Physical design

Elena Baralis
Politecnico di Torino

Physical design

- Workload characteristics
  - aggregate queries which require accessing a large fraction of each table
  - read-only access
  - periodic data refresh, possibly rebuilding physical access structures (indices, views)
- Physical structures
  - index types different from OLTP
    - bitmap index, join index, bitmapped join index, ...
    - B$^+$-tree index not appropriate for
      - attributes with low cardinality domains
      - queries with low selectivity
  - materialized views
    - query optimizer should be able to exploit them

Physical design

- Optimizer characteristics
  - should consider statistics when defining the access plan (cost based)
  - aggregate navigation
- Physical design procedure
  - selection of physical structures supporting most frequent (or most relevant) queries
  - selection of structures improving performance of more than one query
  - constraints
    - disk space
    - available time window for data update

Physical design

- Tuning
  - a posteriori change of physical access structures
  - workload monitoring tools are needed
  - frequently required for OLAP applications
- Parallelism
  - data fragmentation
  - query parallelization
    - inter-query
    - intra-query
  - join and group by lend themselves well to parallel execution

Bitmap index

- Based on a bit matrix
  - one column for each different value of the indexed attribute domain
  - one row for each tuple (i.e., RID or record identifier)
  - cell (*i*,*j*) is 1 if tuple *i* takes value *j*, 0 otherwise

Example: Index on the Job column in the Employee table
Engineer – Consultant – Manager – Programmer
Assistant – Accountant

RID	Eng.	Cons.	Man.	Prog.	Assis.	Acc.
1	0	0	1	0	0	0
2	0	0	0	1	0	0
3	0	0	0	0	1	0
4	0	0	0	1	0	0
5	0	0	0	0	0	1

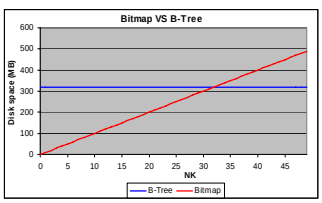
From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Bitmap index

- Suitable for dimensional attributes with small cardinality domain
  - storage requires a limited disk space
  - required disk space increases with growing domain cardinality (NK)

B-tree NR×*Len*(Pointer)
Bitmap NR × NK × 1 bit

Len(*Pointer*) = 4×8 bit



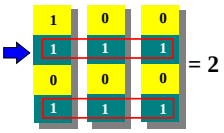
From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Bitmap index

- Efficient for checking boolean expressions of constraints
  - bit to bit and/or on bitmaps

Example: "*How many males in Romagna are insured ?*"

RID	Gender	Insur.	Region
1	M	N	LO
2	M	Y	E/R
3	F	N	LA
4	M	Y	E/R



Example from Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Join index

- Precomputed join between two tables
  - It stores tuples with RIDs corresponding to tuples satisfying join predicates

Sales

Shop_ID	Week_ID	Product_ID	Quantity	Amount
123	13	41	100	10,00
123	24	17	150	15,00
367	36	94	350	35,00
367	27	11	120	12,00

Sales -RID	Shop-RID
1	1
2	1
3	2
4	2

Shop

Shop_ID	Shop	City	States	SalesManager
123	N1	RM	I	R1
367	N3	MI	I	R2

Pairs of RIDs which verify the join condition:
Sales.Shop_ID= Shop.Shop_ID

From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Star join index

- Precomputed join among two or more tables
  - It stores tuples with RIDs corresponding to tuples satisfying join predicates

Weeks

Week_ID	Week	Month
13	Jan1	Jan.
24	Jan2	Jan.

Shops

Shop_ID	Shop	City	State
123	N1	RM	I
367	N3	MI	I

Sales

Shop_ID	Week_ID	Product_ID	Quantity	Amount
123	13	41	100	10,00
123	24	17	150	15,00
367	13	17	350	35,00
367	24	41	120	12,00

Sales RID	Shops RID	Weeks RID	Prod RID
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Products

Prod_ID	Prod.	Type	Category	Supplier
41	P1	A	X	F1
17	P2	A	X	F1

Star join index

- Advantages
  - efficient computation of joins involving first index columns (or all columns)
- Disadvantages
  - useful only for specific join combinations
    - for general usage, it is necessary to store a high number of indices
  - required space may be significant
    - joins always involve the fact table

Bitmapped join index

- Bit matrix which precomputes the join between a dimension and the fact table
  - one column for each dimension RID
  - one row for each fact table RID
  - cell (*i*,*j*) is 1 if fact table tuple *i* joins dimension tuple *j*, 0 otherwise
- May be exploited jointly with traditional bitmap indices to answer complex queries with dimension predicates and multiple joins

RID	1	2	3	…
1	1	0	0	…
2	0	1	0	…
3	0	0	1	…
4	0	1	0	…
5	1	0	0	…
…	…	…	…	…

SHOP dimension RIDs

SALES fact table RIDs

Row 4 in table SALES joins with row 2 in table SHOP

From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Bitmapped join index

By means of bit to bit OR, the RID_i vector of RIDs satisfying all predicates in a dimension table is obtained

Bitmap index on attribute $DT_i.b_i$

RID	Val_1	Val_2	…	Val_k	…	Val_n
1	1	0	…	0	…	0
2	0	0	…	0	…	1
3	0	1	…	0	…	0
4	0	0	…	1	…	0
5	0	0	…	1	…	0
…	…	…	…	…	…	…

RID_i

1

Bitmapped join index $FT.a_i = DT_i.a_i$

RID	1	2	3	4	5	…
1	0	0	0	1	0	…
2	0	0	0	1	0	…
3	0	1	1	0	0	…
4	1	0	0	0	0	…
5	0	0	0	0	1	…
6	0	1	0	0	0	…
…	…	…	…	…	…	…

2

RID 4

1
1
0
0
0
0
…

3 OR bit to bit

RID 5

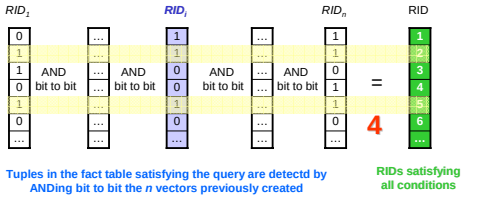
0
0
0
0
1
0
…

=

1
1
0
0
1
0
…

From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Bitmapped join index



Tuples in the fact table satisfying the query are detectd by ANDing bit to bit the *n* vectors previously created

RIDs satisfying all conditions

From Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

Index selection

- Indexing dimensions
  - attributes frequently involved in selection predicates
  - if domain cardinality is high, then B-tree index
  - if domain cardinality is low, then bitmap index
- Indices for join
  - indexing only foreign keys in the fact table is *rarely* appropriate
  - star join index should be used with caution (column order issue)
  - bitmapped join index is suggested (if available)
- Indices for group by
  - use materialized views