# 数据库week11

## 15.2

> Consider the bank database of Figure 15.14, where the primary keys are underlined, and the following SQL query:

*branch*(*branch_name*, *branch_city*, *assets*)
*customer* (*customer_name*, *customer_street*, *customer_city*)
*loan* (*loan_number*, *branch_name*, *amount*)
*borrower* (*customer_name*, *loan_number*)
*account* (*account_number*, *branch_name*, *balance*)
*depositor* (*customer_name*, *account_number*)

**Figure 15.14** Bank database.

> **select** *T.branch_name*
>
> **from** *branch T*, *branch S*
>
> **where** *T.assets* > *S.assets* **and** *S.branch_city* = "Brooklyn"
>
> Write an efficient relational-algebra expression that is equivalent to this query. Justify your choice.

Expression:

$\Pi_{T.branch\_name}((\Pi_{branch\_name,assets}(\rho_T(branch))) \bowtie_{T.assets>S.assets} (\Pi_{assets}(\sigma_{branch\_city='Brooklyn'}(\rho_S(branch)))))$

By eliminating the unneeded attributes from both the operands, we can decrease the blocks we will transfer.

## 15.3

> Let relations $r_1$(*A*, *B*, *C*) and $r_2$(*C*, *D*, *E*) have the following properties: $r_1$ has 20,000 tuples, $r_2$ has 45,000 tuples, 25 tuples of $r_1$ fit on one block, and 30 tuples of $r_2$ fit on one block. Estimate the number of block transfers and seeks required using each of the following join strategies for $r_1 \bowtie r_2$:

$b_{r_1} = \frac{20000}{25} = 800$

$b_{r_2} = \frac{45000}{30} = 1500$

> a. Nested-loop join.

If $r_1$ is the outer relation, we need $n_{r_1} * b_{r_2} + b_{r_1} = 30,000,800$ block transfers and $n_{r_1} + b_{r_1} = 20800$ disk seeks.

If $r_2$ is the outer relation, we need $n_{r_2} * b_{r_1} + b_{r_2} = 36001500$ block transfers and $n_{r_2} + b_{r_2} = 46,500$ disk seeks.

> b. Block nested-loop join.

If $r_1$ is the outer relation, we need $\lceil \frac{b_{r_1}}{M-2} \rceil * b_{r_2} + b_{r_1}$ block transfers and $2 * \lceil \frac{b_{r_1}}{M-2} \rceil$ disk seeks.

If $r_2$ is the outer relation, we need $\lceil \frac{b_{r_2}}{M-2} \rceil * b_{r_1} + b_{r_2}$ block transfers and $2 * \lceil \frac{b_{r_2}}{M-2} \rceil$ disk seeks.

> c. Merge join.

- **Assuming that $r_1$ and $r_2$ are not initially sorted on the join key and $b_b = 1$**, the total sorting transfers cost inclusive of the output $T\_Sort$ is

  $T\_Sort(r_i) = b_{r_i}(2\lceil \log_{M-1}(b_{r_i}/M) \rceil + 1)$

  the total sorting transfers cost inclusive of the output $S\_Sort$ is

  $S\_Sort(r_i) = 2\lceil \frac{b_{r_i}}{M} \rceil + b_{r_i}(2\lceil \log_{M-1}(b_{r_i}/M) \rceil - 1)$

So we need $T\_Sort(r_1) + T\_Sort(r_2) + b_{r_1} + b_{r_2}$ block transfers and $S\_Sort(r_1) + S\_Sort(r_2) + b_{r_1} + b_{r_2}$ disk seeks.

- **If $r_1$ and $r_2$ are sorted on the join key and $b_b = 1$**

  So we need $b_{r_1} + b_{r_2}$ block transfers and $b_{r_1} + b_{r_2}$ disk seeks.

> d. Hash join.

Assuming that the $b_b = 1$.

- **If $r_1$ as probe input,$r_2$ as build input**

  If doesn't need recursive partitioning,we need $3(b_{r_1} + b_{r_2}) = 6900$ disk accesses and $2(b_{r_1} + b_{r_2}) = 4600$ seeks,else,we need $2(b_{r_1} + b_{r_2})\lceil \log_{M-1} \frac{b_{r_2}}{M} \rceil + b_{r_1} + b_{r_2}$ disk accesses and $2(b_{r_1} + b_{r_2})\lceil \log_{M-1} \frac{b_{r_2}}{M} \rceil$ seeks.

- **If $r_2$ as probe input,$r_1$ as build input**

  If doesn't need recursive partitioning,we need $3(b_{r_1} + b_{r_2}) = 6900$ disk accesses and $2(b_{r_1} + b_{r_2}) = 4600$ seeks,else,we need $2(b_{r_1} + b_{r_2})\lceil \log_{M-1} \frac{b_{r_1}}{M} \rceil + b_{r_1} + b_{r_2}$ disk accesses and $2(b_{r_1} + b_{r_2})\lceil \log_{M-1} \frac{b_{r_1}}{M} \rceil$ seeks.