

数据库查询基本流程

数据库查询

- 可以看做数据集合上做运算，运算的基本单位是算子
 - 投影
 - 扫描(理论上存储提供)
 - 选择(事实很多时候存储提供)
 - JOIN
 - 聚集
 - 排序
 - ...

SQL

- 我们首先关注SPJ(Select-Project-Join)查询

Select <attribute list>

From <relation list>

Where <condition list>

- 表(table, 或关系relation)R表示为R(A,B,C)
- 表R上带有选择条件的查询示例:

Select **B**

From **R**

Where **R.A = "c" \wedge R.C > 10**

- 表R(A,B,C)和S R(C,D,E)上的连接示例:
- **Select B, D**
From R, S
Where R.A = "c" \wedge S.E = 2 \wedge R.C = S.C

R	A	B	C	S	C	D	E
	a	1	10		10	x	2
	b	1	20		20	y	2
	c	2	10		30	z	2
	d	2	35		40	x	1
	e	3	45		50	y	3

Answer:

B | D
2 | x

逻辑上执行查询的方案1

- 方案1:

1. 做的笛卡儿积
2. 选择元组
3. 做投影

R X S

Select B,D

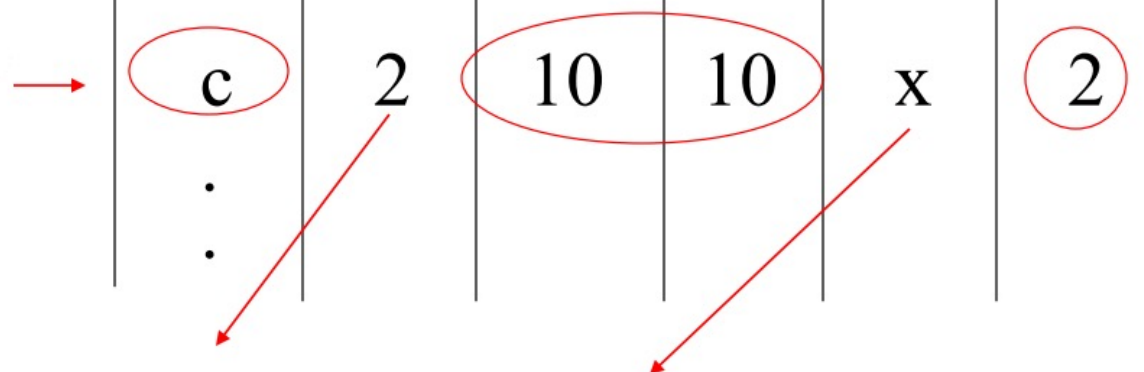
From R,S

Where R.A = "c"

\wedge S.E = 2 \wedge

R.C=S.C

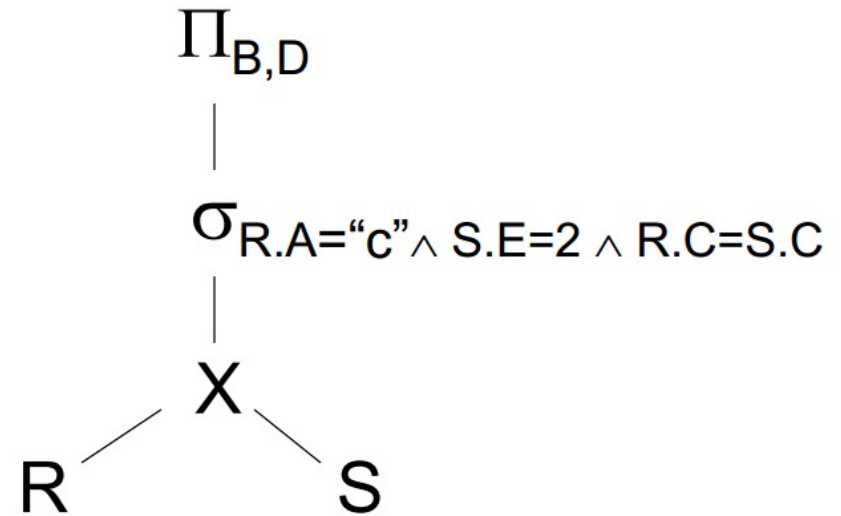
R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2
a	1	10	20	y	2
.					
.					
c	2	10	10	x	2
.					
.					



关系代数(Relational Algebra)

查询树

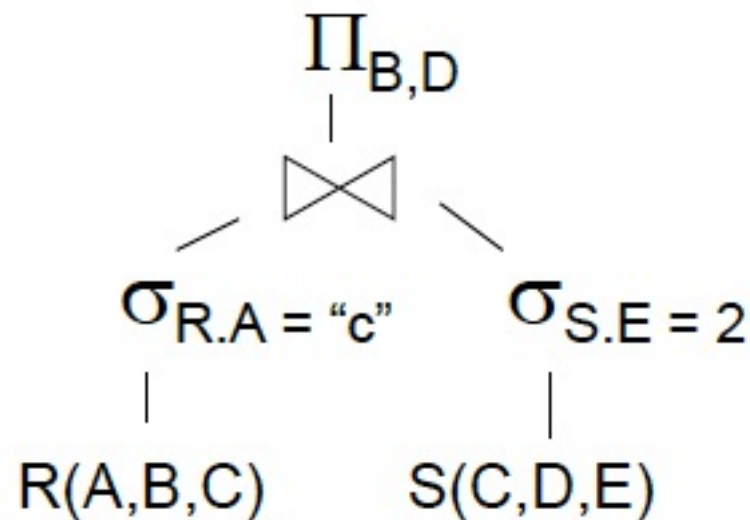
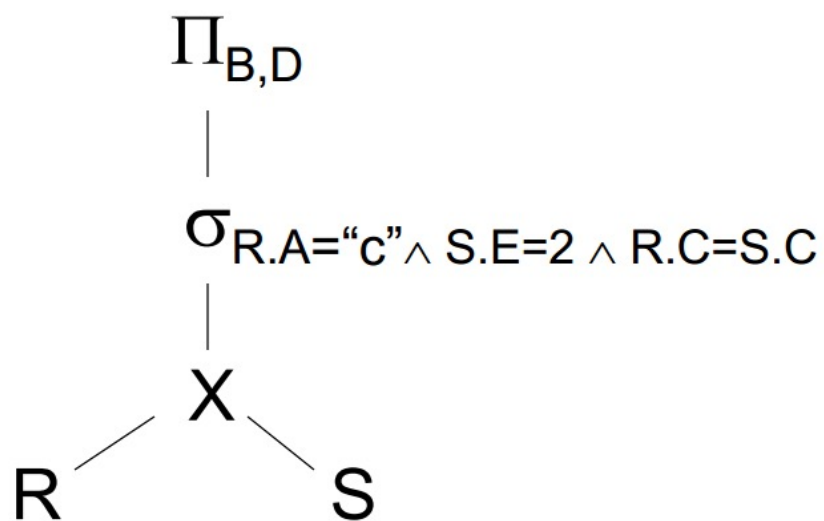
- 关系代数表示查询计划
- Select: $\sigma_{R.A="c" \wedge R.C=10}$
Project: $\Pi_{B,D}$
Cartesian Product: $R \times S$
Inner Join: $R \bowtie S$

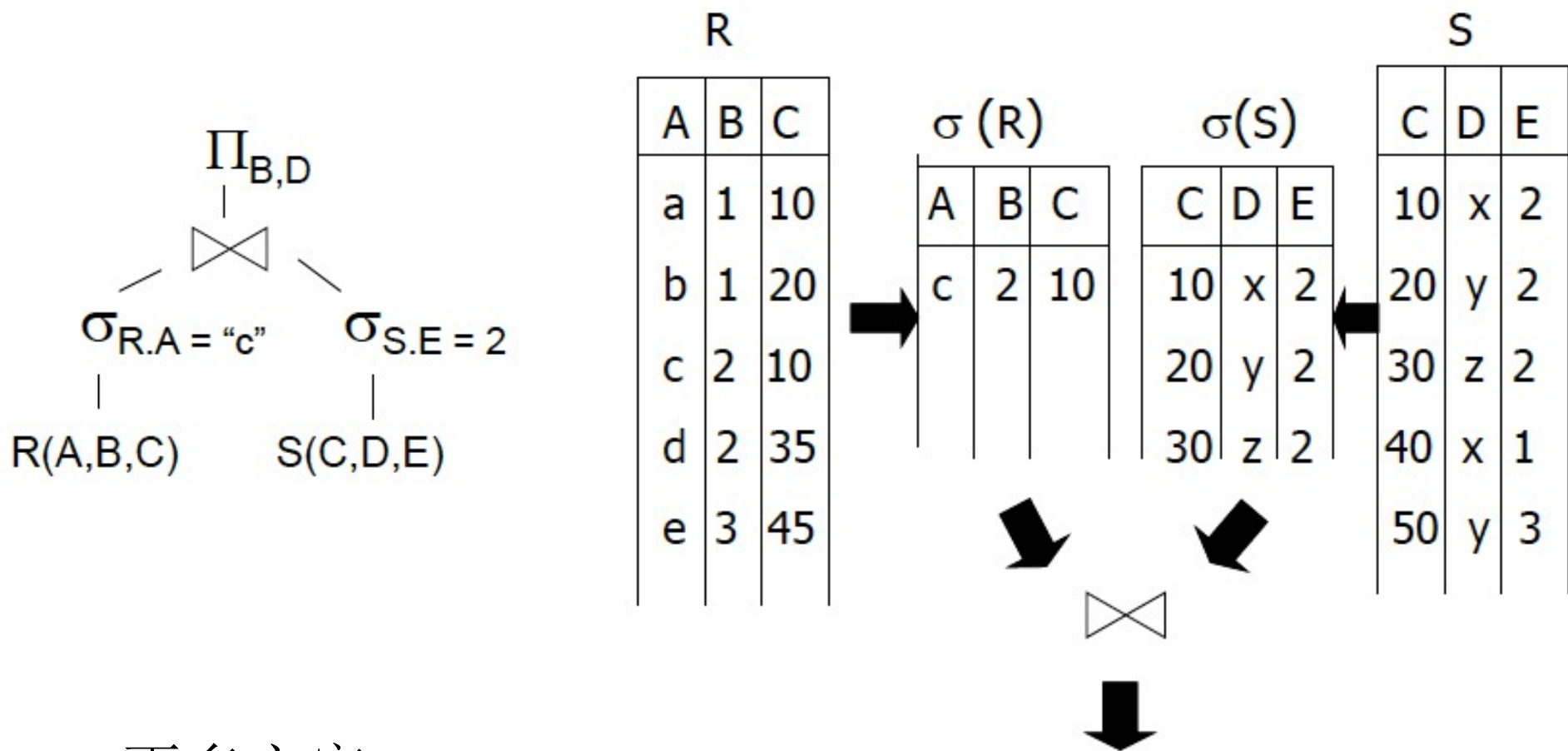


- 或: $\Pi_{B,D} [\sigma_{R.A="c" \wedge S.E=2 \wedge R.C=S.C} (R \times S)]$

关系代数的等价变换

- 方案二





- 更多方案：

- 使用表R中R.A的索引（R.A="c"）
- 对于任意的R.C的值，使用表S中S.C的索引查找

关系数据库的查询执行过程

- SQL \rightarrow Plans \rightarrow Best Plan \rightarrow Results

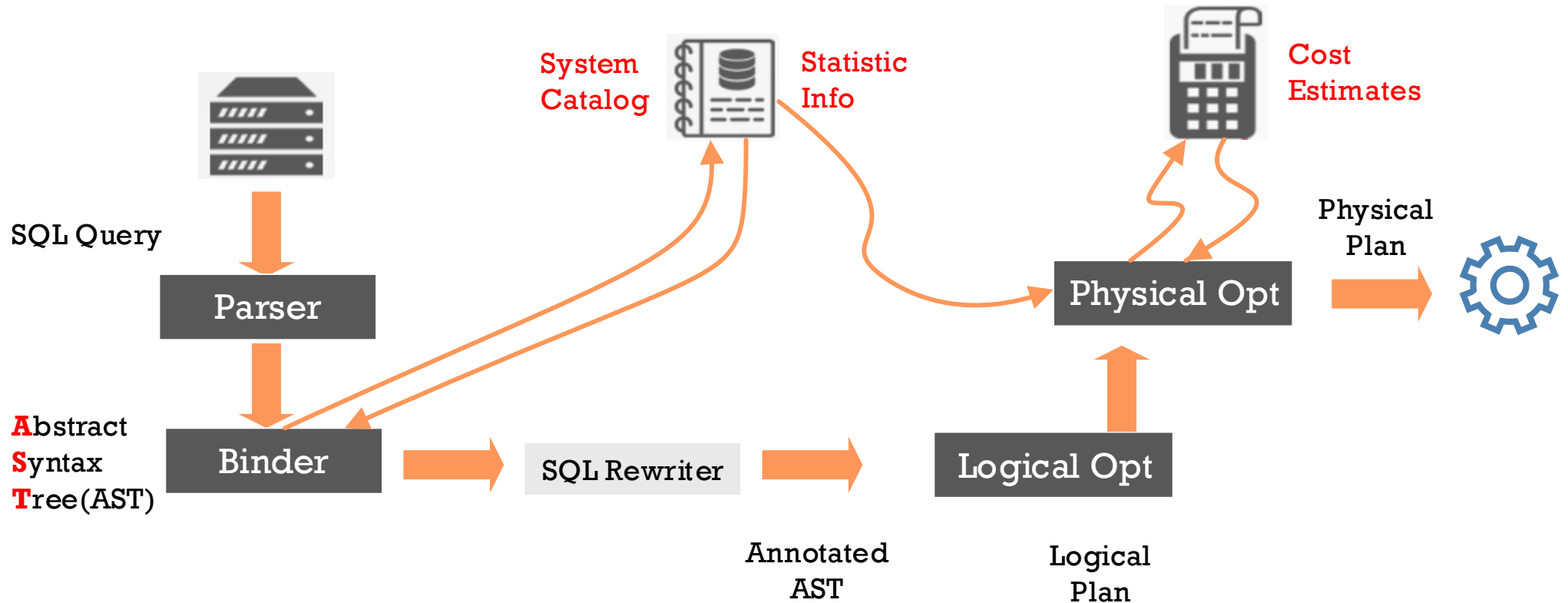
- SQL

```
SELECT C.name,C.type,I.ino  
FROM Customer C, Invoice I  
WHERE I.amount>10000 AND  
C.country='Sweden' AND  
C.cno=I.cno
```

Plans

$$\begin{aligned} & \Pi_{\text{name,type,ino}}(\sigma_{\text{amount}>10000 \wedge \text{country} = \text{'Sweden'}}(\text{Customer} \bowtie \text{Invoice})) \\ & \Pi_{\text{name,type,ino}}(\sigma_{\text{amount}>10000}(\text{Invoice}) \bowtie \\ & \quad \sigma_{\text{country} = \text{'Sweden'}}(\text{Customer})) \\ & \Pi_{\text{name,type,ino}}(\sigma_{\text{amount}>10000}(\text{Invoice} \bowtie \\ & \quad \sigma_{\text{country} = \text{'Sweden'}}(\text{Customer}))) \\ & \Pi_{\text{name,type,ino}}(\sigma_{\text{country} = \text{'Sweden'}} \\ & \quad (\sigma_{\text{amount}>10000}(\text{Invoice}) \bowtie \text{Customer})) \end{aligned}$$

关系数据库的查询执行过程

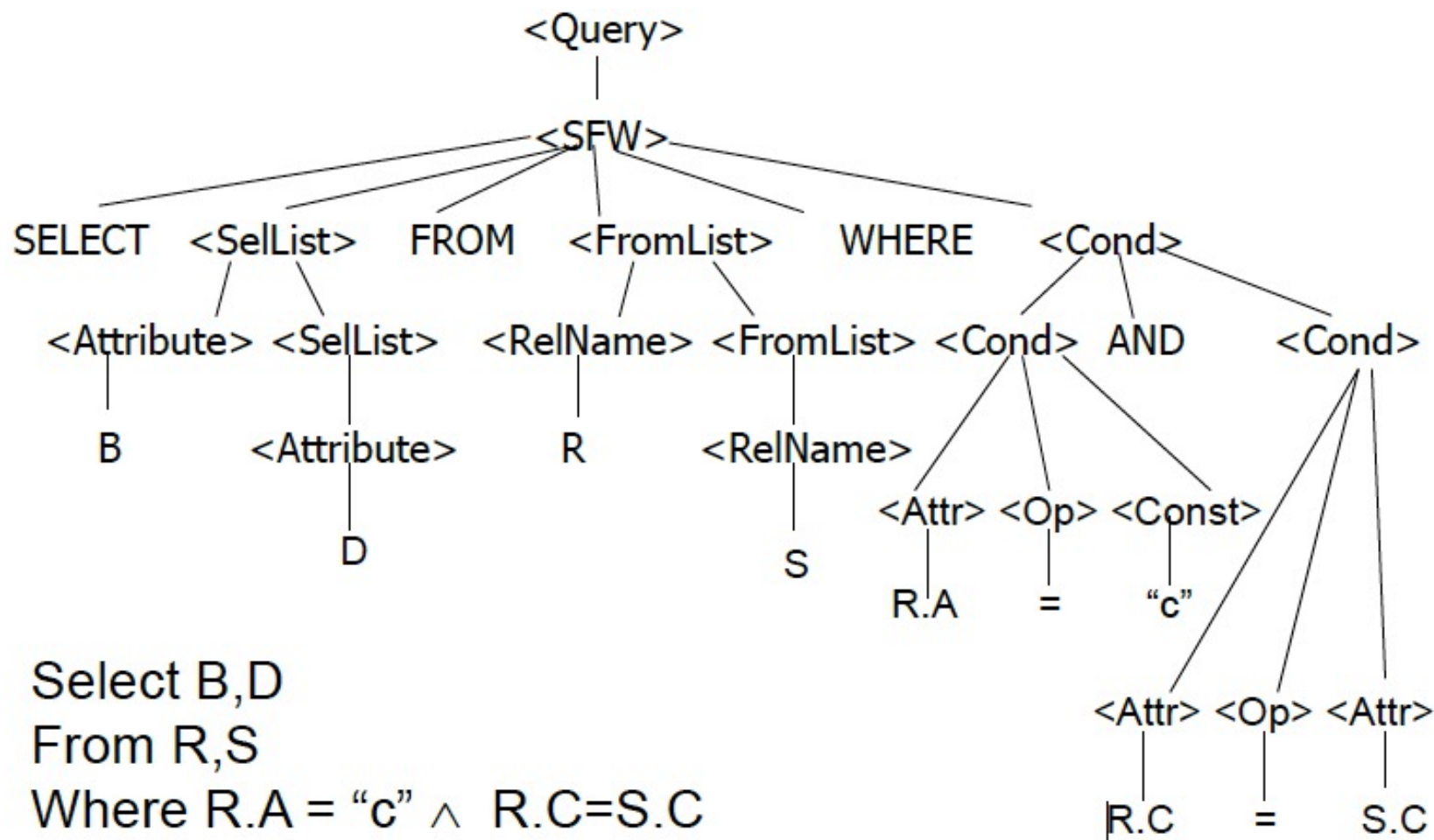


- SQL → Plans : Interpretation
- Plans → Best Plan : Query Optimization
- Best Plan → Results : Query Execution

关系数据库的查询执行过程

- 逻辑上讲
 - 查询->语法树->逻辑优化->物理优化->查询执行
 - 逻辑优化： 关系代数的等价变换
 - 物理优化： 访问路径选择， 算子执行路径选择
- 实现
 - 实际上可能很耦合， 与具体实现相关
 - 查询--(编译)-->语法树的数据结构--（展开数据结构）-->逻辑优化/物理优化耦合-->执行

Parse Tree(编译词法 语法分析)



Parser & Rewriter

- 语义检查
 - 投影列是否存在于对应的关系中?
 - 属性是否明确? (歧义? 存在?)
 - 类型检查 (R.A (int) > “2.8” (double))
- 展开视图

SQL^{Parser}----->语法树(Query Tree)_{Rewriter}

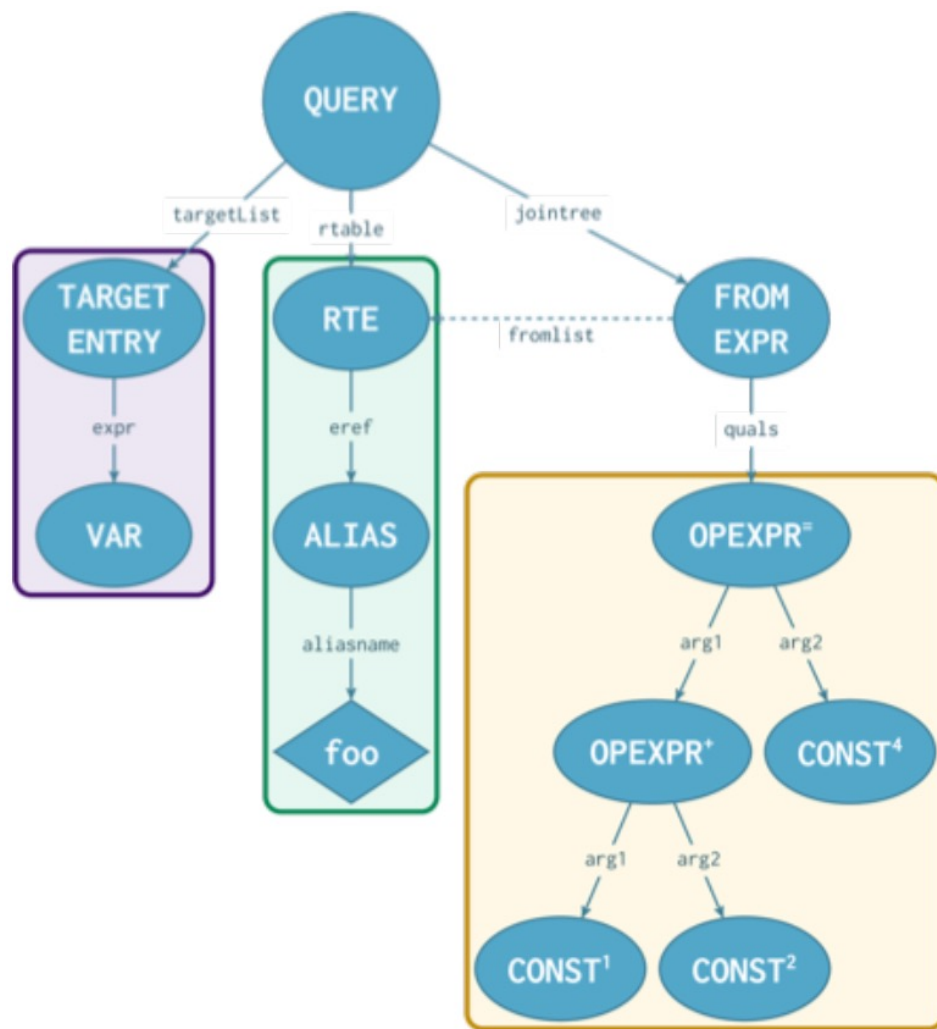
```
Select B,D
From R,S
Where R.A = "c"
      And  R.C=S.C
```

```
typedef struct Query{
    NodeTag type;
    CmdType commandType; /* 类型: select、insert、update、delete、
                           utility */
    int resultRelation; /* rtable index of target relation */
    RangeVar *into; /* target relation for SELECT INTO */
    List *intoOptions; /* options from WITH clause */
    char *intoTableSpaceName; /* table space to use, or NULL */
    bool hasAggs; /* has aggregates in tlist or havingQual */
    bool hasSubLinks; /* has subquery SubLink */
    List *rtable; /* list of range table entries */
    FromExpr *jointree; /* table join tree (FROM and WHERE clauses) */
    List *targetList; /* target list (of TargetEntry) */
    List *returningList; /* return-values list (of TargetEntry) */
    List *groupClause; /* a list of GroupClause's */
    Node *havingQual; /* qualifications applied to groups */
    List *distinctClause; /* a list of SortClause's */
    List *sortClause; /* a list of SortClause's */
    Node *limitOffset; /* # of result tuples to skip (int8 expr) */
    Node *limitCount; /* # of result tuples to return (int8 expr) */
    List *rowMarks; /* a list of RowMarkClause's */
    Node *setOperations; /* set-operation tree i */
    List *resultRelations; /* integer list of RT indexes, or NIL */
    List *returningLists; /* list of lists of TargetEntry, or NIL */
} Query;
```

Parser

语法树的数据结构

- SELECT a
FROM foo
WHERE 1 + 2 = 4;

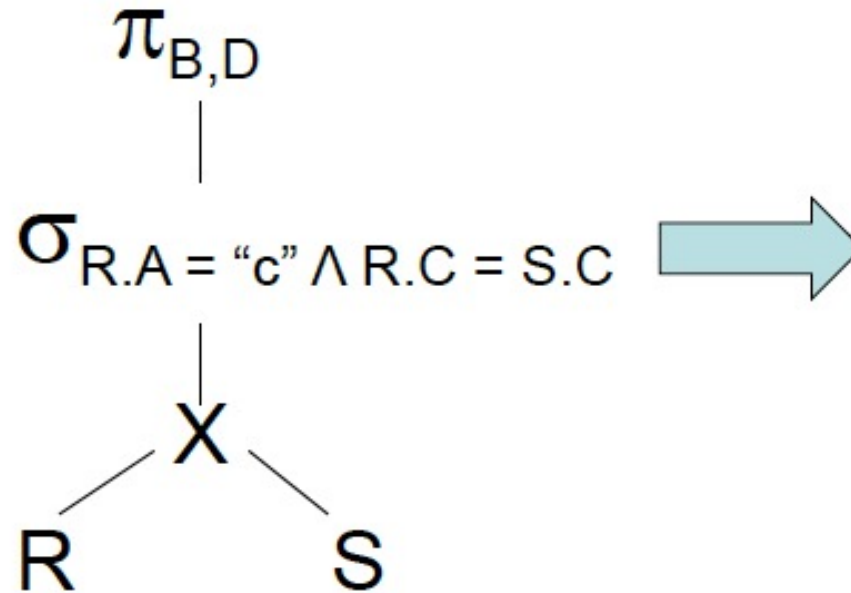


查询计划

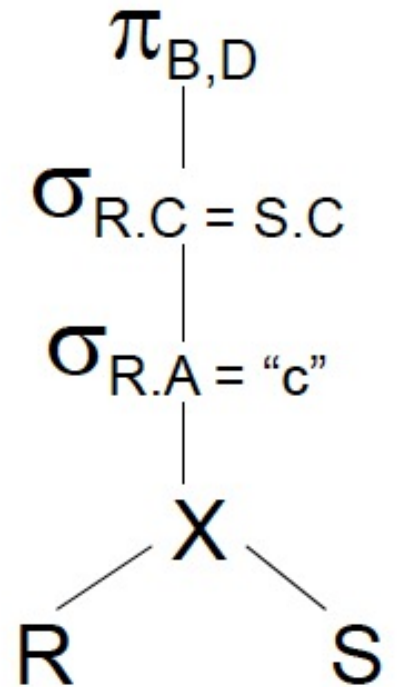
- 在数据库实现中表现为不同的数据结构
 - 通常为树状递归结构
 - 从语法树展开得到查询计划

逻辑查询优化

Select B,D
From R,S
Where R.A = "c"
And R.C=S.C



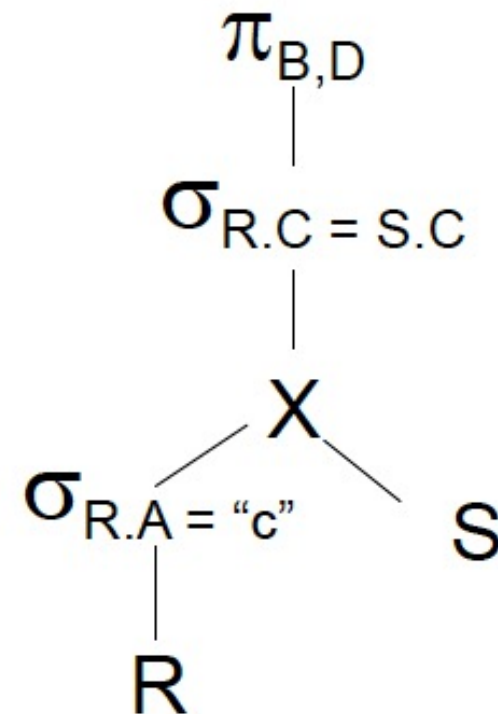
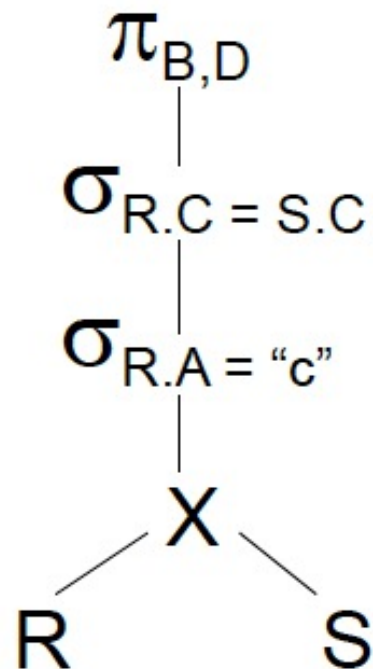
- $\Pi_{B,D} [\sigma_{R.A="c" \wedge R.C = S.C} (R \times S)]$



- $\Pi_{B,D} [\sigma_{R.C=S.C} [\sigma_{R.A="c"} (R \times S)]]$

逻辑查询优化

Select B,D
From R,S
Where R.A = "c"
And R.C=S.C

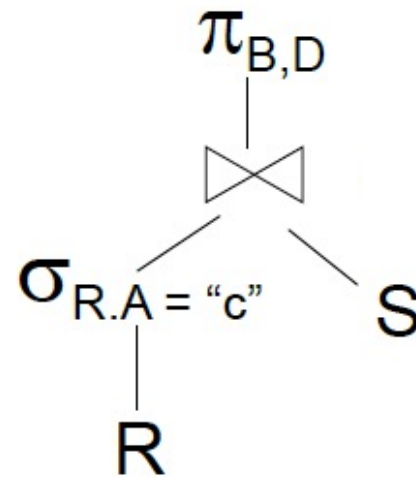
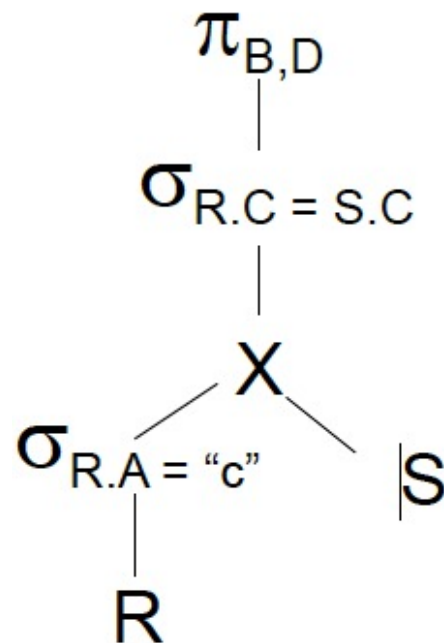


$\Pi_{B,D} [\sigma_{R.C=S.C} [\sigma_{R.A="c"}(R \times S)]]$

$\Pi_{B,D} [\sigma_{R.C=S.C} [\sigma_{R.A="c"}(R)] \times S]$

逻辑查询优化

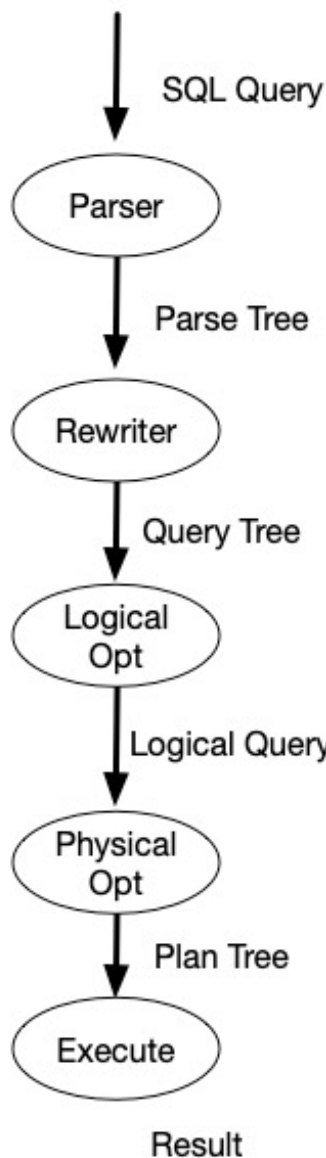
Select B,D
From R,S
Where R.A = "c"
And R.C=S.C



$\Pi_{B,D} [\sigma_{R.C=S.C} [\sigma_{R.A="c"}(R)] \bowtie S]$

$\Pi_{B,D} [[\sigma_{R.A="c"}(R)] \bowtie S]$

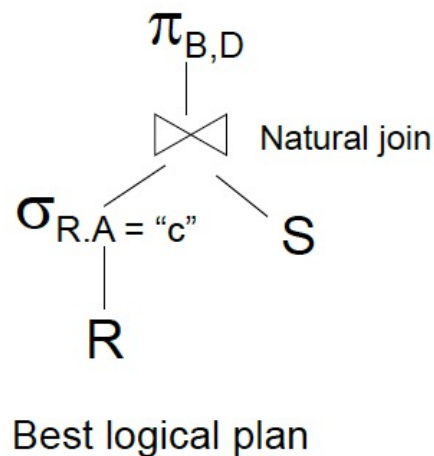
物理查询优化



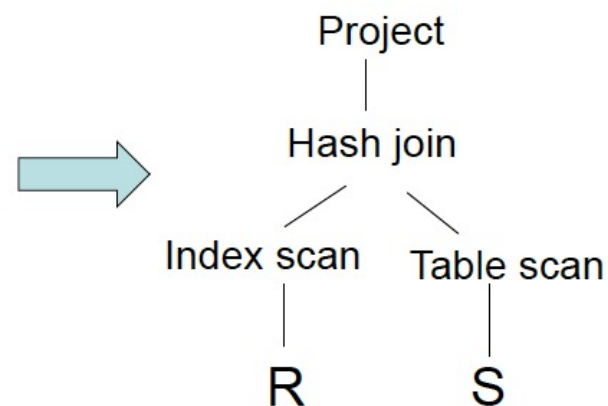
Enumerate possible physical plans

Find the cost of each plan

Pick plan with minimum cost

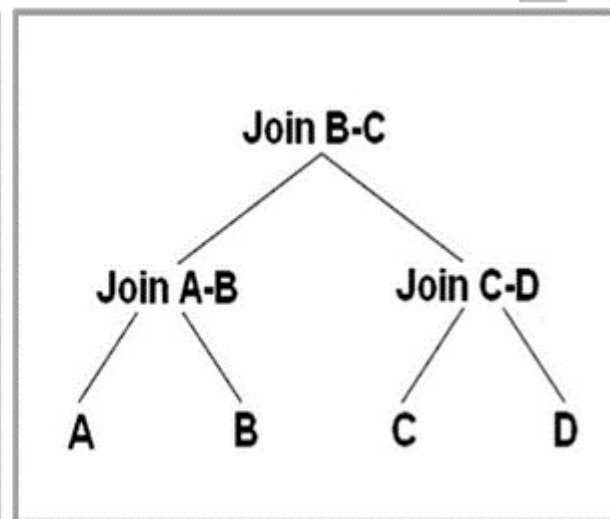
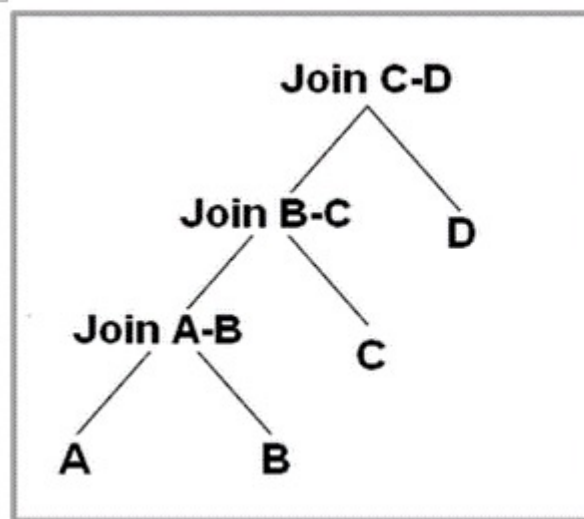


Select B,D
From R,S
Where R.A = "c"
And R.C=S.C

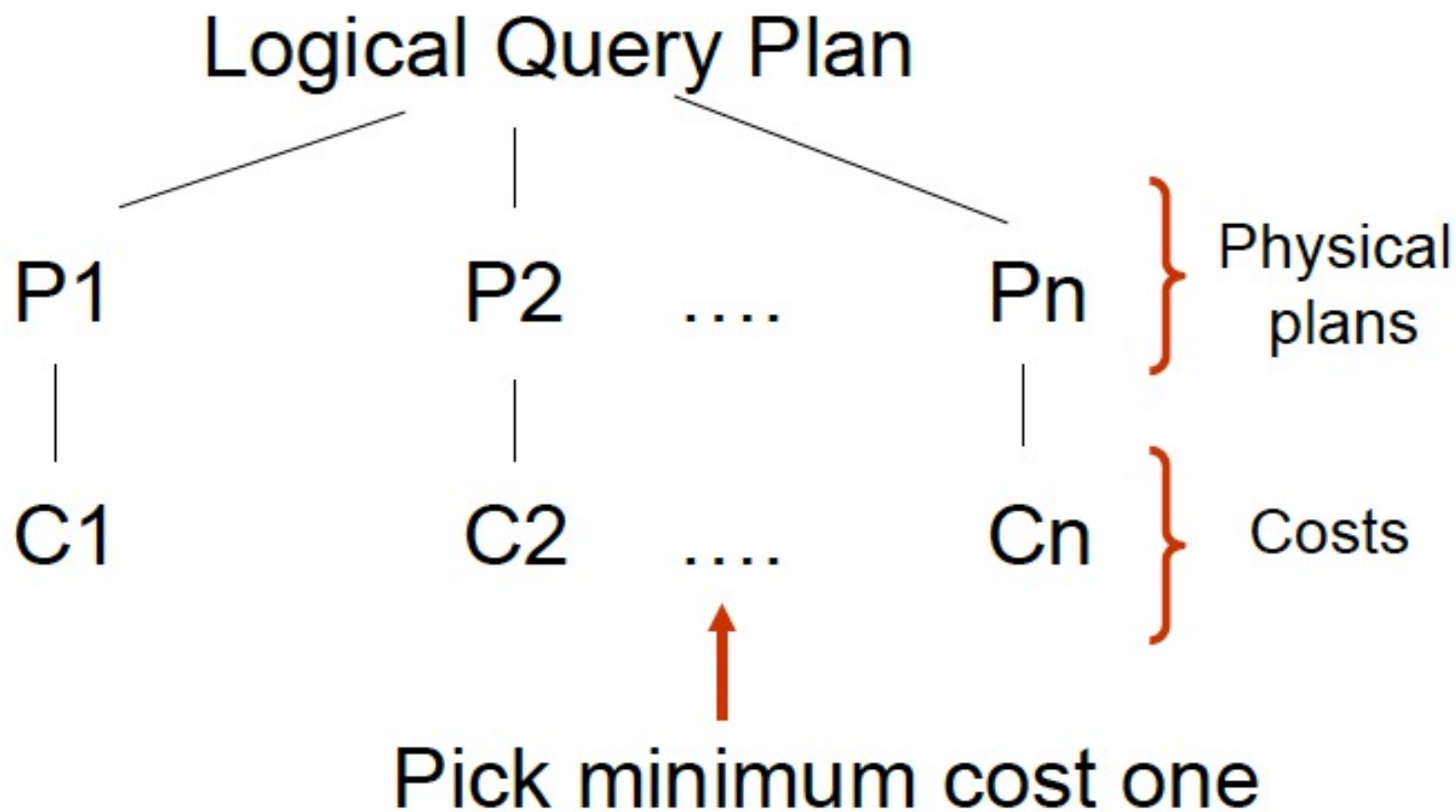


物理查询优化

- 路径选择
- 连接方式
- 连接顺序



物理查询优化



物理计划案例

- 可以使用explain功能输出物理计划

psql (9.3.3)

Type "help" for help.

```
pat=> EXPLAIN SELECT "users".* FROM "users" WHERE "users"."name" = 'Captain Nemo'
      ORDER BY "users"."id" ASC LIMIT 1;
               QUERY PLAN
```

```
-----
Limit  (cost=21.56..21.57 rows=1 width=551)
->  Sort  (cost=21.56..21.57 rows=1 width=551)
     Sort Key: id
     ->  Seq Scan on users  (cost=0.00..21.55 rows=1 width=551)
          Filter: ((name)::text = 'Captain Nemo'::text)
```

(5 rows)