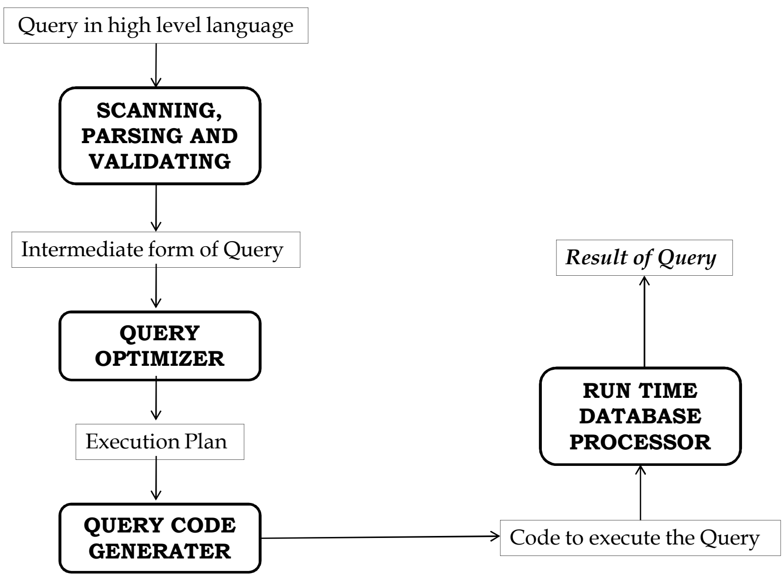
Query Optimization

Swarup Kr Ghosh

Introduction



Query Optimization

- ✓ **What is Query Optimization ?**
- ✓ **What is the utility of Query Optimization?**
- ✓ **Query Optimization is based on what ?**

❖*Relational Algebra*

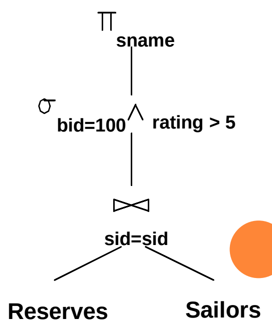
(Select (±), Project (ē), Cartesian Product (X), Join (⋈) etc)

Query Optimization Overview

- Query can be converted to relational algebra
- Rel. Algebra converted to tree, joins as branches
- Each operator has implementation choices
- Operators can also be applied in different order!

```
SELECT  S.sname
FROM  Reserves R, Sailors S
WHERE  R.sid=S.sid AND
   R.bid=100 AND S.rating>5
```

$\pi_{(sname)}\sigma_{(bid=100 \wedge rating > 5)}$ (Reserves $\bowtie$ Sailors) $\bowtie$



Query Optimization Technique

- ✓ ***Heuristics Technique in Query Optimization.***
- ✓ ***Cost Estimates Technique in Query Optimization.***

Heuristics Technique of Query Optimization

- ➢ **One of the main heuristic rules is to apply SELECT and PROJECT operations before applying the JOIN or other binary operations.**
- ➢ **Generating equivalent *Query Tree* and *Query Graph*.**

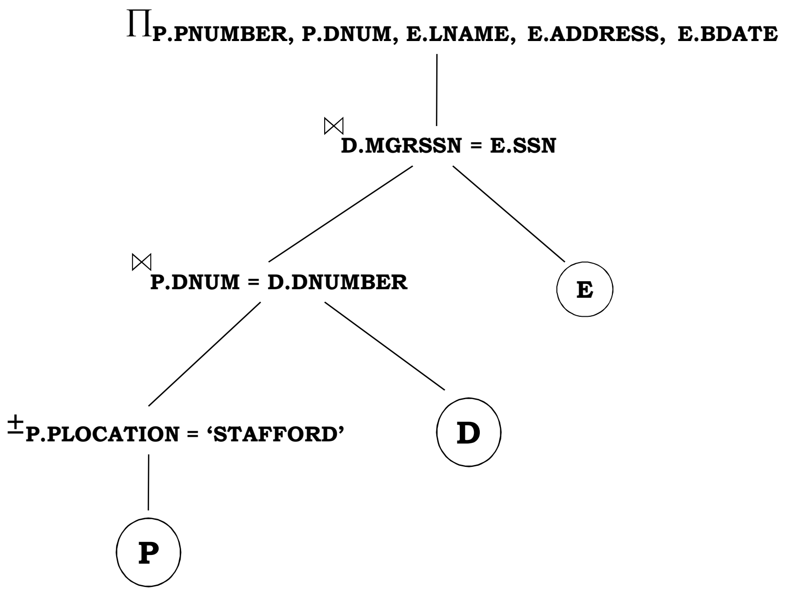
Example :

SQL Query: **SELECT P.PNUMBER, D.DNUM, E.LNAME, E.ADDRESS, E.BDATE**
FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E
WHERE D.MGRSSN = E.SSN AND P.DNUM = D.DUMBER
AND P.PLOCATION = 'STAFFORD'

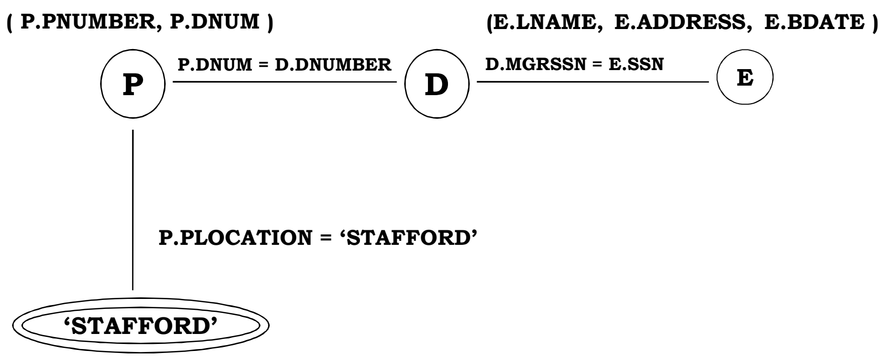
Relational Algebra:

ē $_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}$ (± $_{\text{PLOCATION = 'STAFFORD'}}$ **(PROJECT)**
$\bowtie_{\text{DNUM = DNUMBER}}$ **(DEPARTMENT)** $\bowtie_{\text{MGRSSN = SSN}}$ **(EMPLOYEE))**

Query Tree



Query Graph



General Transformation Rules for Relational Algebra Operations

1. ***Cascade of σ*** **:** A conjunctive selection condition can be broken up into a cascade (that is a sequence) of individual σ operations:

$$\sigma_{c1 \text{ and } c2 \text{ and } \ldots\ldots\ cn}\ (R) \equiv \sigma_{c1}(\sigma_{c2}\ (\ldots\ldots(\sigma_{cn}\ (R))\ldots\ldots))$$

2. ***Commutative of σ:*** The **σ** operation is commutative.

$$\sigma_{c1}\ (\sigma_{c2}\ (R)) \equiv \sigma_{c2}\ (\sigma_{c1}\ (R))$$

3. ***Cascade of Π*** *:* In a cascade (sequence) of **Π** operations, all but the last one can be ignored:

$$\Pi_{list1}\ (\Pi_{list2}\ (\ldots\ldots.(\Pi_{listn}\ (R))\ \ldots\ldots\ldots)) \equiv \Pi_{list1}\ (R)$$

4. ***Commuting σ with Π****:* If the selection condition c involves only those attributes A1, ….., An in the projection list, the two operations can be commuted.

$$\Pi_{A1,\ A2,\ \ldots\ldots.\ ,\ An}\ (\sigma_c\ (R)) \equiv \sigma_c\ (\Pi_{A1,\ A2,\ \ldots\ldots.\ ,\ An}(R))$$

General Transformation Rules for Relational Algebra Operations (Cont.)

5. ***Commutative of ⋈ (and X) :*** The ⋈ operation is commutative , as is the **X** operations:

$$R \bowtie_c S \equiv S \bowtie_c R$$

$$R \times S \equiv S \times R$$

6. ***Commutative σ with ⋈ (or X):*** If all the attributes in the selection condition *c* involve only the attributes of one of the relations being joined — say, *R* — the two operations can be commuted as follows:

$$\sigma_c\ (R \bowtie S) \equiv (\sigma_c\ (R)\ ) \bowtie S$$

Alternatively, if the selection condition *c* can be written as (*c*1 AND *c*2), where condition *c*1 involves only the attributes of *R* and condition *c*2 involves only the attributes of *S,* the operations commute as follows:

$$\sigma_c\ (R \bowtie S) \equiv (\sigma_{c1}\ (R)) \bowtie (\sigma_{c2}\ (S))$$

The same rules apply if the is replaced by a X operation

General Transformation Rules for Relational Algebra Operations (Cont.)

*7. **Commuting Π with ⋈ (or X):*** Suppose that the projection list is $L = \{ A_1, \ldots\ldots\ An, B_1, \ldots\ldots\ Bm \}$, where $A_1, \ldots, An$ are attributes of *R* and $B_1, \ldots, Bm$ are attributes of *S*. If the join condition *c* involves only attributes in *L*, the two operations can be commuted as follows:

$$\Pi_L (R \bowtie_c S) \equiv (\Pi_{A1, \ldots\ldots An}(R)) \bowtie_c (\Pi_{B1, \ldots\ldots, Bm} (S))$$

If the join condition *c* contains additional attributes not in *L*, these must be added to the projection list, and a final **Π** operation is needed. For example, if attributes $A_{n+1}, \ldots\ldots, A_{n+k}$ of *R* and $B_{m+1}, \ldots\ldots, B_{m+p}$ of *S* are involved in the join condition *c* but are not in the projection list *L*, the operations commute as follows:

$$\Pi_L(R \bowtie_c S) \equiv \Pi_L(\Pi_{A1,\ldots An, An+1,\ldots,An+k}(R)) \bowtie_c (\Pi_{B1,\ldots,Bm, Bm+1,\ldots\ldots,Bm+p}(S)).$$

For **X**, there is no condition *c*, so the first transformation rule always applies by replacing *c* with **X**.

*8. **Commutativity of set operations***: The set operations ∩ and **U** are commutative but **-** is not.

General Transformation Rules for Relational Algebra Operations (Cont.)

*9. **Associativity of ⋈ , X, U, and ∩ :*** These four operations are individually associative; that is, if **θ** stands for any one of these four operations (throughout the expression), we have:

$$(R\ \theta\ S)\ \theta\ T \equiv R\ \theta\ (S\ \theta\ T)$$

*10. **Commuting σ with set operations:*** The **σ** operation commutes with **U**, **∩**, and **-**. If **θ** stands for any one of these three operations (throughout the expression), we have:

$$\sigma_c (R\ \theta\ S) \equiv (\sigma_c (R))\ \theta\ (\sigma_c (S))$$

*11. **The Π operation commutes with U:***

$$\Pi_L (R \cup S) \equiv (\Pi_L (R)) \cup (\Pi_L (S))$$

*12. **Converting a (σ, X) sequence into ⋈ :*** If the condition *c* of a **σ** that follows a **X** corresponds to a join condition, convert the (**σ**, X) sequence into a as follows:

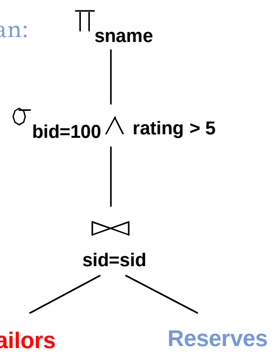
$$(\sigma_c (R \times S)) \equiv (R \bowtie_c S)$$

MOTIVATING EXAMPLE

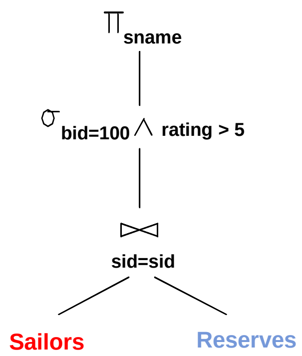
```
SELECT  S.sname
FROM  Reserves R, Sailors S
WHERE  R.sid=S.sid AND
   R.bid=100 AND S.rating>5
```

- Cost: ?
- By no means the worst plan!
- Misses several opportunities: selections could have been `pushed' earlier, no use is made of any available indexes, etc.
- *Goal of optimization:* To find more efficient plans that compute the same answer.

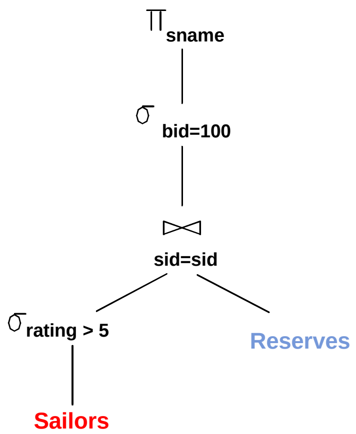
Plan:



ALTERNATIVE PLANS

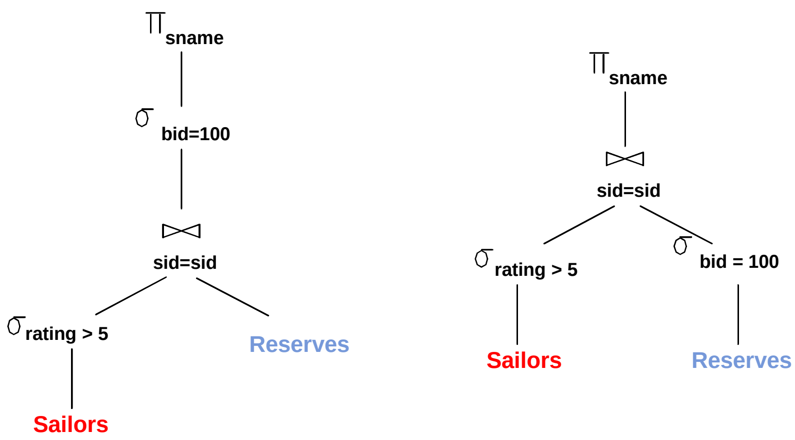


More operations



Less operations

Alternative Plans



Example :

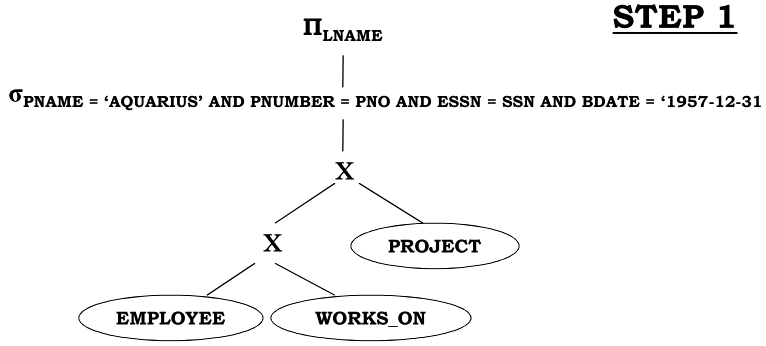
SQL Query:

SELECT **LNAME**
FROM **EMPLOYEE, WORKS_ON, PROJECT**
WHERE **PNAME = 'AQUARIUS' AND PNUMBER = PNO AND ESSN = SSN AND BDATE = '1957-12-31';**

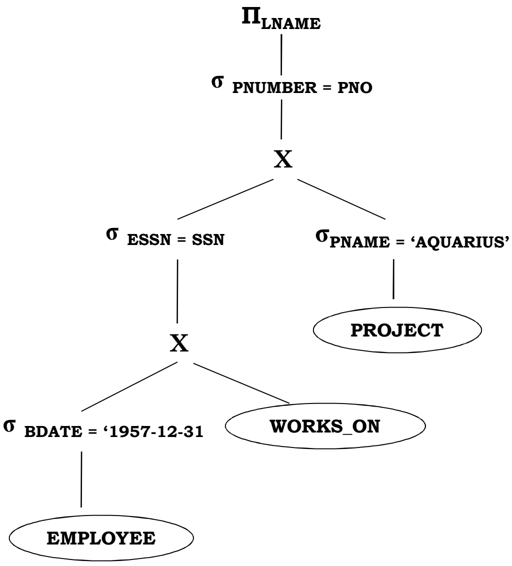
Relational Algebra:

Π_{LNAME} ($\sigma_{PNAME\ =\ 'AQUARIUS'\ AND\ PNUMBER\ =\ PNO\ AND\ ESSN\ =\ SSN\ AND\ BDATE\ =\ '1957-12-31'}$ **(EMPLOYEE X WORKS_ON X PROJECT))**

STEP 1



STEP 2



Using Rules:

Rule 1:

$\sigma_{c1 and ...cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(..(\sigma_{cn}(R)) ..))$

Rule 2:

$\sigma_{c1} (\sigma_{c2} (R)) \equiv \sigma_{c2} (\sigma_{c1} (R))$

Rule 4:

$\Pi_{A1,....,An}(\sigma_c (R)) \equiv \sigma_c(\Pi_{A1,...,An}(R))$

Rule 6:

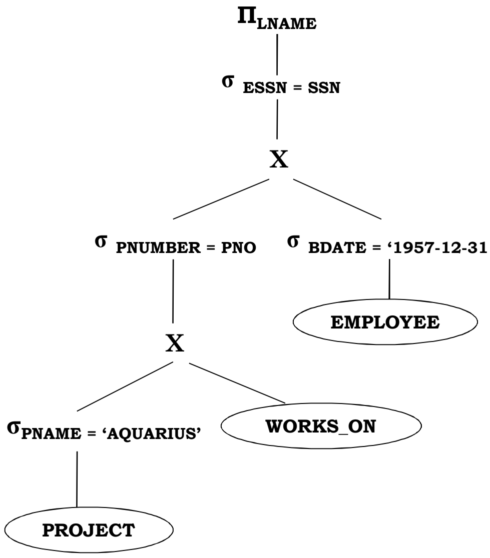
$\sigma_c (R \text{ X } S) \equiv (\sigma_c (R)) \text{ X } S$

$\sigma_c (R \text{ X } S) \equiv (\sigma_{c1} (R)) \text{ X } (\sigma_{c2}(S))$

Rule 10:

$\sigma_c (R \,\theta\, S) \equiv (\sigma_c (R)) \,\theta\, (\sigma_c (S))$

STEP 3



Using Rules:

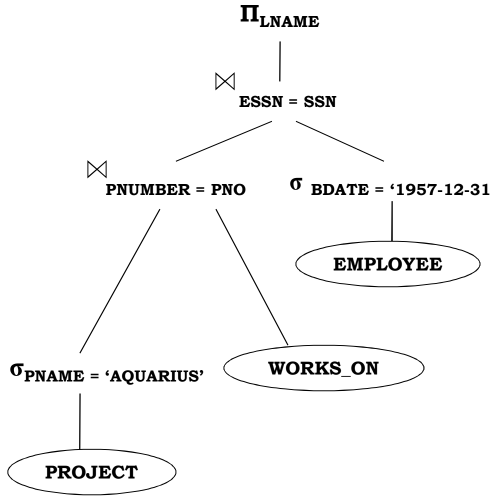
Rule 5:

R X S ≡ S X R

Rule 9:

$(R \,\theta\, S) \,\theta\, T \equiv R \,\theta\, (S \,\theta\, T)$

STEP 4

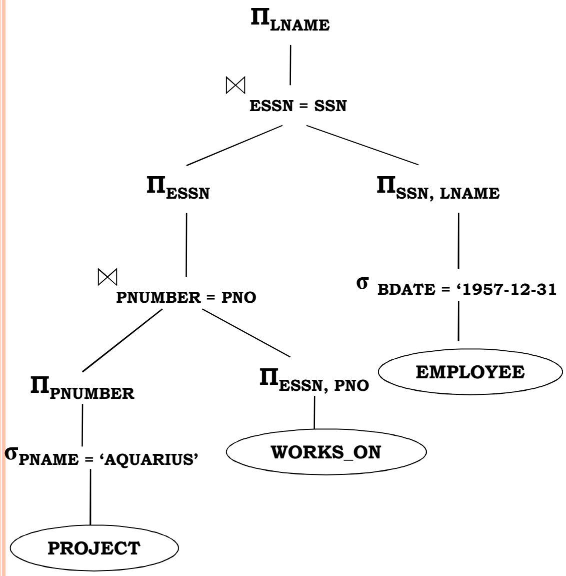


Using Rules:

Rule 12:

$(\sigma_c\ (R \times S)) \equiv (R \bowtie_c S)$

STEP 5



Using Rules:

Rule 3:

$\Pi_{list1}(\Pi_{list2}(..(\Pi_{listn}(R))..)) \equiv \Pi_{list1}\ (R)$

Rule 4:

$\Pi_{A1,....,An}(\sigma_c\ (R)) \equiv \sigma_c(\Pi_{A1,....,An}(R))$

Rule 7:

$\Pi_L\ (R \bowtie_c S) \equiv (\Pi_{A1,\ldots An}\ (R)) \bowtie_c (\Pi_{B1,\ldots,Bm}(S))$

Rule 11:

$\Pi_L\ (R \cup S) \equiv (\Pi_L\ (R)) \cup (\Pi_L\ (S))$

Selectivity and Cost Estimates in Query Optimization

Cost Component for Query Execution

- ✓ **Access cost to secondary storage**
- ✓ **Storage Cost**
- ✓ **Computation Cost**
- ✓ **Memory usage Cost**
- ✓ **Communication Cost**

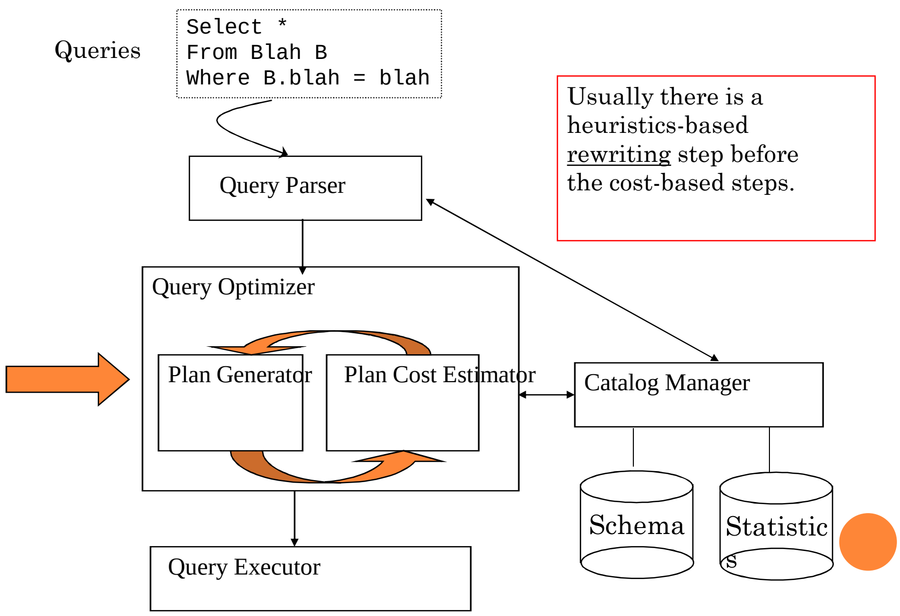
NB: **Generally we use *Access Cost to Secondary Storage* mainly and use *Communication Cost* for Distributed Database only.**

Catalog Information Used in Cost Function

- **Number of Records (tuples) *(r).***
- **Record Size *(R)***
- **Number of Blocks *(b)***
- **Blocking factor *(bfr)***
- **Number of Levels *(x)***
- **Number of first-level index blocks *(b_{I1})***
- **Number of Distinct values *(d)***
- **Selectivity *(sl)***
- **Selection Cardinality *(s = sl * r)***

NB: ***For a key attribute, d = r, sl = 1/r and s = 1. For a nonkey attribute, by making an assumption that the d distinct values are uniformly distributed among the records, we estimate sl = (1/d) and so s = (r/d)***

COST-BASED QUERY SUB-SYSTEM



Schema for Examples

Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

- Reserves:
  - Each tuple is 40 bytes long, 100 tuples per page, 1000 pages.
  - Assume there are 100 boats
- Sailors:
  - Each tuple is 50 bytes long, 80 tuples per page, 500 pages.
  - Assume there are 10 different ratings

Cost Functions for SELECT

- ***Linear Search (Brute force) Approach.***

$$C_{s1a} = b$$
$$C_{s1b} = (b/2)$$

- ***Binary Search.***

$$C_{s2} = \log_2 b + (s/bfr) - 1$$

- ***Using a primary index or hash key to retrieve a single record.***

$$C_{s3a} = x + 1 \textit{ (for Primary index)}$$
$$C_{s3b} = 1 \textit{ (for hash key)}$$

- ***Using an ordering index to retrieve multiple records.***

$$C_{s4} = x + (b/2)$$

- ***Using a clustering index to retrieve multiple records.***

$$C_{s5} = x + (s/bfr)$$

Example to Illustrate Cost Based Query Optimization

SQL Query: **SELECT P.PNUMBER, D.DNUM, E.LNAME, E.ADDRESS, E.BDATE**
FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E
WHERE D.MGRSSN = E.SSN AND P.DNUM = D.DUMBER
AND P.PLOCATION = 'STAFFORD'

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE
PROJECT	PLOCATION	200	1	200
PROJECT	PNUMBER	2000	1	2000
PROJECT	DNUM	50	1	50
DEPARTMENT	DNUMBER	50	1	50
DEPARTMENT	MGRSSN	50	1	50
EMPLOYEE	SSN	10000	1	10000
EMPLOYEE	DNO	50	1	50
EMPLOYEE	SALARY	500	1	500

Example to Illustrate Cost Based Query Optimization (Cont.)

TABLE_NAME	NUM_ROWS	BLOCKS
PROJECT	2000	100
DEPARTMENT	50	5
EMPLOYEE	10000	2000

INDEX_NAME	UNIQUENES	BLEVEL	LEAF_BLOCKS	DISTINCT_KEYS
PROJ_PLOC	NONUNIQUE	1	4	200
EMP_SSN	UNIQUE	1	50	10000
EMP_SAL	NONUNIQUE	1	50	500

PROJECT ⋈ DEPARTMENT ⋈ EMPLOYEE
DEPARTMENT ⋈ PROJECT ⋈ EMPLOYEE
DEPARTMENT ⋈ EMPLOYEE ⋈ PROJECT
EMPLOYEE ⋈ DEPARTMENT ⋈ PROJECT

THANK YOU