 28: Query Optimization II

Xiangyao Yu

4/3/2024

Module B3 Query Processing

Relational operators I

Relational operators II

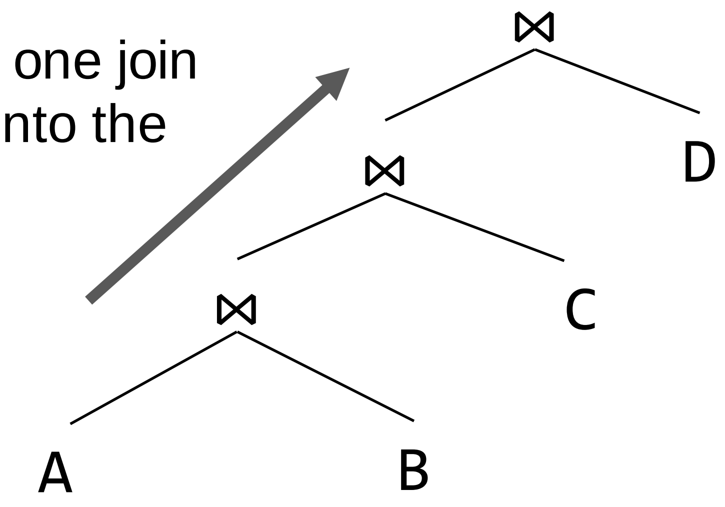
Query optimization I

Query optimization II

Column Store

Clarification on Pipelined Hash Join

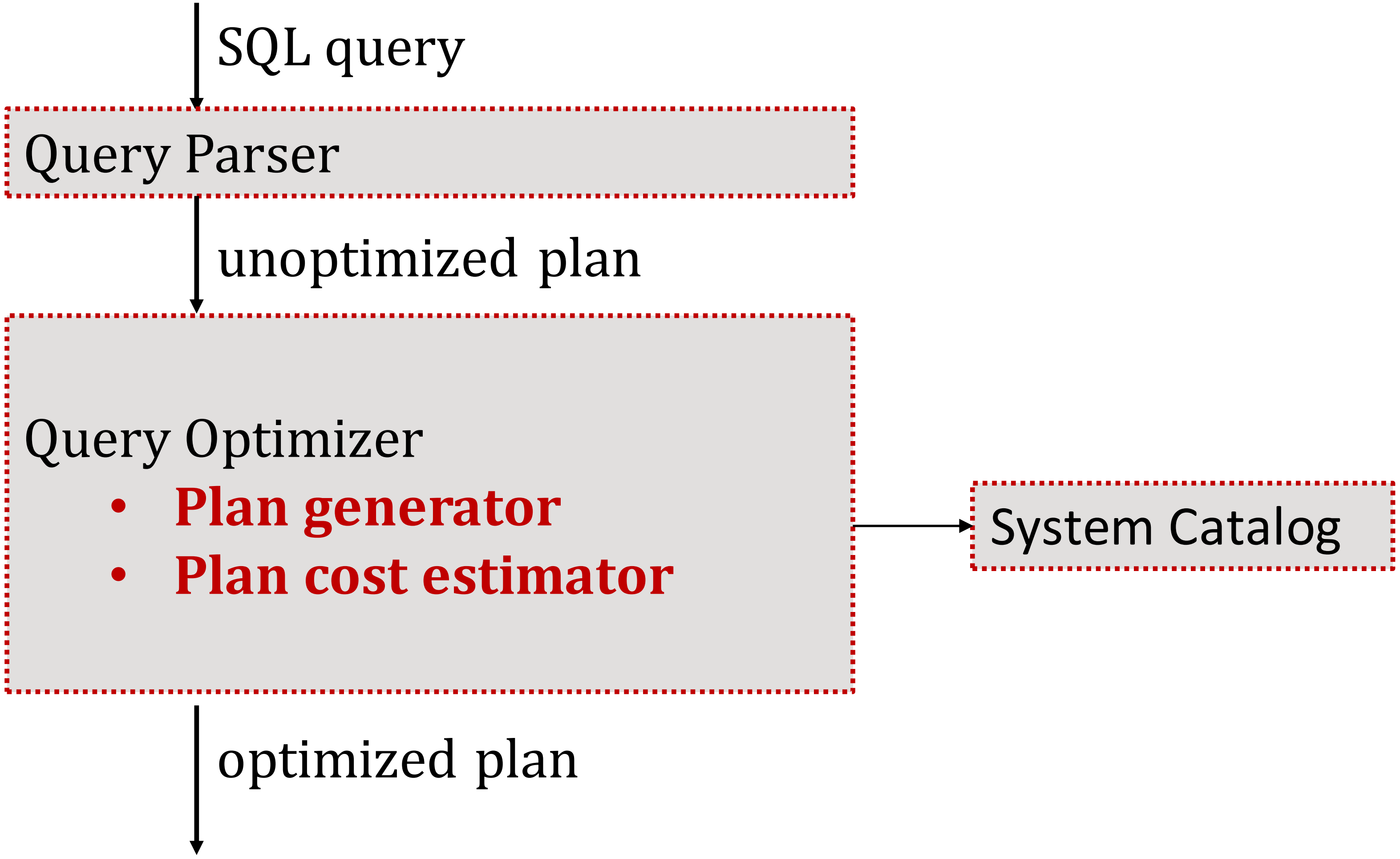
Results of one join pipelined into the next join



Step 1: Build hash tables on relations B, C, and D

Step 2: Read table A tuple by tuple and lookup hash tables in order

Architecture of an Optimizer



Outline

Aggregation pushdown

System catalog

Cost estimation

- Cardinality estimation
- IO cost estimation

Plan enumeration

Group-By Aggregation Pushdown

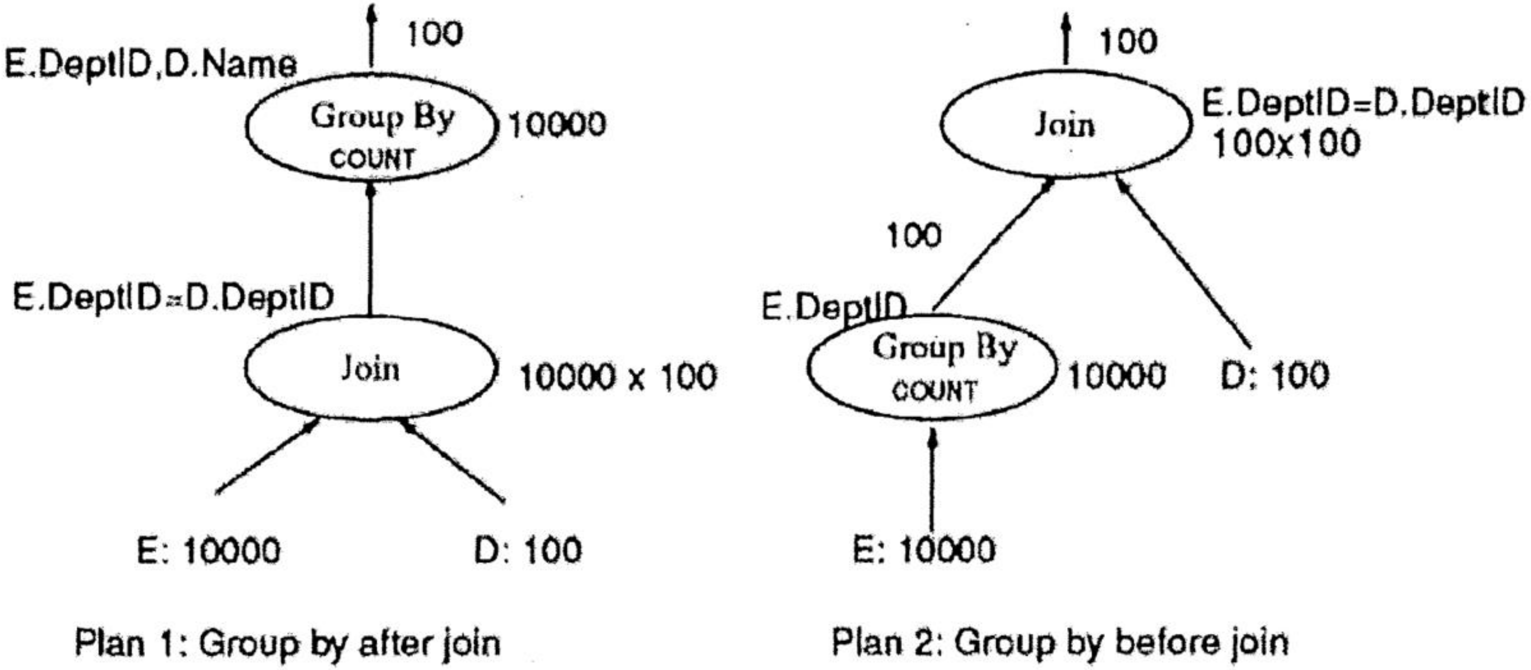
Partial group by can also reduce cost

Example:

SELECT D.name, count(*)
FROM EMP as E, DEPT as D
WHERE E.DeptID = D.DeptID
GROUP BY D.name

E has 10000 tuples

D has 100 tuples



Outline

- Aggregation pushdown
- **System catalog**
- Cost estimation
  - Cardinality estimation
  - IO cost estimation
- Plan enumeration

System Catalog

Catalog tables: Special tables that store the descriptive information of every table and index

Example:

attr_name	rel_name	type	position
sid	Sailors	integer	1
sname	Sailors	string	2
rating	Sailors	integer	3
age	Sailors	real	4
sid	Reserves	integer	1
bid	Reserves	integer	2
day	Reserves	dates	3
rname	Reserves	string	4

Information in the Catalog

For each table

- Table name, file name, file structure (e.g., heap file)
- Attribute name and type of each attribute
- Index name
- Integrity constraints (e.g., primary key and foreign key)

For each index

- Index name and the structure of the index
- Search key attributes

For each view

- View name and definition

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Table size: The number of pages ***NPages(R)*** for each table *R*

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Table size: The number of pages ***NPages(R)*** for each table *R*

Index cardinality: The number of distinct key values ***NKeys(I)*** for each index *I*

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Table size: The number of pages ***NPages(R)*** for each table *R*

Index cardinality: The number of distinct key values ***NKeys(I)*** for each index *I*

Index size: The number of pages ***INPages(I)*** for each index *I* (for a B+ tree index, we take *INPages* to be the number of leaf pages)

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Table size: The number of pages ***NPages(R)*** for each table *R*

Index cardinality: The number of distinct key values ***NKeys(I)*** for each index *I*

Index size: The number of pages ***INPages(I)*** for each index *I* (for a B+ tree index, we take *INPages* to be the number of leaf pages)

Index height: The number of nonleaf levels ***IHeight(I)*** for each index *I*

Statistics About Tables and Indexes

Table cardinality: The number of tuples ***NTuples(R)*** for each table *R*

Table size: The number of pages ***NPages(R)*** for each table *R*

Index cardinality: The number of distinct key values ***NKeys(I)*** for each index *I*

Index size: The number of pages ***INPages(I)*** for each index *I* (for a B+ tree index, we take *INPages* to be the number of leaf pages)

Index height: The number of nonleaf levels ***IHeight(I)*** for each index *I*

Index range: The minimum present key value ***ILow(I)*** and the maximum present key value ***IHigh(I)*** for each index *I*

Outline

Aggregation pushdown

System catalog

Cost estimation

- Cardinality estimation
- IO cost estimation

Plan enumeration

Cost Estimation

Estimating the cost of a query plan involves:

1. Estimating the cost of each operation in the plan
   - Depends on input cardinalities
   - Algorithm cost
2. Estimating the size of intermediate results (cardinality estimation)
   - We need statistics about input relations
   - For selections and joins, we typically assume independence of predicates

Outline

- Aggregation pushdown
- System catalog
- Cost estimation
  - **Cardinality estimation**
  - IO cost estimation
- Plan enumeration

Cardinality Estimation

Cardinality [Mathematical]: The number of values in a set

Cardinality [Database]: the number of distinct values in a table column

Cardinality estimation: Predict the number of rows that a (sub-)query returns

– Plays a crucial rule in query optimization!

Estimating Result Size

Consider the following query block

```
SELECT attribute list
FROM   relation list
WHERE  term₁
  AND  term₂
  ...
  AND  termₙ
```

We associate each term with a reduction factor: ratio of the expected result size to the input size

Estimated output size = maximum size times the product of the reduction factors for the terms in the WHERE clause

Reduction Factors

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)

Assume uniform distribution

If we have more statistics (i.e., number of distinct values), can do better than 1/10

Reduction Factors

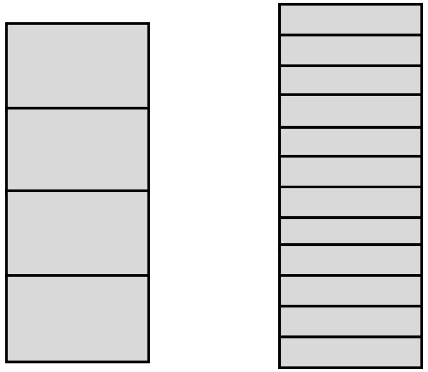
Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$

Assumes each key in the index with smaller cardinality has a matching value in the other index

Why Max(Nkeys1, Nkeys2)

Format of terms	**Reduction factor**
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$

Assumes each key in the index with smaller cardinality has a matching value in the other index



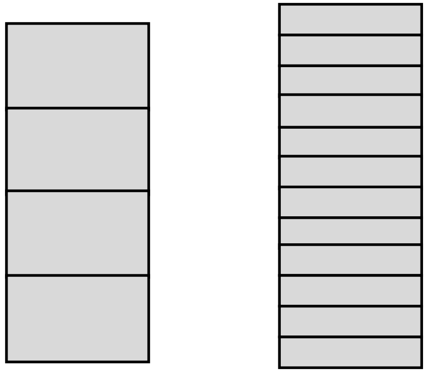
ICARD1 < ICARD2

For each record in relation 1, (NTuples2 / NKeys2) tuples in relation 2 will satisfy the predicate

Why Max(Nkeys1, Nkeys2)

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$

Assumes each key in the index with smaller cardinality has a matching value in the other index



ICARD1 < ICARD2

For each record in relation 1, (NTuples2 / NKeys2) tuples in relation 2 will satisfy the predicate

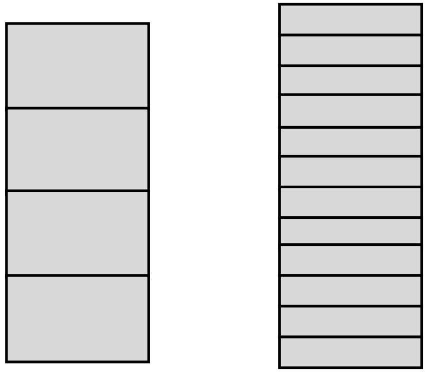
Total number of selected tuples =

NTuples1 * NTuples2 / NKeys2

Why Max(Nkeys1, Nkeys2)

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1), NKeys(I2))}$

Assumes each key in the index with smaller cardinality has a matching value in the other index



ICARD1 < ICARD2

For each record in relation 1, (NTuples2 / NKeys2) tuples in relation 2 will satisfy the predicate

Total number of selected tuples =

NTuples1 * NTuples2 / NKeys2

Reduction factor **F = 1 / NKeys2**

Reduction Factors

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$ If have one index (e.g., *I1*): $\frac{1}{NKeys(I1)}$ Otherwise: 1/10

Reduction Factors

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$ If have one index (e.g., *I1*): $\frac{1}{NKeys(I1)}$ Otherwise: 1/10
column > value	If have index *I*: $\frac{High(I) - value}{High(I) - Low(I)}$ Otherwise: arbitrary value (e.g., 1/3)

Reduction Factors

Format of terms	Reduction factor
column = value	If have index *I*: $\frac{1}{NKeys(I)}$ Otherwise: 1/10 (arbitrary guess)
column1 = column2	If have index *I1* and *I2:* $\frac{1}{MAX(NKeys(I1),NKeys(I2))}$ If have one index (e.g., *I1*): $\frac{1}{NKeys(I1)}$ Otherwise: 1/10
column > value	If have index *I*: $\frac{High(I) - value}{High(I) - Low(I)}$ Otherwise: arbitrary value (e.g., 1/3)
column in (list of values)	(Reduction factor of “column = value”) × (number of items in list)

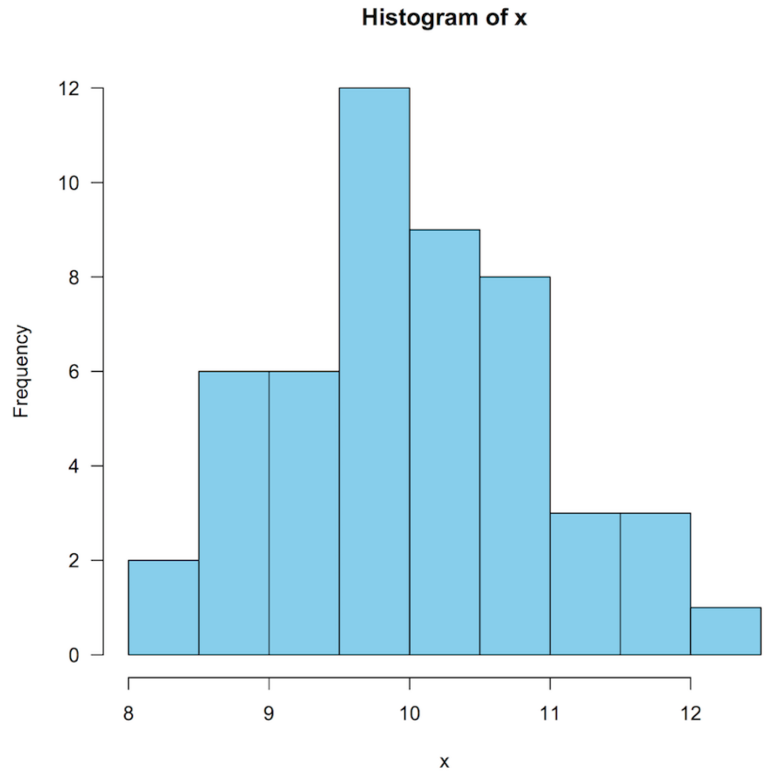
Improved Statistics – Histogram

Consider column > value, We can estimate reduction factor by $\frac{High(I) - value}{High(I) - Low(I)}$, which assumes uniform distribution

Improved Statistics – Histogram

Consider column > value, We can estimate reduction factor by $\frac{High(I) - value}{High(I) - Low(I)}$, which assumes uniform distribution

We can approximate the distribution more accurately with a histogram



Outline

- Aggregation pushdown
- System catalog
- Cost estimation
  - Cardinality estimation
  - **IO cost estimation**
- Plan enumeration

IO cost

Calculate the number of pages access through IO

Table scan

– IO = NPages(R)

IO cost

Calculate the number of pages access through IO

Table scan

- IO = NPages(R)

Unique index matching (e.g., EMP.ID = ‘123’)

- IO = 1 data page + 1-3 index page

IO cost

Calculate the number of pages access through IO

Table scan

- IO = NPages(R)

Unique index matching (e.g., EMP.ID = '123')

- IO = 1 data page + 1-3 index page

Clustered index matching

- IO = F(preds) * (INPages(I) + NPages (R))   # index pages & # data pages

IO cost

Calculate the number of pages access through IO

Table scan

- IO = NPages(R)

Unique index matching (e.g., EMP.ID = ‘123’)

- IO = 1 data page + 1-3 index page

Clustered index matching

- IO = F(preds) * (INPages(I) + NPages (R))   # index pages & # data pages

Non-clustered index matching

- IO = F(preds) * (INPages(I) + NTuples(R))   # index pages & # data page accesses

Outline

- Aggregation pushdown
- System catalog
- Cost estimation
  - Cardinality estimation
  - IO cost estimation
- **Query planning**

Multi-Relation Query Planning

Option 1: Bottom-up optimization

- Estimate cost for individual operators (e.g., joins) and then use dynamic programming to build up the query plan

Option 2: Top-down optimization

- Start with a logical plan; search the plan tree and convert logical operators into physical operators

Nested-Query Decorrelation

Correlated Nested query:

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
              FROM Reserves R
              WHERE R.bid = 103
                AND R.sid = S.sid);
```

correlated subquery

Equivalent to:

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 103;
```

Summary

Aggregation pushdown

System catalog

Cost estimation

- Cardinality estimation
- IO cost estimation

Query planning