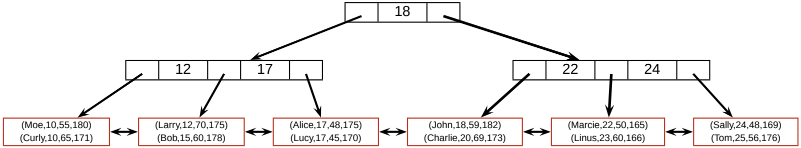
CS5226 Lecture 4
Index Tuning

Index Tuning

- **Input**:
  - a workload $W = \{Q_1, \cdots, Q_n\}$, each Q_i is a query
  - a storage bound B
- **Output**: An index configuration $C = \{I_1, \cdots, I_k\}$ such that
  - $\sum_{j=1}^{k} size(I_j) \leq B$, and
  - $\sum_{j=1}^{n} cost(Q_j, C)$ is minimized
- $size(I)$ = size of an index
- $cost(Q, C)$ = execution cost of the optimal plan for Q wrt C

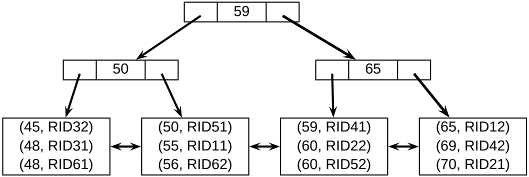
B$^+$-tree index



Clustered index on (age)

Relation R

name	age	weight	height
Moe	10	55	180
Curly	10	65	171
Larry	12	70	175
Bob	15	60	178
Alice	17	48	175
Lucy	17	45	170
John	18	59	182
Charlie	20	69	173
Marcie	22	50	165
Linus	23	60	166
Sally	24	48	169
Tom	25	56	176



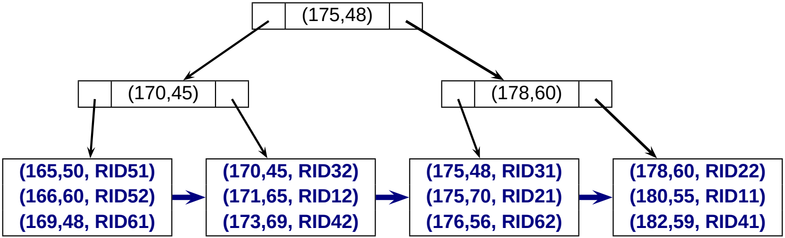
Unclustered index on (weight)

Index access methods

- Index scan
- Index seek [+ RID lookup]
- Index intersection [+ RID lookup]

Index scan

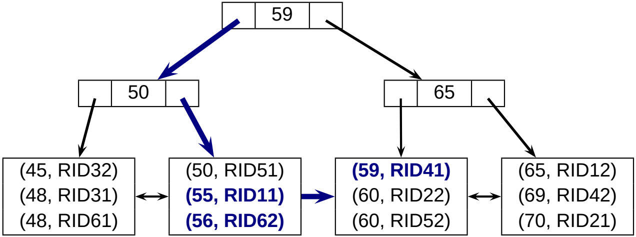
select height
from Student



Index on (height,weight)

Index seek

select weight
from Student
where weight **between** 55 **and** 59

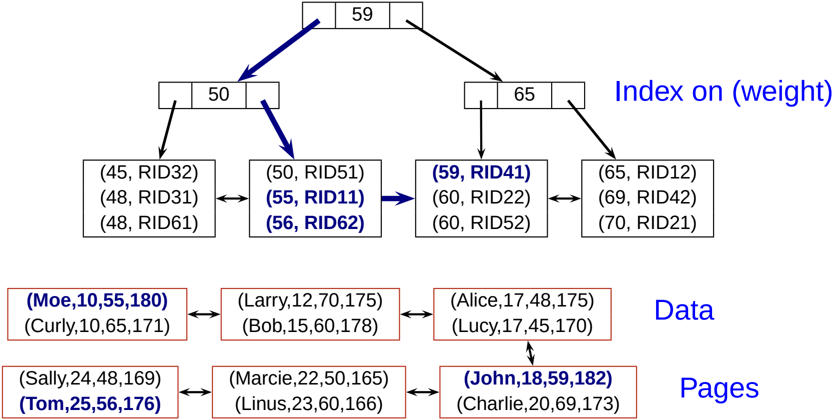


Index on (weight)

Index seek + RID lookups

select name
from Student
where weight **between** 55 **and** 59

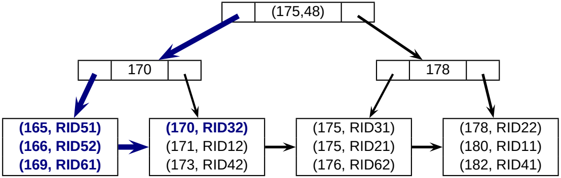
select weight
from Student
where weight **between** 55 **and** 59
and age $\geq$ 20



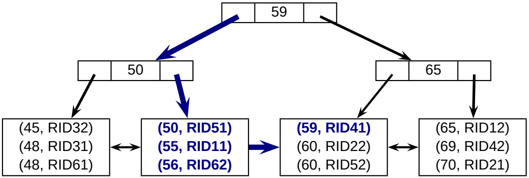
Index intersection

select height, weight **from** Student
where height **between** 164 **and** 170
and weight **between** 50 **and** 59

Index on (height)



Index on (weight)



Sargable predicates

- A selection predicate is a sargable predicate if it is of the form "column comparison-operator value"
  - "sarg" = search argument
- Examples of sargable predicates:
  - year $=$ 2012
  - cost $<$ 2000
- Examples of non-sargable predicates:
  - midterm + final $>$ 50
  - quantity * price = 1000

Selectivity factor

- $sf(p)$ = selectivity factor of a predicate p
  - proportion of records that satisfy p

- **Example**:

```
SELECT   D, E
FROM     R
WHERE    A < 10
AND      B = 20
```

$$sf(A < 10) = \frac{\text{number of records that satisfy } A < 10}{\text{number of records in R}}$$

Covering index

- An index I is a covering index for a query Q if all the columns referenced in Q are part of I
  - Q can be evaluated using I without any RID lookup

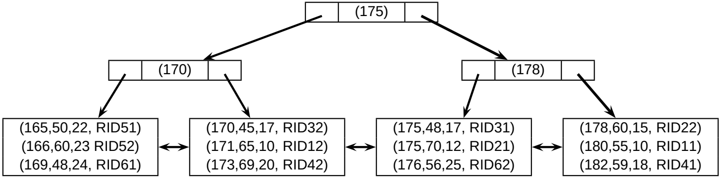
B^+-tree with suffix columns

- Additional columns in leaf node entries
  - I = $(K|S)$
  - K = sequence of key columns
  - S = set of suffix columns (only in leaf nodes)
  - Example: I = $(A, B|C, D)$
- Supported in IBM DB2 & Microsoft SQL Server
- Also known as *included columns*

B^+-tree with suffix columns (cont.)

Example:

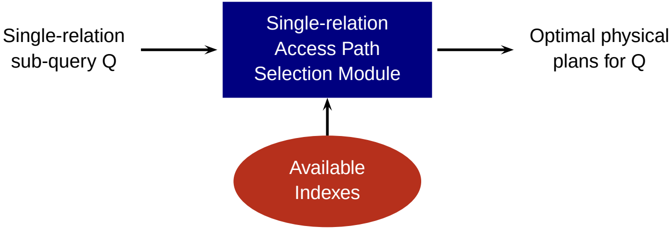
create index IdxHtWtAge **on** Student (height)
include (weight, age)



IdxHtWtAge = $(\textit{height} \mid \textit{weight}, \textit{age})$

Access Path Selection

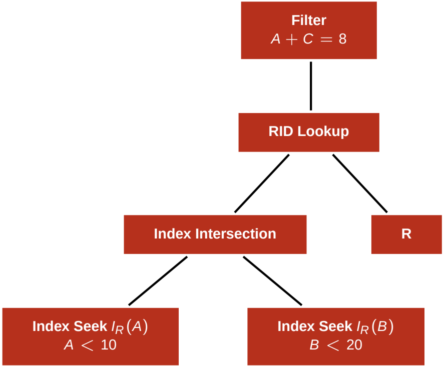
- Find the optimal physical plans for a single relation
- Depends on
  - available indexes
  - selectivity factors of predicates
  - ordering of retrieved tuples



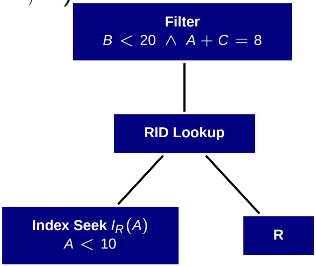
Access Path Selection (cont.)

Query: $\pi_{D,E}(\sigma_{A<10 \wedge B<20 \wedge A+C=8}(R))$

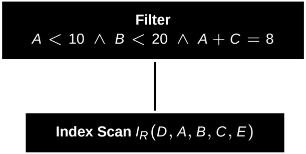
Available indexes: $I_R(A)$, $I_R(B)$, $I_R(D, A, B, C, E)$



Plan 1



Plan 2



Plan 3

Issues in Automated Index Tuning

- Search space of candidate indexes
- Cost model to evaluate index configurations
- Enumeration strategy to search index configurations

Techniques in Automated Index Tuning

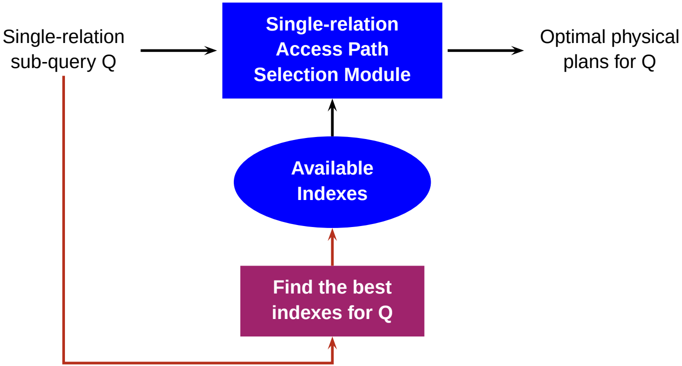
- **What-if API**
  - Hypothetical indexes are not materialized but simulated inside optimizer
  - Add index meta-data & statistics to DB catalogs
- **Dependence on query optimizer**
  - Index tuning is kept “in sync” with optimizer
  - Uses optimizer’s cost model to evaluate candidates

SQL Server's Approach

- Top-down strategy for searching index configurations
- First, obtain an initial optimal index configuration C ($B = \infty$)
- Progressively "relax" C to reduce its size using index transformations
- size(C) = storage requirement for index configuration C
- cost(C) = expected execution cost for workload W given index configuration C

Optimal index configurations

- Instrument the query optimizer with a “tuning mode”



Classification of query attributes

Query attributes can be classified into 5 categories:

1. S_{eq}: attributes in equality sargable predicates
2. S_{range}: attributes in range sargable predicates excluding attributes in S_{eq}
3. N: attributes in non-sargable predicates excluding attributes in $S_{eq} \cup S_{range}$
4. O: attributes in order-by clause
5. A: other referenced attributes excluding attributes in $S_{eq} \cup S_{range} \cup N \cup O$

Example 1

select	D, E
from	R
where	A = 10
and	B $<$ 20
and	$A + C = 8$

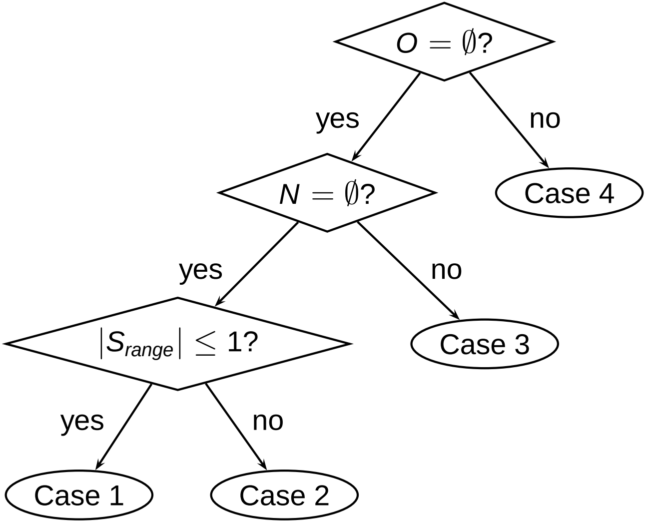
- $S_{eq} = \{A\}$
- $S_{range} = \{B\}$
- $N = \{C\}$
- $O = \emptyset$
- $A = \{D, E\}$

Example 2

select	B, D, E
from	R
where	A = 10
and	B $<$ 20
and	$A + C = 8$
order by	D, B

- $S_{eq} = \{A\}$
- $S_{range} = \{B\}$
- $N = \{C\}$
- $O = \{B, D\}$
- $A = \{E\}$

Classification of optimal index configurations



Case 1: $O = \emptyset$, $N = \emptyset$, $|S_{range}| \leq 1$

- Optimal index = $(K \mid S)$
- $K = K_1, K_2$
  - K_1 = attributes in S_{eq} in any order
  - K_2 = S_{range}
- S = A

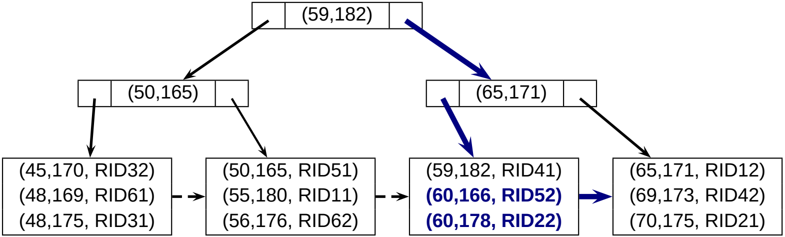
- **Example**:

```
SELECT   D, E
FROM     R
WHERE    A = 10
AND      B < 20
```

Optimal index = $(A, B \mid D, E)$

Why not $K = (K_2, K_1)$?

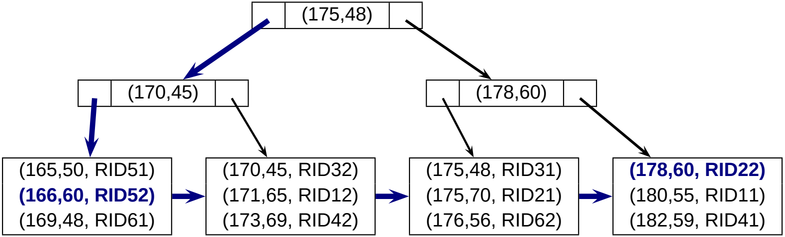
select	name
from	Student
where	height **between** 165 **and** 180
and	weight = 60



Index on (weight,height)

Why not $K = (K_2, K_1)$? (cont.)

select name
from Student
where height **between** 165 **and** 180
and weight = 60



Index on (height,weight)

Case 2: $O = \emptyset$, $N = \emptyset$, $|S_{range}| > 1$

- Optimal index = I or I'
- Index $I = (K|S)$
  - $K = K_1, K_2$
  - K_1 = attributes in S_{eq} in any order
  - K_2 = the attribute in S_{range} with smallest predicate selectivity factor
  - S = $(S_{range} \cup A) - \{K_2\}$
- Index $I' = (K'|S')$
  - $K' = K$
  - L = sequence of attributes in $(S_{range} - K_2)$ ordered in non-descending order of predicates' selectivity factor
  - S' = an optimal prefix of L

Case 2: Example

SELECT N
FROM R
WHERE $W \geq 50$
AND $H \geq 180$
AND $A \geq 40$

Relation R

N	A	W	H
Moe	10	55	180
Curly	10	65	171
Larry	12	70	175
Bob	15	62	178
Alice	17	48	175
Lucy	17	45	170
John	20	59	182
Charlie	20	69	173
Marcie	22	50	180
Linus	23	60	166
Sally	24	48	169
Tom	25	56	176

- Assume $sf(W \geq 50) < sf(H \geq 180) < sf(A \geq 20)$
- Candidate indexes:
  - $I_1 = (W|H, A, N)$
  - $I_2 = (W)$
  - $I_3 = (W|H)$
  - $I_4 = (W|H, A)$

Case 2: Example (cont.)

(45,RID32)
(48,RID61)
(48,RID31)

(50,RID51)
(55,RID11)
(56,RID62)

(59,RID41)
(60,RID52)
(62,RID22)

(65,RID12)
(69,RID42)
(70,RID21)

$I_2 = (W)$

(45,170,RID32)
(48,169,RID61)
(48,175,RID31)

(50,180,RID51)
(55,180,RID11)
(56,176,RID62)

(59,182,RID41)
(60,166,RID52)
(62,178,RID22)

(65,171,RID12)
(69,173,RID42)
(70,175,RID21)

$I_3 = (W|H)$

(45,170,17,RID32)
(48,169,24,RID61)

(48,175,17,RID31)
(50,180,22,RID51)

(55,180,10,RID11)
(56,176,25,RID62)

(59,182,20,RID41)
(60,166,23,RID52)

(62,178,15,RID22)
(65,171,10,RID12)

(69,173,20,RID42)
(70,175,12,RID21)

$I_4 = (W|H,A)$

Case 3: $O = \emptyset, N \neq \emptyset$

Similar to case 2 except the following:

- $S = (S_{range} \cup A \cup N) - \{K_2\}$
- L = sequence of attributes in $(S_{range} \cup N) - \{K_2\}$ in non-descending order of predicates' selectivity factor

Example:

```
SELECT   F
FROM     R
WHERE    A > 20
AND      B ≥ 40
AND      B + C ≤ 100
AND      D * E = 80
```

- Assume $sf(A > 20) < sf(B \geq 40) < sf(B + C \leq 100) < sf(D * E = 80)$
- Candidate indexes:
  - $I_1 = (A|B, C, D, E, F)$
  - $I_2 = (A)$
  - $I_3 = (A|B)$
  - $I_4 = (A|B, C)$
  - $I_5 = (A|B, C, D, E)$

Case 4: $O \neq \emptyset$

- Let I_{opt} be the optimal index obtained under case 3
- Let P be the query plan using I_{opt}
- If the results produced by P are ordered wrt O, then the optimal plan is to evaluate using I_{opt}
- Otherwise, the optimal plan is one of the following:
  1. P followed by sorting on O, or
  2. Evaluate using index $I = (K_1, K_2 | S)$
     - ★ K_1 = attributes in S_{eq} in any order
     - ★ K_2 = attributes in $(O - S_{eq})$
     - ★ S is determined as in case 3

Case 4: Example 1

SELECT	A
FROM	R
WHERE	A $>$ 20
AND	B $\geq$ 40
AND	B + C $\leq$ 100
ORDER BY	A

- Assume $sf(A > 20) < sf(B \geq 40) < sf(B + C \leq 100)$
- Candidate indexes under case 3:
  - $I_1 = (A|B, C)$
  - $I_2 = (A)$
  - $I_3 = (A|B)$
- The optimal index = optimal index under case 3

Case 4: Example 2

SELECT	D, E
FROM	R
WHERE	$A = 20$
AND	$B \geq 40$
AND	$C \leq 100$
ORDER BY	D

- Assume $sf(B \geq 40) < sf(C \leq 100)$
- Candidate indexes under case 3:
  - $I_1 = (A, B | C, D, E)$
  - $I_2 = (A, B)$
  - $I_3 = (A, B | C)$

- Suppose the optimal index under case 3 is I_1
- Optimal index (with sorting): I_1
- Candidate indexes (without sorting):
  - $I_4 = (A, D | B, C, E)$
  - $I_5 = (A, D)$
  - $I_6 = (A, D | B)$
  - $I_7 = (A, D | B, C)$

Handling space constraint

Q1: SELECT A, B, C
FROM R
WHERE $10 < A < 20$

Q2: SELECT A, B, D
FROM R
WHERE $50 < A < 100$

Relaxing index configurations

- An index configuration C can be relaxed if C can be transformed to C' by some index transformation operation such that $size(C') < size(C)$
- Index transformations
  - Merging: merges two indexes into a single index
  - Splitting: splits two indexes into a set of up to 3 indexes
  - Prefixing: reduces the key width of an index
  - Promotion: changes an index to a clustered index
  - Removal: removes an index

Operations on Sequences

- Let S_1 and S_2 be sequences
- $S_1 \cap S_2$ returns the sequence that has elements in the intersection of S_1 and S_2 and in the same order as they appear in S_1
- $S_1 - S_2$ returns the sequence that has elements in the difference of S_1 and S_2 and in the same order as they appear in S_1
- **Example**:
  - $S_1 = [a, b, c, d, e]$
  - $S_2 = [e, f, c, a, g]$
  - $S_1 \cap S_2 = [a, c, e]$
  - $S_2 \cap S_1 = [e, c, a]$
  - $S_1 - S_2 = [b, d]$
  - $S_2 - S_1 = [f, g]$

Merging, $I_1 \oplus I_2$

- Let $I_1 = (K_1 | S_1)$ and $I_2 = (K_2 | S_2)$

$$I_1 \oplus I_2 = \begin{cases} (K_2 | (S_1 \cup S_2) - K_2) & \text{if } K_1 \text{ is a prefix of } K_2 \\ (K_1 | (S_1 \cup K_2 \cup S_2) - K_1) & \text{otherwise} \end{cases}$$

- **Example**:
  - $I_1 = (a, b, c | d, e, f)$
  - $I_2 = (c, d, g | e)$
  - $I_3 = (a | c, e, h)$
  - $I_1 \oplus I_2 = (a, b, c | d, e, f, g)$
  - $I_2 \oplus I_1 = (c, d, g | a, b, e, f)$
  - $I_1 \oplus I_3 = (a, b, c | d, e, f, h)$
  - $I_3 \oplus I_1 = (a, b, c | d, e, f, h)$

Splitting, $I_1 \otimes I_2$

- Let $I_1 = (K_1|S_1)$ and $I_2 = (K_2|S_2)$

$$\begin{aligned}
I_1 \otimes I_2 &= \{I_C|K_1 \cap K_2 \neq \emptyset\} \cup \{I_{R1}|K_1 - K_2 \neq \emptyset\} \cup \{I_{R2}|K_2 - K_1 \neq \emptyset\} \\
I_C &= (K_1 \cap K_2|S_1 \cap S_2) \\
I_{R1} &= (K_1 - K_2|S_1 - S_2) \\
I_{R2} &= (K_2 - K_1|S_2 - S_1)
\end{aligned}$$

- **Example**:
  - $I_1 = (a, b, c|d, e, f)$
  - $I_2 = (c, a|e)$
  - $I_3 = (a, b|d, g)$
  - $I_1 \otimes I_2 = \{I_C, I_{R1}\}$, $I_C = (a, c|e)$, $I_{R1} = (b|d, f)$
  - $I_2 \otimes I_1 = \{I_C, I_{R2}\}$, $I_C = (c, a|e)$, $I_{R2} = (b|d, f)$
  - $I_2 \otimes I_3 = \{I_C, I_{R1}, I_{R2}\}$, $I_C = (a)$, $I_{R1} = (c|e)$, $I_{R2} = (b|d, g)$

Prefixing, $\rho(I, K')$

- Let $I = (K|S)$ and K' be a non-empty prefix of K
- $\rho(I, K') = (K')$
- **Example**:
  - $I_1 = (a, b, c|d, e, f)$
  - $\rho(I_1, ab) = (ab)$
  - $\rho(I_1, a) = (a)$

Promotion, $\tau(I)$

- Let $I = (K \mid S)$ be an index on table R, where R has no clustered index
- $\tau(I)$ changes I to a clustered index on R with search key K

Index Transformation Penalty

- Let C be relaxed to C' using transformation t
- Reduction in space, $\Delta S_t = size(C) - size(C')$
- Maximum increase in cost, $\Delta T_t = cost(C') - cost(C)$
- Transformation penalty, $penalty(t) = \frac{\Delta T_t}{\Delta S_t}$
- Refined penalty:

$$penalty(t) = \frac{\Delta T_t}{\min\{size(C) - B, \Delta S_t\}}$$

Cost Model

- $size(C)$ can be estimated quite easily
- But $cost(C)$ is costly to compute
  - Approximate by computing an upper bound $costBound(C)$

Search strategy

Input: Workload W & space constraint B
Output: A valid index configuration C for W, $size(C) \leq B$

Find optimal configuration C_i for each $q_i \in W$
$C_{best} = \bigcup_{q_i \in W} \{C_i\}$
$S = \{C_{best}\}$; $C_{best} = null$; cost(C_{best}) = ∞;
while (time limit is not exceeded)
    pick $C \in S$ that can be relaxed
    relax C into C' using transformation t that minimizes $penalty(t)$
    $S = S \cup \{C'\}$
    if (size(C') $\leq B$) and (cost(C') $<$ cost(C_{best}))
        $C_{best} = C'$
return C_{best}

How to pick configuration to relax?

- Let C_k be the last relaxed configuration
- Case 1: $size(C_k) > B$
  - Pick C_k
- Case 2: $size(C_k) \leq B$
  - Let $P = \{C_0, C_1, \cdots, C_k\}$ where
    - ★ C_0 is the initial optimal configuration
    - ★ each C_{i-1} is relaxed to C_i using transformation t_{i-1}
  - Let $P' = \{C_i \in P \mid C_i \text{ can be further relaxed}\}$
  - Case 2a: $P' \neq \emptyset$
    - ★ Pick the configuration $C_i \in P'$ where $penalty(t_i) = \max_{C_j \in P'} \{penalty(t_j)\}$
  - Case 2b: $P' = \emptyset$
    - ★ Pick the configuration with the miniumum cost that can be further relaxed

References

Required Readings

- N. Bruno and S. Chaudhuri, Automatic Physical Database Tuning: A Relaxation-based Approach, SIGMOD 2005.

Additional Readings

- Chapters 3, 4, 5, & 6 of Bruno's book
- G. Valentin, M. Zuliani, D.C. Zilio, G.M. Lohman, A. Skelley, DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes, ICDE 2000.