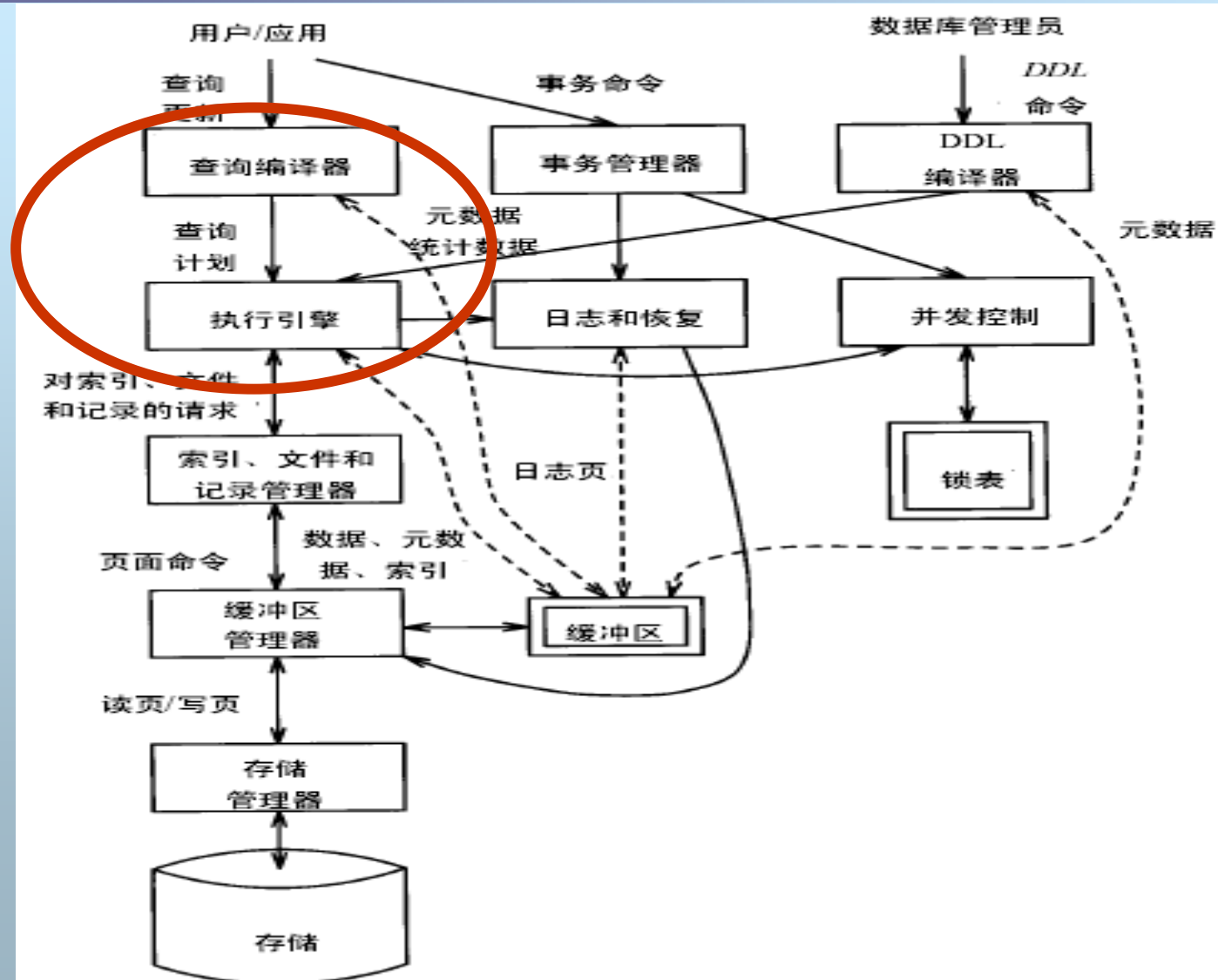


# Query Optimization

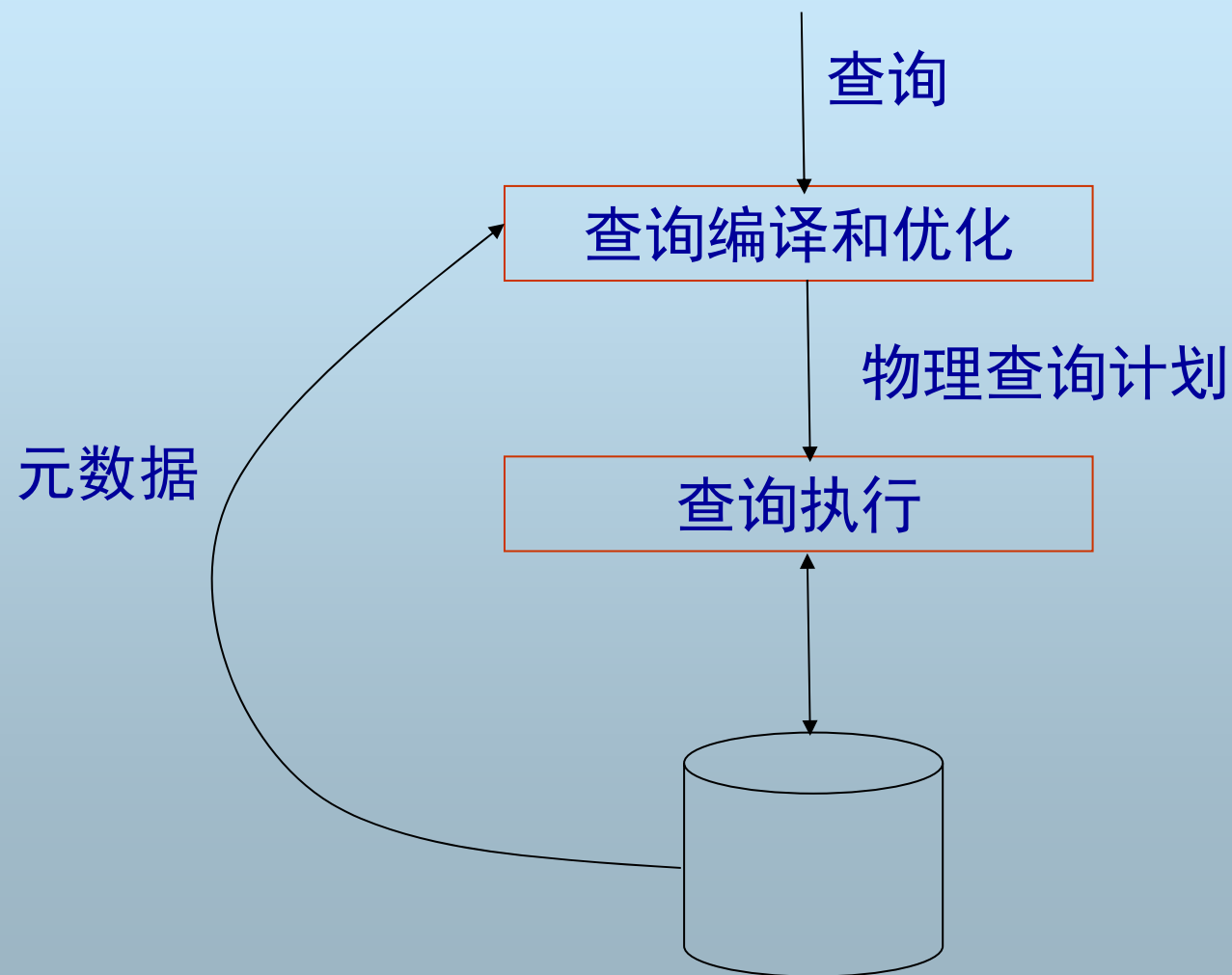


**Chp.16 in textbook**

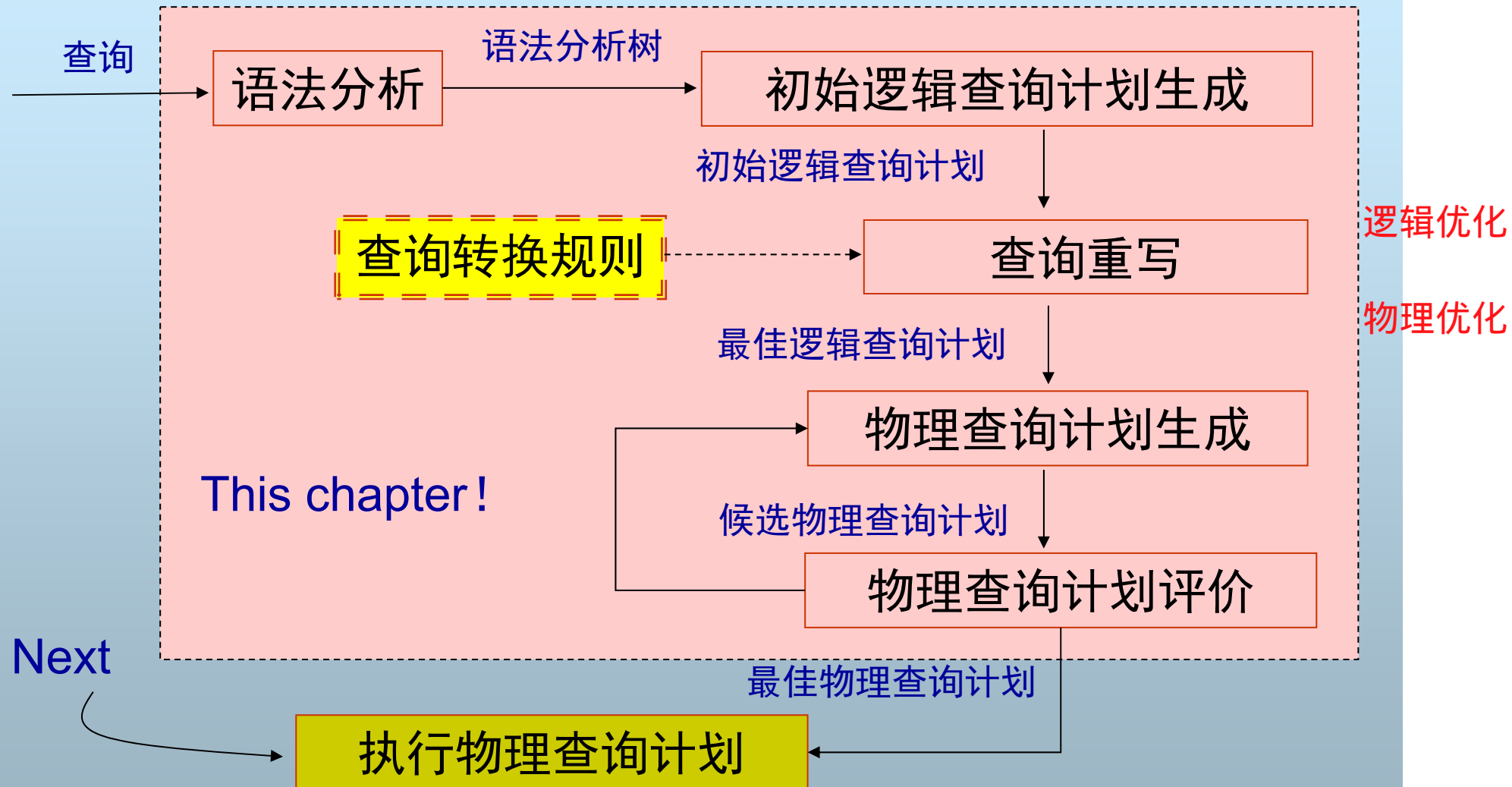
# 查询处理器



# 查询处理概述



# 查询处理概述



# 主要内容

- 语法分析(Parsing)
- 逻辑查询计划生成(Logical Query Plan)
- 查询重写(Query Rewrite)
- 查询计划代价估计(Cost Estimation)
- 物理查询计划选择(Physical Query Plan)

# 一、语法分析

## ■ 构造语法分析树(Parsing Tree)

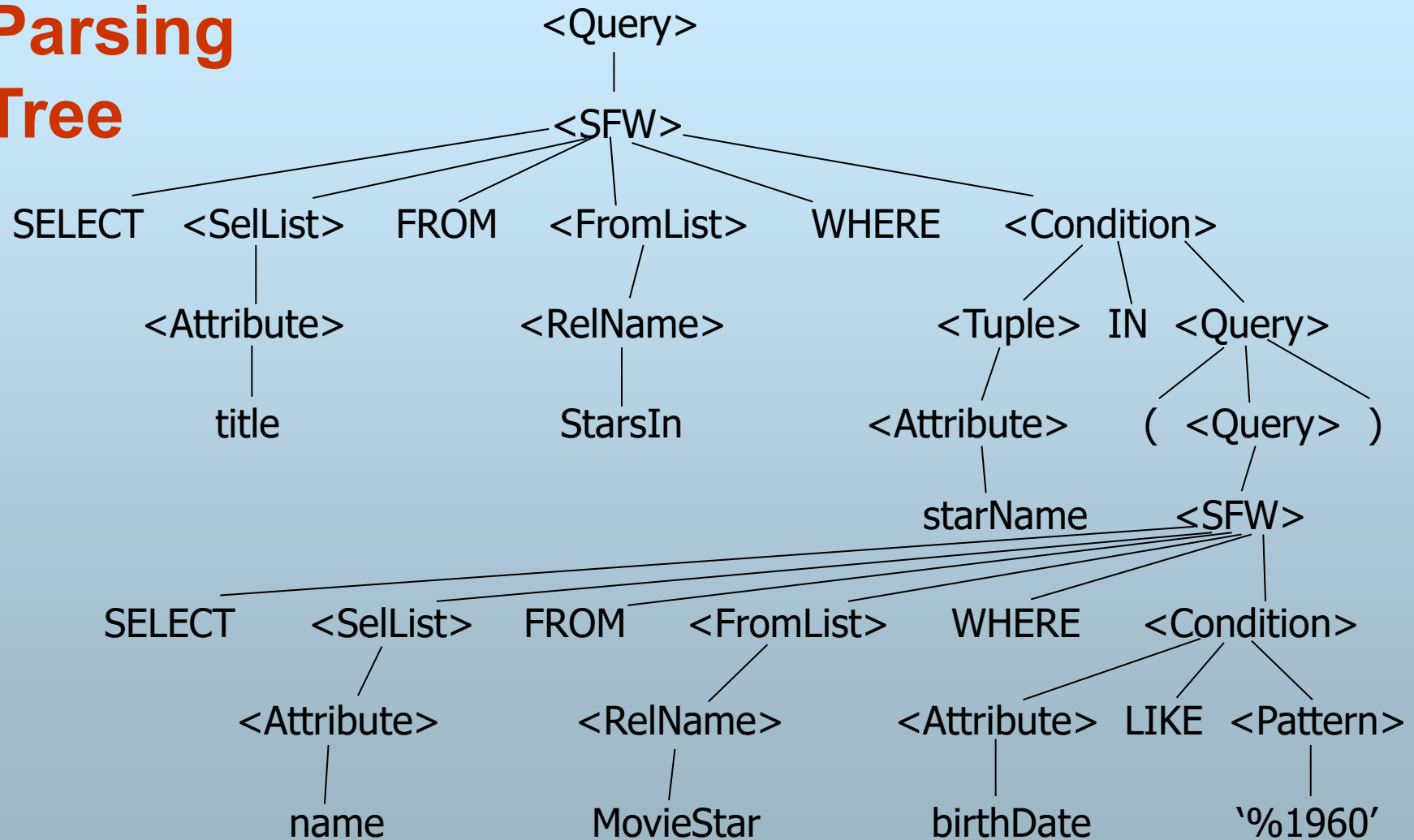
# 1、SQL查询语法分析

```
SELECT title  
FROM StarsIn  
WHERE starName IN (  
    SELECT name  
    FROM MovieStar  
    WHERE birthdate LIKE '%1960'  
);
```

**(Find the movies with stars born in 1960)**

# 1、SQL查询语法分析

## Parsing Tree

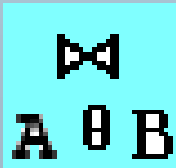




## 二、初始逻辑查询计划生成

- 将语法分析树转换为代数表达式树——逻辑查询计划
  - **Typical:** 关系代数

# 1、关系代数回顾

- $\cup$ : 并
- $\cap$ : 交
- $\sigma$ : 选择
- $\Pi$ : 投影
- $-$ : 差
- $\bowtie$ : 自然连接
-  : **Theta 连接**  
 $A \theta B$
- .....

## 2、关系代数与SQL

- 关系代数是**SQL**的代数表达
- 关系代数表达式  $\Leftrightarrow$  **SQL**查询

### Note:

- 关系代数基于集合(**SET**)运算
- **SQL**基于包(**BAG**)运算
- 集合：无重复元素
- 包：允许重复元素

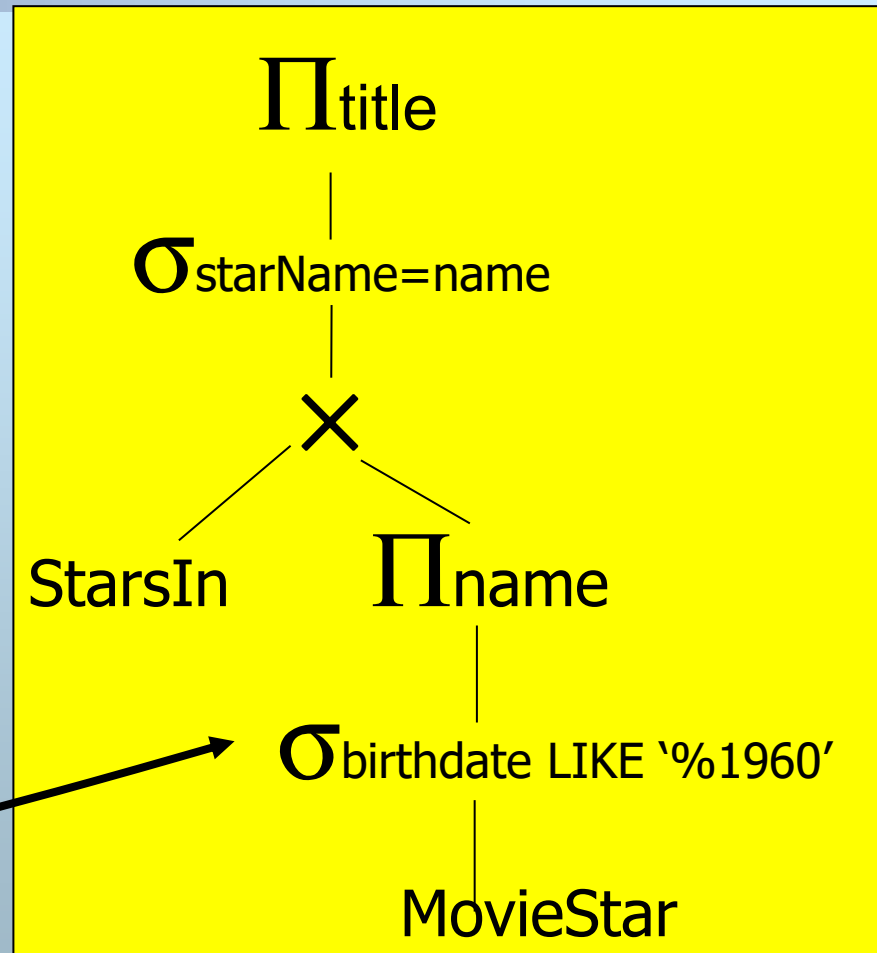
# 3、逻辑查询计划

- 与语法分析树类似
  - 内结点：关系运算符
  - 叶结点：关系

# 例：逻辑查询计划生成

```
SELECT title
FROM StarsIn
WHERE starName IN (
  SELECT name
  FROM MovieStar
  WHERE birthdate LIKE
  '%1960'
);
```

Logical Query Plan

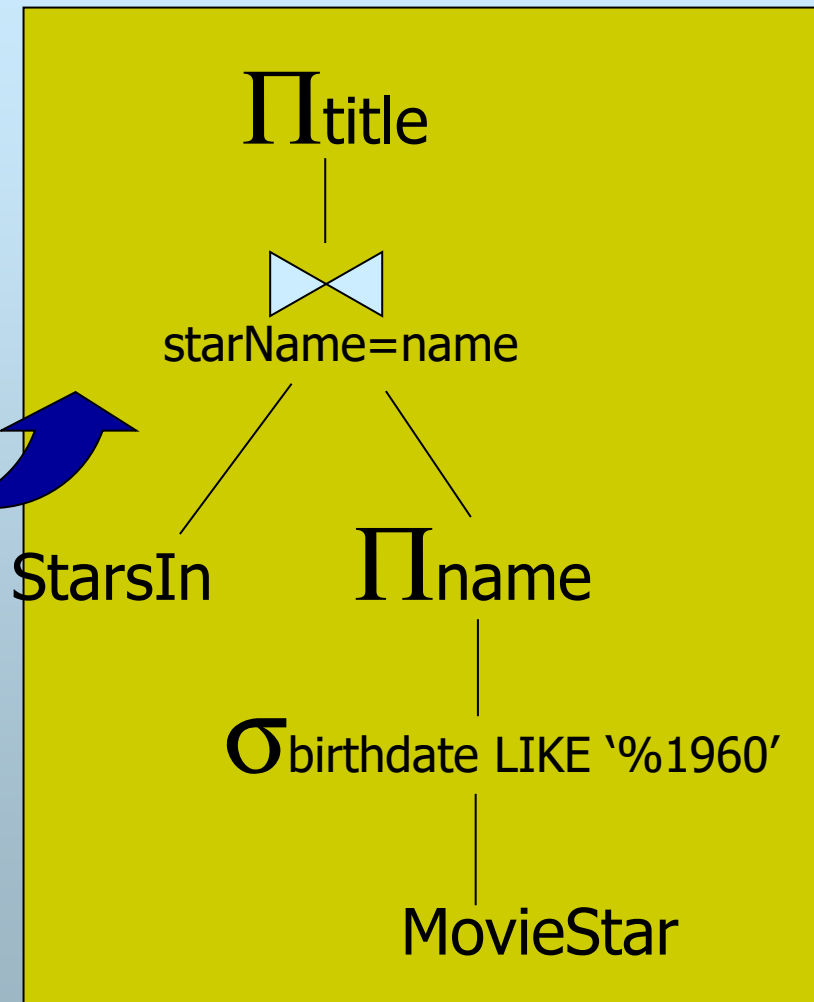
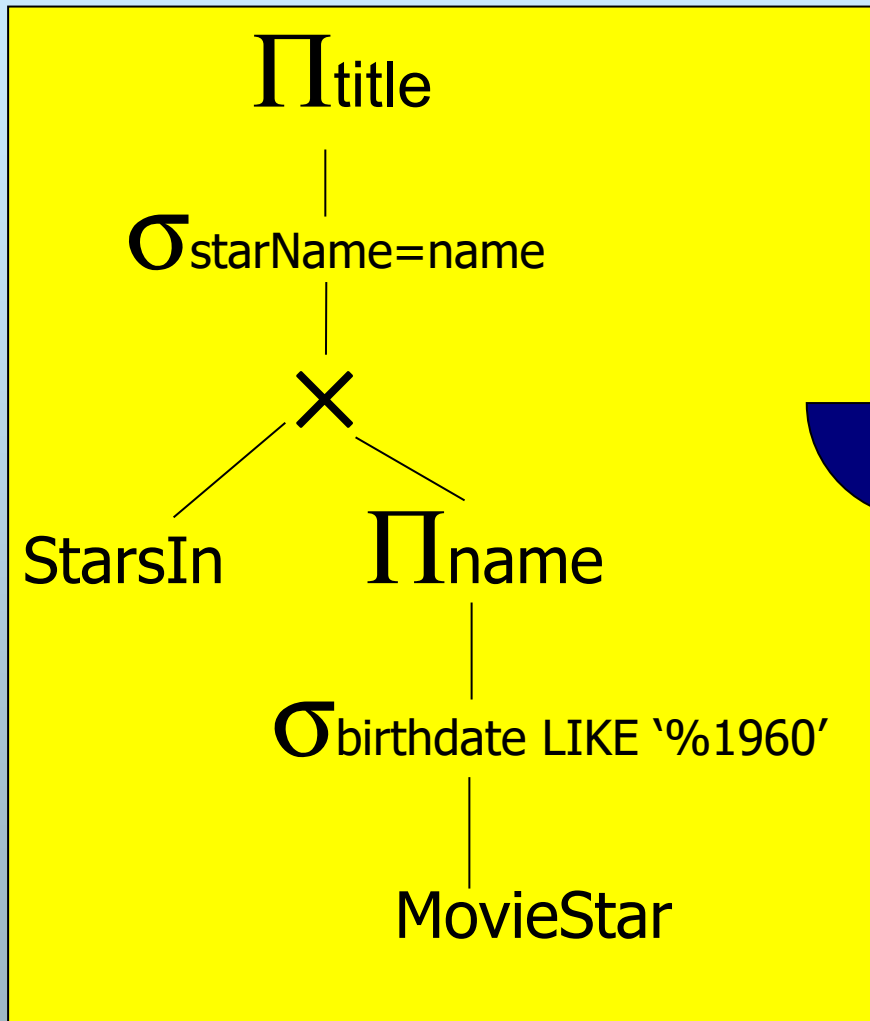


$\Pi_{\text{title}}(\sigma_{\text{starName}=\text{name}}(\text{StarsIn} \times \Pi_{\text{name}}(\sigma_{\text{birthdate LIKE '%1960'}}(\text{MovieStar}))))$

## 三、查询重写

- 将初始逻辑查询计划转换为优化的逻辑查询计划(Maybe)
  - 基于代数转换规则

# 1、查询重写例子



## 2、转换规则

- **Transformation rules**
- 运用转换规则，将一个代数表达式转换为另一个等价的代数表达式



## 2、转换规则

### ■ 涉及自然连接、并、交、笛卡儿积的交换律和结合律

- $R \times S = S \times R; (R \times S) \times T = R \times (S \times T)$

- $R \bowtie S = S \bowtie R; (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

- $R \cup S = S \cup R; (R \cup S) \cup T = R \cup (S \cup T)$

- $R \cap S = S \cap R; (R \cap S) \cap T = R \cap (S \cap T)$

## 2、转换规则

R

A	B
10	20
20	30
30	40

S

A	C
10	20
20	30
30	40

T

C	D
20	20
10	30
10	40

- $(R \bowtie S) \bowtie T$ : 中间结果  $R \bowtie S$  产生 3 条记录
- $R \bowtie (S \bowtie T)$ : 中间结果  $S \bowtie T$  产生 1 条记录

查询代价不同

## 2、转换规则

### ■ 选择上的转换规则

●  $\sigma_{c1 \wedge c2}(R) = \sigma_{c1}(\sigma_{c2}(R))$

●  $\sigma_{c1 \vee c2}(R) = (\sigma_{c1}(R)) \cup (\sigma_{c2}(R))$

集合并

## 2、转换规则

### ■ SQL

- **Union All** ———包并（**SUM**）
- **Union**———集合并
- 例如，**Student**表和**Staff**表
  - ◆ “返回所有男学生和男教师的姓名”

```
Select name from student where gender='M'  
Union All  
Select name from staff where gender='M'
```

## 2、转换规则

### ■ 选择 + 自然连接

Let  $p$  = predicate with only  $R$  attribs

$q$  = predicate with only  $S$  attribs

$m$  = predicate with only  $R, S$  attribs

下推选择：选择尽可能早做

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

可推出另一些规则

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

## 2、转换规则

### ■ 选择 + 自然连接

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) = \sigma_m [(\sigma_p R) \bowtie (\sigma_q S)]$$

$$\sigma_{p \vee q} (R \bowtie S) =$$

$$[(\sigma_p R) \bowtie S] \cup [R \bowtie (\sigma_q S)]$$

## 2、转换规则

### ■ 投影+自然连接

Let  $x$  = subset of  $R$  attributes

$y$  = subset of  $S$  attributes

$z$  = intersection of  $R, S$  attributes

$$\pi_{xy} (R \bowtie S) =$$

$$\pi_{xy} \{ [\pi_{xz} (R)] \bowtie [\pi_{yz} (S)] \}$$

减小参与连接操作的磁盘块数

### 3、转换规则的几点思考

#### ■ 转换的最终目的

- 减少查询的开销(I/O次数)

#### ■ 转换的直接目的

- 减少查询执行时的中间关系大小（元组数）
- 减少元组的大小

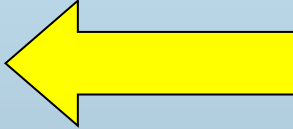


Do select early  
Choose join orders

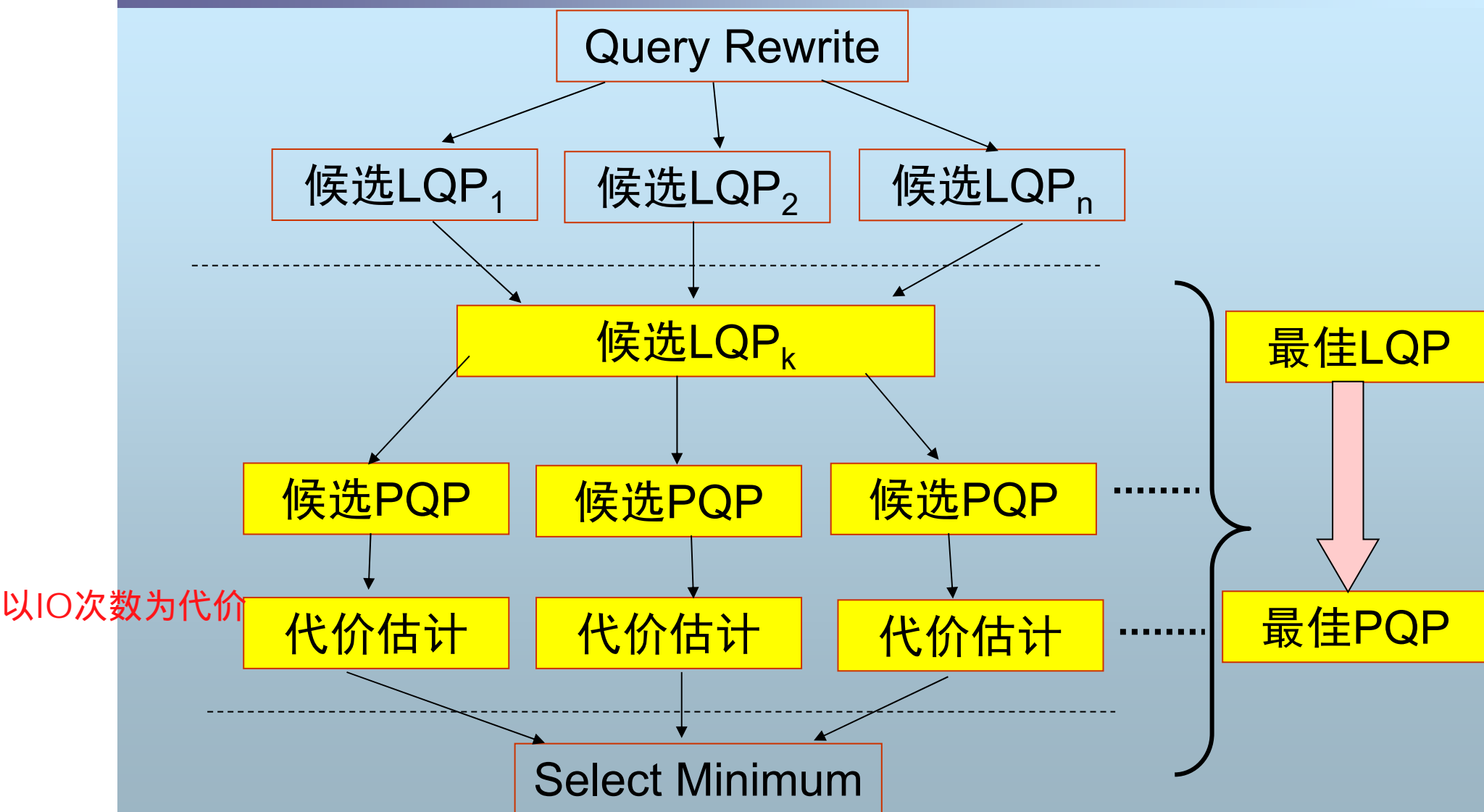
Do project before selections



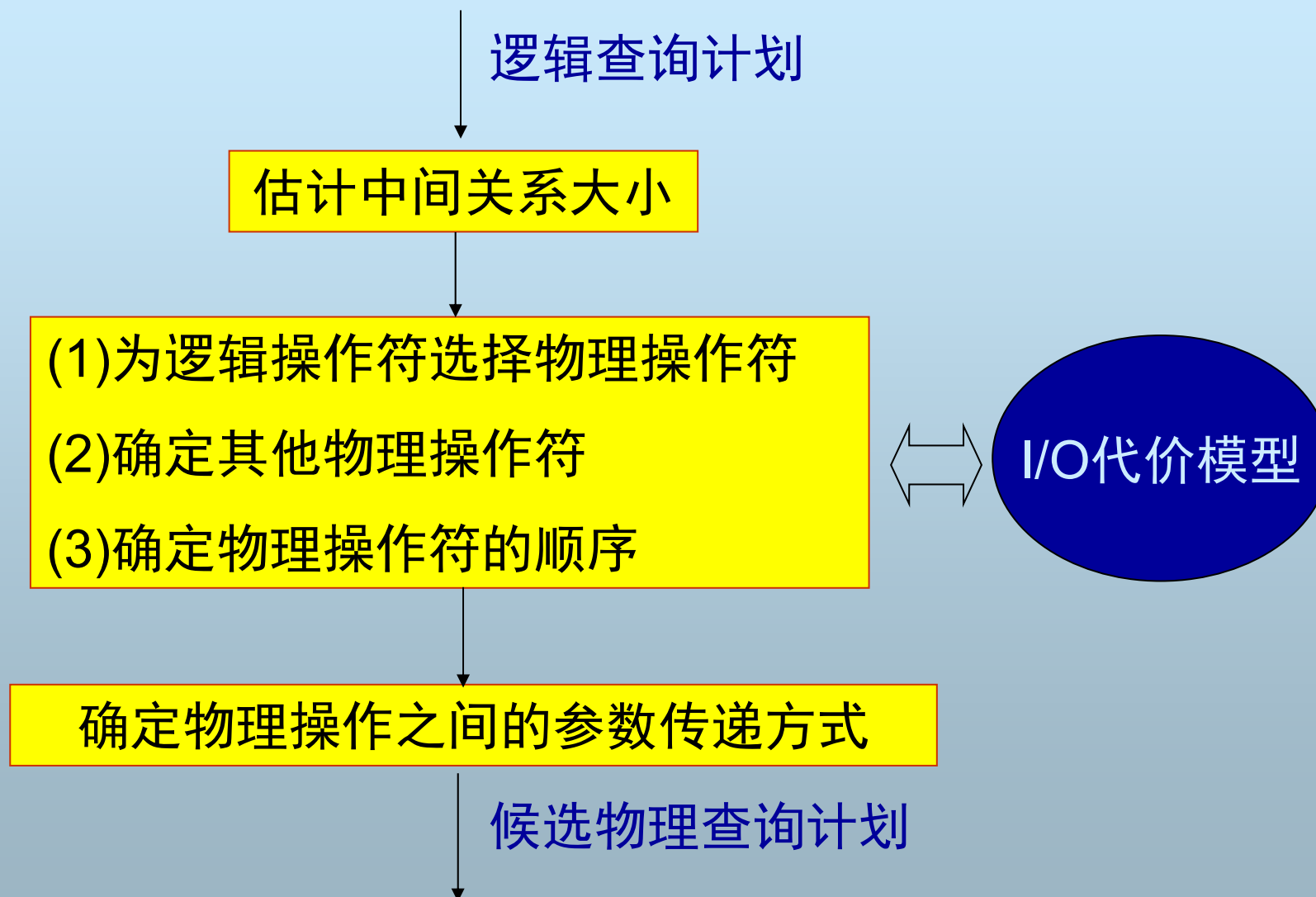
# Where we are?

- 语法分析(Parsing)
- 逻辑查询计划生成(Logical Query Plan)
- 查询重写(Query Rewrite)
- 查询计划代价估计  Next
- 物理查询计划选择(Physical Query Plan)

## 四、查询代价估计(Cost Estimation)



## 四、查询代价估计(Cost Estimation)



## 四、查询代价估计(Cost Estimation)

- 中间关系大小估计
- **I/O**代价估计
- 物理查询计划生成

# 1、中间结果的大小估计

- 需要使用一些统计量(**statistics**)
  - **$T(R)$**  :  $R$ 的元组数
  - **$S(R)$**  :  $R$ 中每个元组的大小(**bytes**)
  - **$V(R, A)$**  :  $R$ 的属性 **$A$** 上的不同值数
  - **$B(R)$** : 容纳 **$R$** 所有元组所需的块数 估计IO次数时使用
- **These statistics should be held in the database!**

# 1、中间结果的大小估计

## Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

$$T(R) = 5$$

$$S(R) = 37$$

$$V(R,A) = 3$$

$$V(R,C) = 5$$

$$V(R,B) = 1$$

$$V(R,D) = 4$$

# 1、中间结果的大小估计

## ■ $W = R1 \times R2$ 的大小估计

- $T(W) = T(R1) * T(R2)$
- $S(W) = S(R1) + S(R2)$

# 1、中间结果的大小估计

- $W = \sigma_{A=a}(R)$  的大小估计
  - $S(W) = S(R)$
  - $T(W) = ?$



# 1、中间结果的大小估计

## ■ $W = \sigma_{A=a}(R)$ 的大小估计

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

# 1、中间结果的大小估计

- $W = \sigma_{z=val}(R)$ : 假设 $z$ 上的值在 $V(R,z)$ 个不同值上均匀分布

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$



$$T(W) = \frac{T(R)}{V(R, z)}$$

# 1、中间结果的大小估计

## ■ $W = \sigma_{z > val}(R)$ 的大小估计

### ● 一种估计

◆  $T(W) = T(R) / 2$

### ● 另一种估计

◆  $T(W) = T(R) / 3$

### ● 使用“范围”



# 1、中间结果的大小估计

- $W = \sigma_{z > val} (R)$  的大小估计：使用范围

Example

R	Z

Min=1       $V(R,Z)=10$

↕

Max=20       $W = \sigma_{z > 15} (R)$

$$f = \frac{20-15}{20-1+1} = \frac{5}{20} \quad (\text{fraction of range})$$

$$T(W) = f \times T(R)$$

# 1、中间结果的大小估计

- $W = \sigma_{z \neq \text{val}}(R)$  的大小估计

$$T(W) = T(R) - \frac{T(R)}{V(R, z)}$$

# 1、中间结果的大小估计

■ 总结：选择大小的估计  $W = \sigma_p(R)$

■  $T(W) = s * T(R)$

●  $s$ 是选中率

$$s = \begin{cases} 1 / V(R,z) & p \text{ 为 “=” 比较时} \\ (V(R,z) - 1) / V(R,z) & p \text{ 为 “≠” 比较时} \\ \left. \begin{array}{l} 1 / 2 \text{ 或} \\ 1 / 3 \text{ 或} \\ \text{范围命中率 } f \end{array} \right\} & p \text{ 为 “≥、≤、<、>” 时} \end{cases}$$

# 1、中间结果的大小估计

- $W = R1 \bowtie R2$  的大小估计

Let  $x$  = attributes of  $R1$

$y$  = attributes of  $R2$

Case 1

$$X \cap Y = \emptyset$$

Same as  $R1 \times R2$

# 1、中间结果的大小估计

Case 2

$$W = R1 \bowtie R2$$

$$X \cap Y = A$$

R1

A	B	C
...	...	...

R2

A	D
...	...

## Assumption:

$V(R1, A) \leq V(R2, A) \Rightarrow R1.A$ 上的值都在R2中

$V(R2, A) \leq V(R1, A) \Rightarrow R2.A$ 上的值都在R1中

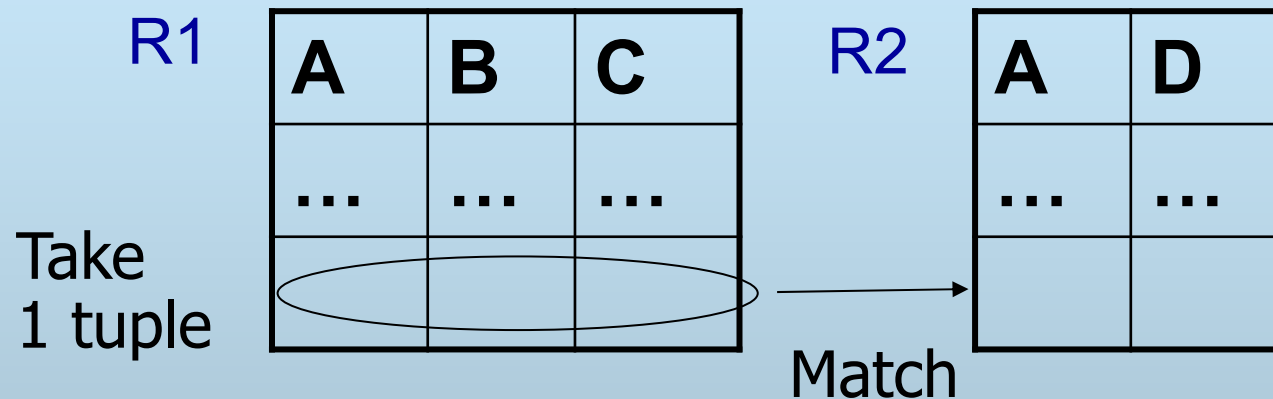
“值集的包含 **containment of value sets**”

**see Sec. 16.4.4**



# 1、中间结果的大小估计

■  $W = R1 \bowtie R2 : V(R1, A) \leq V(R2, A)$



R1中的一个元组在R2中有  $\frac{T(R2)}{V(R2, A)}$  个元组匹配

$$T(W) = \frac{T(R2)}{V(R2, A)} \times T(R1)$$

# 1、中间结果的大小估计

■  $W = R1 \bowtie R2 : V(R2, A) \leq V(R1, A)$

$$T(W) = \frac{T(R1)}{V(R1, A)} \times T(R2)$$

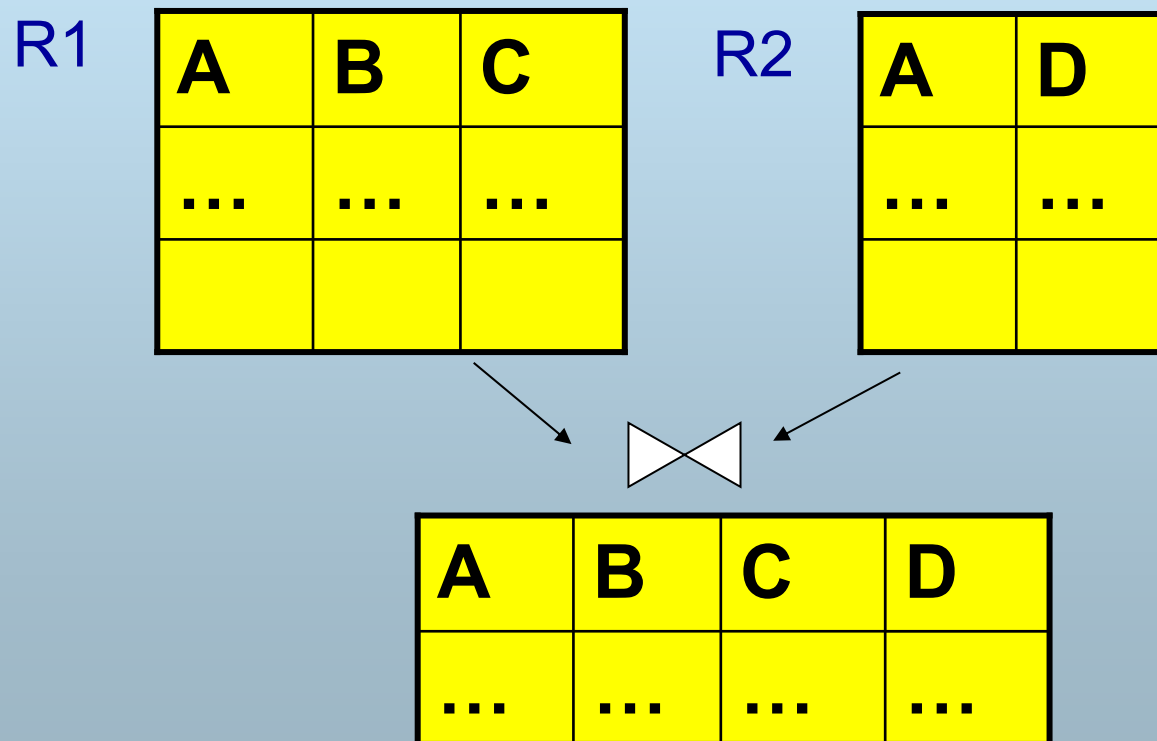
In General

$$T(W) = \frac{T(R1) \cdot T(R2)}{\max \{V(R1, A), V(R2, A)\}}$$

# 1、中间结果的大小估计

■  $W = R1 \bowtie R2 : S(W) = ?$

●  $S(W) = S(R1) + S(R2) - S(A)$



# 1、中间结果的大小估计

■  $W = R1 \bowtie R2 : V(W,*) = ?$

“preservation of value sets”  
值集的保持: see Sec.16.4.4

对于  $W = R1(A,B,C) \bowtie R2(A,D)$

我们可以假设:

$$V(W,B) = V(R1,B)$$

$$V(W,C) = V(R1,C)$$

$$V(W,D) = V(R2,D)$$

$$V(W,A) = \min\{V(R1,A), V(R2,A)\}$$

假设满足值集的包含

# 1、中间结果的大小估计

- $W = R1 \times R2$
- $W = \sigma_p(R)$
- $W = R1 \bowtie R2$
- $W = [ \sigma_{A=a}(R1) ] \bowtie R2 \xrightarrow{\text{Next}}$

# 1、中间结果的大小估计

- $W = [ \sigma_{A=a}(R1) ] \bowtie R2$
- 需要估计中间结果  $U = \sigma_{A=a}(R1)$  的大小

$$T(U) = T(R1) / V(R1, A)$$

$$S(U) = S(R1)$$

$$V(U, *) = ?$$

# 1、中间结果的大小估计

Example

R1

$U = \sigma_{A=a}(R1)$

A	B	C	D
cat	1	10	10
cat	1	20	20
dog	1	30	10
dog	1	40	30
bat	1	50	10

$V(R,A)=3$

$V(R,B)=1$

$V(R,C)=5$

$V(R,D)=3$

$V(U,A) = 1$

$V(U,B) = 1$

$V(U,C) = T(U) = T(R1) / V(R1,A)$

$V(U,D) =$  介于 1 和  $T(R1) / V(R1,A)$  之间

# 1、中间结果的大小估计

■  $W = [ \sigma_{A=a}(R1) ] \bowtie R2$

●  $U = \sigma_{A=a}(R1)$

●  $V(U,*)$ 的一种可能的估计方法

◆  $V(U,A) = 1$

◆  $V(U,B) = V(R1,B)$

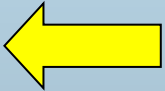
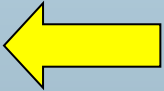


# 1、中间结果的大小估计

## ■ 其它情况的代价估计

- $\Pi_{AB}(R)$  : see Sec. 16.4.2
- $\sigma_{A=a \wedge B=b}(R)$ : see Sec. 16.4.3
- $R \bowtie S$  with multiple common attribs.  
: see Sec. 16.4.5
- Union, intersection, diff, .... :  
see Sec. 16.4.7

# Where are we?

- 语法分析(Parsing)
- 逻辑查询计划生成(Logical Query Plan)
- 查询重写(Query Rewrite)
- 查询计划代价估计(Cost Estimation)
  - 中间结果大小估计  We are here !
  - I/O代价估计  Next
- 物理查询计划选择(Physical Query Plan)

## 2、I/O代价估计

### ■ 影响查询计划I/O代价的因素

- 实现查询计划的逻辑操作符
  - ◆ 在选择逻辑查询计划时已确定
- 中间结果的大小 **already discussed!**
- 实现逻辑操作符的物理操作符 **下一章讨论**
  - ◆ 例如，连接操作是用索引连接还是散列连接？
- 相似操作的顺序 **see Sec.16.6**
  - ◆ 例如，多关系的连接顺序
- 物理操作符之间的参数传递方式 **see Sec.16.7**
  - ◆ **Pipeline**（流水线）还是**Materialization**（物化）？

## 2、I/O代价估计

### ■ 物理操作符之间的参数传递

#### ● 物化方式

- ◆ 操作依次执行，并且每个操作的结果（中间关系）都写到磁盘上供其它操作存取
- ◆ 通过磁盘物理进行数据传递
- ◆ 节省主存空间

#### ● 流水线

- ◆ 多个操作同时执行，一个操作产生的元组直接通过共享内存传递给其它操作
- ◆ 节省I/O
- ◆ 但占用主存，若缓冲区出现“颠簸”则I/O增加

# 五、物理查询计划选择

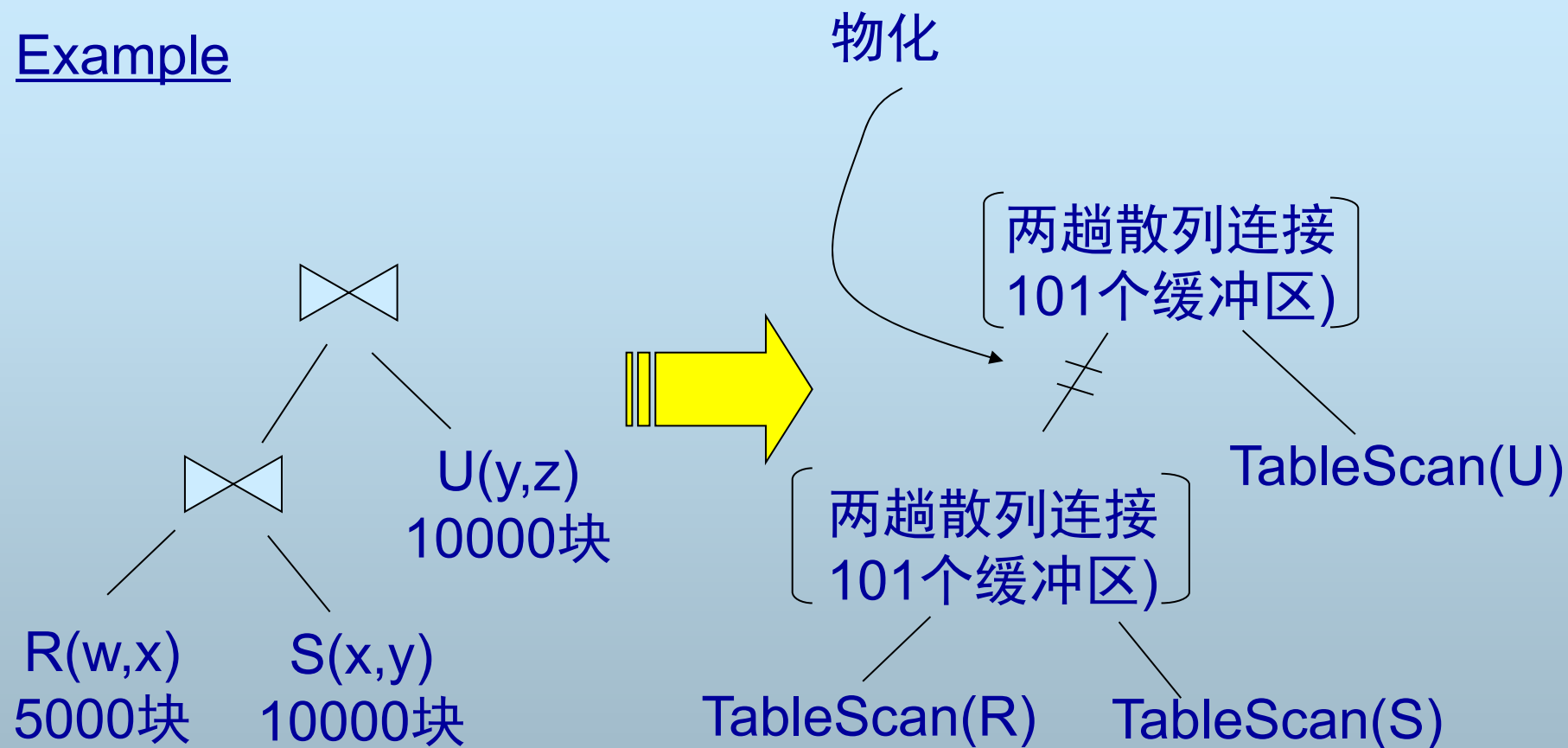
## ■ 物理查询计划生成

- 逻辑查询计划
- 估计中间关系大小
- 为逻辑操作符选择物理操作符
- 确定其它物理操作符
- 确定物理操作符的顺序
- 确定物理操作之间的参数传递方式

→ Next Chp.

## 五、物理查询计划选择

### Example



# 小结

- 语法分析(Parsing)
- 逻辑查询计划生成(Logical Query Plan)
- 查询重写(Query Rewrite)
- 查询计划代价估计(Cost Estimation)
- 物理查询计划选择(Physical Query Plan)