**<u>Database Systems: The Complete Book</u>**

## ▼❑Chapter 16

### ▼❑Section 1

▼❑1

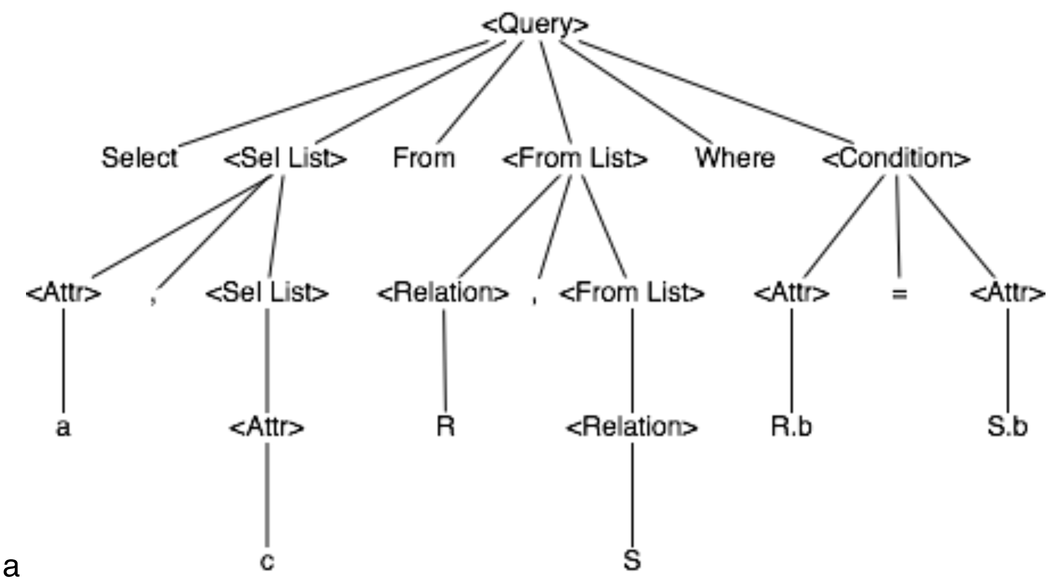
- ❑a <Query> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>
  <SelList> ::= DISTINCT <Attribute>
- ❑b <Query> ::= SELECT <SelList> FROM <FromList> WHERE <Condition> GROUP BY <GBList> HAVING <Condition>
  <GBList> ::= <Attribute> , <GBList>
  <GBList> ::= <Attribute>
- ❑c <Query> ::= SELECT <SelList> FROM <FromList> WHERE <Condition> ORDER BY <OBList>
  <OBList> ::= <Attribute> , <OBList>
  <OBList> ::= <Attribute>
- ❑d <Query> ::= SELECT <SelList> FROM <FromList>

▼❑2

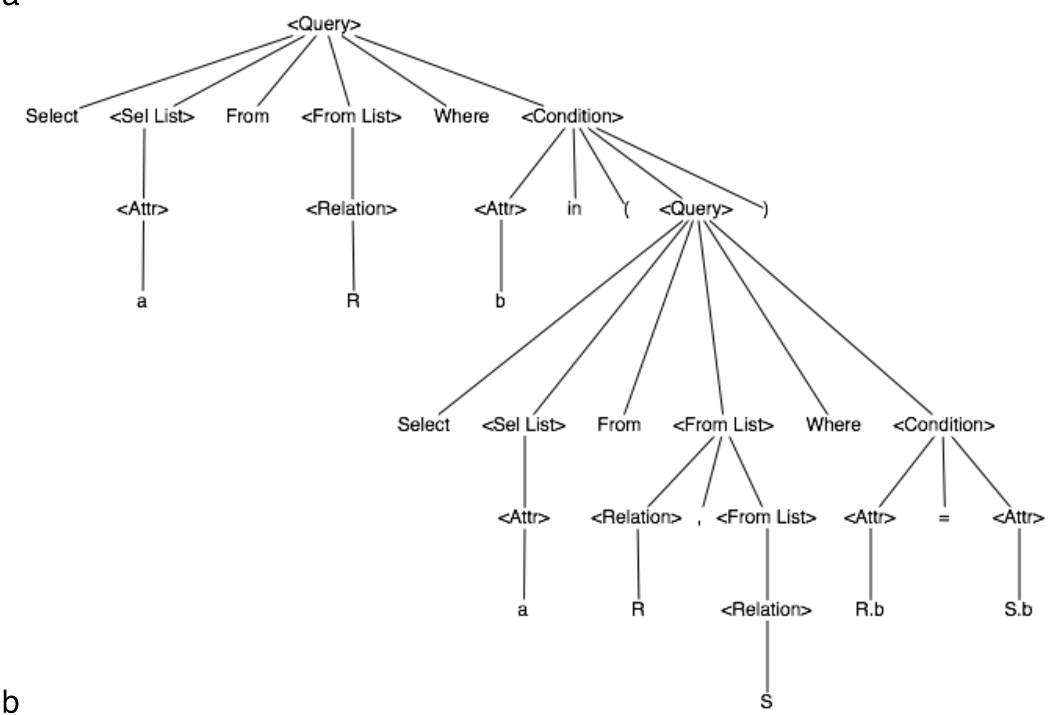
- ❑a <Condition> ::= <Condition> OR <Condition>
  <Condition> ::= NOT <Condition>
- ❑b <Condition> ::= <Attribute> > <Attribute>
  <Condition> ::= <Attribute> >= <Attribute>
  <Condition> ::= <Attribute> < <Attribute>
  <Condition> ::= <Attribute> <= <Attribute>
- ❑c <Condition> ::= ( <Condition> )
- ❑d <Condition> ::= EXISTS ( <Query> )

▼❑3

- ❑a




- ❑b




## ❑Section 2

- ❑1 $\sigma_c(R \cap S)$ and there is an index on S.   Assuming that there C attributes in both R and S, the options are:
  $\sigma_c(R \cap S)$ -- Larger intermediate set to select, likely to scan more data doing intersect, then selection on the intermediate result (no index available).
  $\sigma_c(R) \cap \sigma_c(S)$ -- Smaller set to intersect.

### ❑2

- ❑a $\pi_L(R \cup S) \neq \pi_L(R) \cup \pi_L(S)$
  R(C1, C2) = {(1,1) (1,2) (1,2)}
  S(C1, C2) = {(1,3) (1,4) (1,5)}

$L = C1$

- ❑b $\pi_L(R-S) \neq \pi_L(R)-\pi_L(S)$
  $R(C1, C2) = \{(1,1)\ (1,2)\ (1,5)\}$
  $S(C1, C2) = \{(1,1)\ (1,3)\ (1,4)\}$
  $L = C1$
- ❑c $\delta(\pi_L(R) \neq \pi_L(\delta(R)$
  $R(C1, C2) = \{(1,1)\ (1,2)\ (2,3)\ (3,4)\ (1,1)\}$
  $L = C1$
- ❑d $\delta(R \cup_B S) \neq \delta(R) \cup_B \delta(S)$
  $R(C1, C2) = \{(1,1)\ (1,2)\}$
  $S(C1, C2) = \{(1,1)\ (1,3)\}$

- ❑3 $\pi_L(R \cup_B S) = \pi_L(R) \cup_B \pi_L(S)$
  Attributes eliminated do not appear in results. Attributes eliminated are not used in operators above, and their absence does not affect the results (by definition of bag union).

▼❑4

- ❑a Counterexample: $R(C1) = \{1\}$
- ❑b Every record in R has a corresponding record in R. The intersection is the same.
- ❑c Same as 16.2.4b
- ❑d Adding the elements from R before or after $\cap_B$ doesn't matter. If applied to both S and T, they will appear in both intermediate sets, and appear in the results.

▼❑5

- ❑a If $R \subseteq S$, then $R \cup S = S$
  True. Every R element has a corresponding S element that appear at least more times than the R element. Thus, the U operator does not add anything to the S set.
- ❑b If $R \subseteq S$, then $R \cap S = R$
  True. Every R element has a corresponding S element that appear at least as many times as the R element. Any excess element will not be included in the intersection, leaving only R.
- ❑c If $R \subseteq S$ and $S \subseteq R$, then $R = S$
  True. Every R element has a corresponding S element that appears as many times as the R element, and vice versa. This leaves the only possibility that the sets are equal.

▼❑6

- ❑a $\pi_L(\pi_{b,c}(R)$ JOIN $\pi_{b,c,d}(S))$
- ❑b $\pi_L(R$ JOIN $\pi_{b,c,d}(S))$

- ❑7 Push the aggregation before the join.

▼❑8

- ❑a Counterexample: R(a,b) = {(1,50) (1,50) (2,100) (2,100)}
- ❑b Counterexample: R(a,b) = {(1,50) (1,50) (2,100) (2,100) (3,100) (3,150) (4,100)}

▼❑9

- ❑a True. Any records eliminated by the selection can be done before or after the semi-join. Doing it before will reduce the intermediate result.
- ❑b Counterexample:
  R(C1,C2) = {(1,2) (1,3)}
  S(C1,C3) = {(1,2) (2,3)}
- ❑c True. The record selected could be:
  1. Dangling: Will still be dangling if selection is pushed down.
  2. Matched. Will still be a match if selection is pushed down.
  Conditions for match or dangle haven't changed.
- ❑d Counterexample:
  R(C1,C2) = {(1,2) (1,3) (3,3)}
  S(C1,C3) = {(1,2) (2,2)}
- ❑e Counterexample:
  R(C1,C2) = {(1,2) (1,3) (2,2)}
  S(C1,C3) = {(1,2) (2,2)}
- ❑f True. Full outer join is associative. No data is lost.
- ❑g True. Full outer join is commutative. No data is lost.
- ❑h Counterexample:
  R(C1,C2) = {(1,2) (1,3) (2,2)}
  S(C1,C3) = {(1,3)}
- ❑i Counterexample:
  R(C1,C2) = {(1,2) (2,2)}
  S(C1,C3) = {(1,3)}

- ❑10 The SUM function will skip over null values. Let $a_k$ = null.
  SUM($a_1,a_2,...,a_n$). $a_1 + a_2 + ... + a_n$ = null. The law doesn't hold.

▼❑**Section 3**

▼❑1

- ❑a ( ($\pi_{a,b,c}$( R(a,b) JOIN R.b = S.b S(b,c)) ) JOIN $_{S.c = T.c}$ T(c,d) )
- ❑b $\pi_{a,b,c,d,e}$( ($\pi_{a,b,c \to x}$( R(a,b) JOIN $_{R.b = S.b}$ S(b,c) ) ) JOIN $_{x.c = y.c}$ ( $\pi_{c,d,e \to y}$( T(c,d) JOIN $_{T.d = U.d}$ U(d,e) ) ) )
- ❑c $\pi_{a,b,c,d}$( ($\pi_{a,b,c \to x}$( R(a,b) JOIN $_{R.b = S.b}$ S(b,c) ) ) JOIN $_{x.a = y.a}$ ( $\pi_{a,c,d \to y}$( T(c,d) JOIN $_{T.d = U.d}$ U(a,d) ) ) )

▼❑2

- ❑a $\pi_{a,c}$( R JOIN S)
- ❑b $\pi_a$( σ( R(1), (<Attr> IN $\pi_a$( R(2) JOIN S ) ) )
  $\pi_a$( R JOIN (δ ($\pi_a$( R(2) JOIN S ) ))

▼❑3

    - ❑a $\pi_R(\sigma_{count}(\text{ R CROSS JOIN } \gamma_{count}(<\text{Query}>)))$
    - ❑b $\pi_R(\text{ R JOIN } \delta(\pi_a(<\text{Query}>)) )$
    - ❑c $\sigma_{count(a)=1}(\text{ R JOIN } \gamma(\delta(\pi_a(<\text{Query}>))))$
  - ▼❑4
    - ❑a $\pi_R(\sigma_{count>0}(\gamma_{count}(\text{R CROSS JOIN } <\text{Query}>)))$
    - ❑b $\pi_{a,b,c}(\delta(\pi_a(\text{R JOIN } <\text{Query}>)))$
    - ❑c $\sigma_{count(a)=1}(\delta(\pi_a(\text{R JOIN } <\text{Query}>)))$
  - ❑5 4! x 3! x 4! x 3!
- ▼❑**Section 4**
  - ▼❑1
    - ❑a 8000
    - ❑b 5
    - ❑c 6
    - ❑d 48
    - ❑e 30000
    - ❑f 133
    - ❑g 0
    - ❑h 2
    - ❑i 20000
  - ❑2 0
  - ❑3 T(R)/V(S)
  - ❑4 V(R,a) = 100.    V(S,a) = 100.
- ▼❑**Section 5**
  - ❑1 245.    This is a more accurate estimate.
  - ❑2 56842
  - ▼❑3
    - ❑a Better than (b) as long as:
      50000/max(V(R,b),200) > 100000/max(V(R,b),200)
    - ❑b Same as (a)
  - ▼❑4
    - ❑a 832
    - ❑b 12
    - ❑c R,S,T,V
    - ❑d R,S,T,V
  - ▼❑5
    - ❑a 700
    - ❑b 100
    - ❑c R,S,T,V

- ❑d R,S,T,V

| | Order | | Order |
|---|---|---|---|
| R SMJ S | b | SMJ T | c |
| | | HASH T | none |
| R HASH S | none | SMJ T | c |
| | | HASH T | none |
| R SMJ T | c | | |
| | | | |
| S HASH T | none | SMJ R | b |
| | | HASH R | none |
| S SMJ R | b | SMJ T | d |
| | | HASH T | none |
| S HASH R | none | SMJ T | d |
| | | HASH T | none |
| T SMJ S | c | | |
| | | | |
| T HASH S | none | SMJ R | |
| | | HASH R | |

- ❑6 The query plan that uses the fewest I/O's is highlighted in yellow.
- ❑7 The best plan for E or F may not provide sorted output, but the join method selected (e.g. SMJ) may required sorted input.

## ❑Section 6

- ❑1

| | W | X | Y | Z | | |
|---|---|---|---|---|---|---|
| **SIZE** | 100 | 200 | 300 | 400 | | |
| **COST** | 0 | 0 | 0 | 0 | | |
| **BEST** | W | X | Y | Z | | |
| | | | | | | |
| | WX | WY | WZ | XY | XZ | YZ |
| **SIZE** | 333 | 30000 | 40000 | 30000 | 80000 | 2400 |
| **COST** | 0 | 0 | 0 | 0 | 0 | 0 |
| **BEST** | WX | WY | WZ | XY | XZ | YZ |
| | | | | | | |
| | WXY | WXZ | XYZ | YZW | | |
| **SIZE** | 1000 | 133333 | 4800 | 240000 | | |
| **COST** | 333 | 333 | 2400 | 2400 | | |
| **BEST** | WXY | WXZ | YZX | YZW | | |

- ❑2

| | W | X | Y | Z | | |
|---|---|---|---|---|---|---|
| **SIZE** | 100 | 200 | 300 | 400 | | |
| **COST** | 0 | 0 | 0 | 0 | | |
| **BEST** | W | X | Y | Z | | |
| | | | | | | |
| | WX | WY | WZ | XY | XZ | YZ |
| **SIZE** | 333 | 30000 | 400 | 30000 | 80000 | 2400 |
| **COST** | 0 | 0 | 0 | 0 | 0 | 0 |
| **BEST** | WX | WY | WZ | XY | XZ | YZ |
| | | | | | | |
| | WXY | WXZ | XYZ | YZW | | |
| **SIZE** | 1000 | 1333 | 4800 | 1600 | | |
| **COST** | 333 | 333 | 2400 | 400 | | |
| **BEST** | WXY | WXZ | YZX | YZW | | |

- ❑3

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | E | F | G | H | | | |
| **SIZE** | 1000 | 2000 | 3000 | 4000 | | | |
| **COST** | 0 | 0 | 0 | 0 | | | |
| **BEST** | E | F | G | H | | | |
| | | | | | | | |
| | EF | EG | EH | FG | FH | GH | |
| **SIZE** | 20 | 10 | 800 | 240 | 200 | 80 | |
| **COST** | 0 | 0 | 0 | 0 | 0 | 0 | |
| **BEST** | EF | EG | EH | FG | FH | GH | |
| | | | | | | | |
| | EFG | EFH | FGH | GHE | | | |
| **SIZE** | 1 | 1 | 1 | 1 | | | |
| **COST** | 10 | 20 | 80 | 10 | | | |
| **BEST** | EGF | EFH | GHF | EGH | | | |
| | | | | | | | |
| | (((EG)F)H) | (((EF)H)G) | (((GH)F)E) | (((EG)H)F) | ((EF)(GH)) | ((EG)(FH)) | ((EH)(FG)) |
| **SIZE** | 11 | 21 | 81 | 11 | 100 | 210 | 1040 |

- ▼❑4
  - ❑a ((SR)T)U). Cost: 10000.
  - ❑b Optimal: ((RS)(TU)). Cost: 2000.
- ▼❑5
  - ❑a 7! x (T(1)T(7) + T(2)T(6) + T(3)T(5) + T(4)T(4) + T(5)T(3) + T(6)T(2) + T(7)T(1)
    Left-deep: 7!
    Right-deep: 7!
  - ❑b 8! x (T(1)T(8) + T(2)T(7) + T(3)T(6) + T(4)T(5) + T(5)T(4) + T(6)T(3) + T(7)T(2) + T(8)T(1)
    Left-deep: 8!
    Right-deep: 8!
- ▼❑6
  - ❑a B(R JOIN S) + B((R JOIN S) JOIN U)
  - ❑b B(R JOIN S) + B(T JOIN U)
  - ❑c B(T JOIN U) + B(S JOIN (T JOIN U))
- ❑7 Left deep only: k!
  All: $2^k$ x k!

## ▼❑Section 7

- ▼❑1
  - ❑a access method: index on b
  - ❑b access method: index on b
  - ❑c access method: index on a
- ▼❑2
  - ❑a T(R)/V(R,x) < T(R)/V(R,y)
  - ❑b R(T)/V(R,x) < B(R)/V(R,y)
  - ❑c T(R)/V(R,x) < B(R)/3
- ▼❑3

- ❑a 100 buckets.  2 blocks per bucket.  Pipelining OK.
- ❑b 100 buckets.  100 blocks per bucket.  Pipelining not possible.
- ❑c 2 buckets.  50 blocks per bucket.  Pipelining OK.

▼❑4

- ❑a B(T) <= 99
- ❑b B(T) >= 100