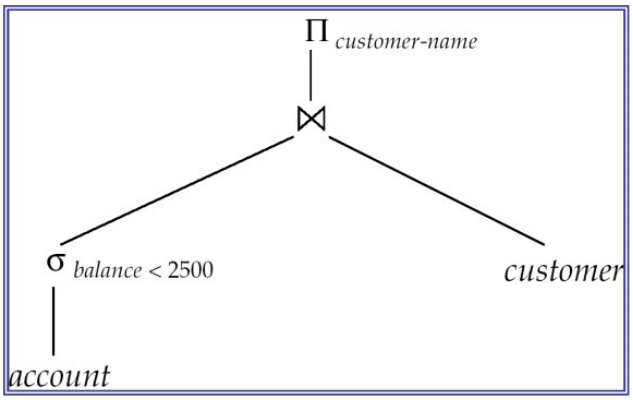
Query Introduction 1

⚠有关选择分支树的评估：

Materialization(具体化): generate results of an expression whose inputs are relations or relations that are already computed. Temporary relations must be materialized (stored) on disk.

[生成表达式的结果，其输入是已经计算过的关系。临时关系必须物化(存储)在磁盘上]



（1）具体化在各种情况下都可以实现，只不过可能需求较大的内存空间。

（2）使用双 buffer 机制，每次操作使用两个输出缓冲区，当一个缓冲区满时，将其写入另一个，这样操作允许磁盘写入与计算重叠，并减少执行时间。

Overall cost = Sum of costs of individual operations + cost of writing intermediate results to disk

Pipelining(流水化): pass on tuples to parent operations even as the operation is being executed.

[即使操作正在执行，也要将元组传递给父级操作]

（1）流水化并不能在各个情况下实现（比如 external merge-sort & hash-join），但节省储存空间。

（2）不操作，直接 join，随后直接将数组递交给 projection（不储存）。

（3）流水化分为两种方式：demand driven(请求驱动的) & producer driven(产生驱动的)。

About demand driven:

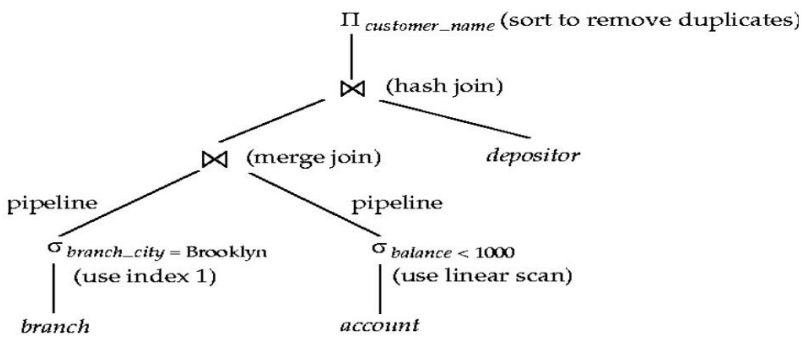
系统重复从顶层操作请求下一个 tuple

About producer driven:

操作符使用的 buffer 由 child tuples 替换 parent tuples，如果空间已满则等待删除。

⚠Evaluation Plan:

- If an edge in the evaluation tree is labelled with pipelining, then pipelining can be used.
- For join operations, unless stated explicitly, ONLY nested loop join can always use pipelining.



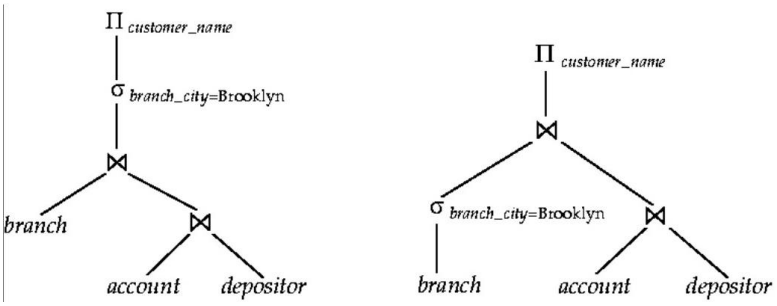
⚠Cost-based Query Optimization:

大的差距会出现在 seconds vs. days

随后则是可以调控操作的步骤而优化查询的时间

评估一般从关系、中间结果、算法成本的统计出发，估算不可避免会有特殊情况，但适合于绝大多数情况

比如如下两类操作就是同效的查询，但步骤不同（顺序不重要，但输出的元组相同则可以认为是等价的）：



⚠Equivalence Rules [等价规则]:

- Rule 1: Conjunctive selection operations can be deconstructed into a sequence of individual selections.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

- Rule 2: Selection operations are commutative.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

- Rule 3: Only the last one in a sequence of projection operations is needed, the others can be omitted.

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

- Rule 4: Selections can be combined with Cartesian products and theta joins.
  - (a). $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
  - (b). $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

- Rule 5: Theta-join operations (and natural joins) are commutative.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

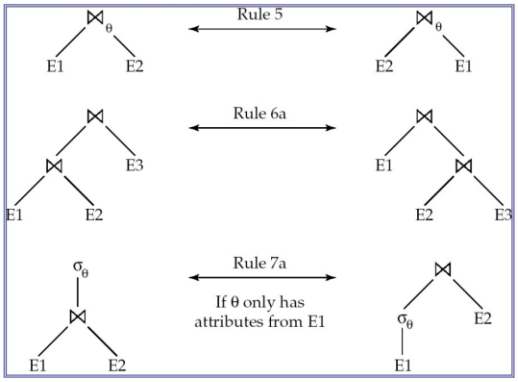
- Rule 6.
  - (a) Natural join operations are associative:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

  - (b) Theta joins are associative in the following manner:

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

    where θ_2 involves attributes from only E_2 and E_3.



- Rule 7. The selection operation distributes over the theta join operation under the following two conditions:
  - (a) When θ_0 involves only the attributes of one of the expressions (E_1) being joined.

$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$

  - (b) When θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2.

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$

- Rule 8. The projection operation distributes over the theta join operation as follows:
  - (a) Let $L1$ and $L2$ be attributes from $E1$ and $E2$, if θ involves only attributes from $L_1 \cup L_2$:

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

  - (b) Consider join $E_1 \bowtie_\theta E_2$.
    - let L_1 and L_2 be sets of attributes from E_1 and E_2, respectively.
    - let L_3 be attributes of E_1 that are involved in join condition θ, but are not in $L_1 \cup L_2$, and
    - let L_4 be attributes of E_2 that are involved in join condition θ, but are not in $L_1 \cup L_2$.

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\Pi_{L_2 \cup L_4}(E_2)))$$

- Rule 9. The set operations union and intersection are commutative (set difference is not commutative)

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

- Rule 10. Set union and intersection are associative.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

- Rule 11. The selection operation distributes over $\cup$, $\cap$ and $-$.

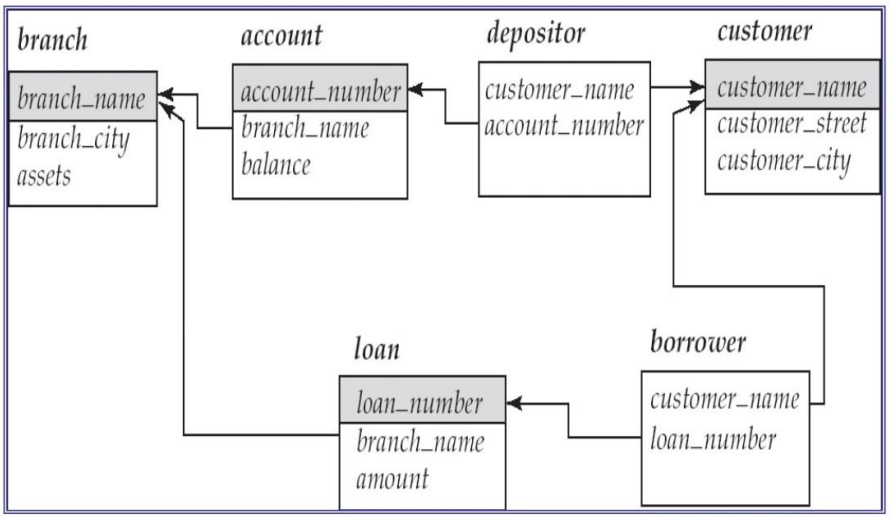
$$\sigma_\theta(E_1 \cup E_2) = \sigma_\theta(E_1) \cup \sigma_\theta(E_2)$$
$$\sigma_\theta(E_1 \cap E_2) = \sigma_\theta(E_1) \cap \sigma_\theta(E_2)$$
$$\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$$

- Rule 12. The projection operation distributes over union

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

⚠ **Example of Equivalence Rules [等价规则的示例]:**

Banking example



(1)

- Query: *Find the names of all customers who have an account at some branch located in Brooklyn.*

$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{"Brooklyn"}}(branch \bowtie (account \bowtie depositor)))$$

- Transformation using rule 7a (distribute the selection).

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"}}(branch)) \bowtie (account \bowtie depositor))$$

Tips: 尽早执行选择可以减少要连接的关系的大小

(2)

- Query: *Find the names of all customers with an account at a Brooklyn branch whose account balance is over $1000.*

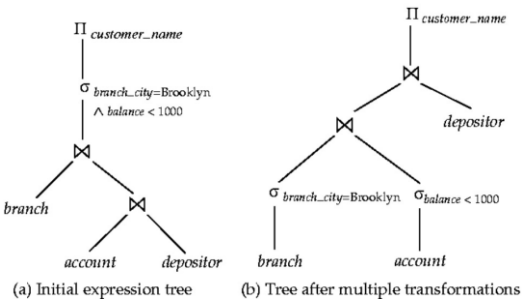
$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{"Brooklyn"} \wedge balance > 1000}(branch \bowtie (account \bowtie depositor)))$$

- Transformation using join associatively (Rule 6a and 7a):

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"} \wedge balance > 1000}(branch \bowtie account)) \bowtie depositor)$$

- Second form provides an opportunity to apply the "perform selections early" (Rule 7b)

$$\sigma_{branch\_city = \text{"Brooklyn"}}(branch) \bowtie \sigma_{balance > 1000}(account)$$



(a) Initial expression tree (b) Tree after multiple transformations

(3)

通过调控 join 的次序而控制空间消耗的成本:

- For all relations r_1, r_2, and r_3,

$$(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$$

(Join Associativity)

- If $r_2 \bowtie r_3$ is quite large and $r_1 \bowtie r_2$ is small, we may choose

$$(r_1 \bowtie r_2) \bowtie r_3$$

so that we compute and store a smaller temporary relation.

(4)

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"}}(branch) \bowtie account) \bowtie depositor)$$

- When we compute

$$(\sigma_{branch\_city = \text{"Brooklyn"}}(branch) \bowtie account)$$

we obtain a relation whose schema is:
(*branch_name, branch_city, assets, account_number, balance*)

- Push projections using equivalence rules 8b; eliminate unneeded attributes from intermediate results to get:

$$\Pi_{customer\_name}((\Pi_{account\_number}((\sigma_{branch\_city = \text{"Brooklyn"}}(branch) \bowtie account)) \bowtie depositor))$$

(HINT: L1 is null, L2 is customer_name; L3=L4=account_number)

(5)

- Consider the expression

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"}}(branch)) \bowtie (account \bowtie depositor))$$

- Could compute "*account* $\bowtie$ *depositor*" first, and join result with

$$\sigma_{branch\_city = \text{"Brooklyn"}}(branch)$$

but "*account* $\bowtie$ *depositor*" is likely to be a large relation.

- Only a small fraction of the customers are likely to have accounts in branches located in Brooklyn
  - it is better to compute first

$$\sigma_{branch\_city = \text{"Brooklyn"}}(branch) \bowtie account$$

▲ 使用 Query optimizers 利用等效规则对于算式进行枚举调整优化:

procedure genAllEquivalent(E)
$EQ = \{E\}$
repeat
    Match each expression E_i in EQ with each equivalence rule R_j
    if any subexpression e_i of E_i matches one side of R_j
        Create a new expression E' which is identical to E_i, except that
            e_i is transformed to match the other side of R_j
        Add E' to EQ if it is not already present in EQ
until no new expression can be added to EQ