Bitmap Indexing and related indexing techniques

Presented by: El Ghailani Maher

Outline

- Introduction
- Why Indexing?
- Factors that determine the convenient Indexing technique
- Criteria to develop a new indexing technique
- Bitmap Indexes
  - Simple Bitmap index
  - Projection Index
  - Bit-Sliced Index
  - Range-Based Indexes
  - Encoded Bitmap Indexes
- Advantages and disadvantages of Bitmap Indexes
- Comparison of the different Indexes techniques
- Conclusion
- References

Introduction

- The growing interest in Data warehousing for decision-makers is becoming more and more crucial to make faster and efficient decisions
- The problem is that most of the queries in a large data warehouse are complex
- Therefore, many indexing techniques are created to speed up access to data within the tables and to answer ad hoc queries in read-mostly environments.

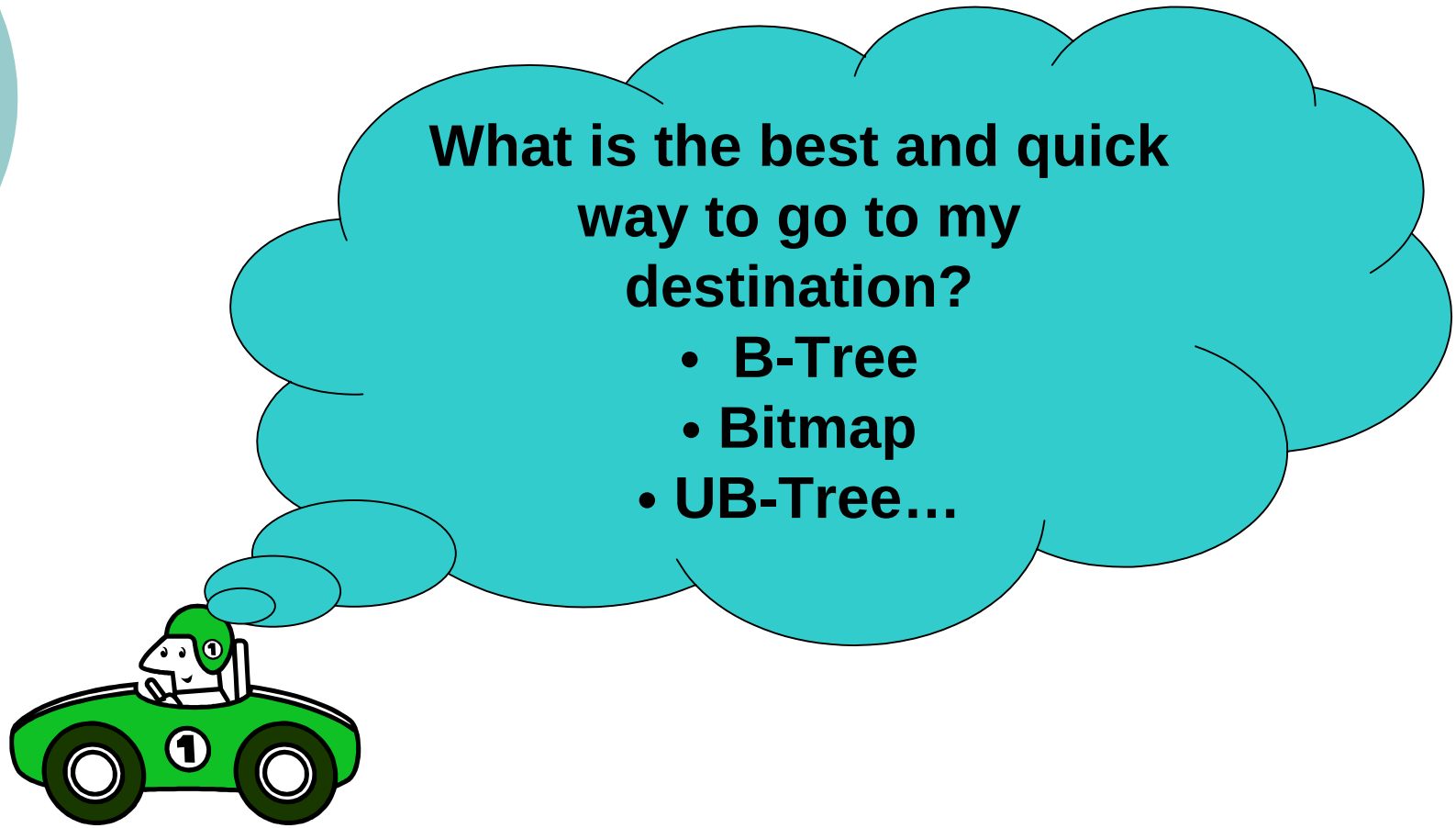
Introduction

- Indexes are database objects associated with database tables and created to speed up access to data within the table.
- They have already existed in the OLTP relational database system but they can not handle large amount of data and complex queries that are common in OLAP systems.

Why Indexing

- Online decision report needs short response.
- Therefore, many indexing techniques have been created to reach this goal in read-only environments.
- the main objective of an indexing technique is to provide the ability to extract data to answer complex and ad hoc queries quickly which is critical for data warehouse applications.

Which Indexing technique should be used in a column?



Factors that determine the convenient Indexing technique:

1. **Cardinality data**
2. **Distribution**
3. **Value Range**

Criteria to develop a new indexing technique:

- The index should be small and utilize space efficiently
- The index should operate with other indexes to fetch the records before accessing raw data.
- The index should support ad hoc and complex queries and speed up join operations
- The index should be easy to build, implement, and maintain

Bitmap Indexes

- Bitmap Indexes were first introduced by O'Neil and implemented in the Model 204 DBMS.
- In data warehouse environments insert, delete operations are not very common therefore, it is better to build an index which optimizes the query performance rather than the dynamic features.

Bitmap Indexes

- In Bitmap indexes complex logical selection operations can be performed very quickly by applying low-cost Boolean operations such as OR, AND, and NOT

- thus, reducing search space before going to the primary source data.

Simple Bitmap Indexes

- The Simple Bitmap Index consists of a collect of bitmap vectors each of which is created to represent each distinct value of the indexed column
- The **ith** bit in a bitmap vector, representing value x, is set to 1 if the **ith** record in the indexed table contains x
- **A Bitmap for a value:** an array of bits where the ith bit is set to 1 if the ith record has the value
- **A Bitmap index:** consists of one bitmap for each value that an attribute can take

Figure 1: Stock Trading Example

Record ID	Ticker Symbol	Trading Volume	Closing Price	Exchange
1	AAPL	4575000	36.625	NASDAQ
2	ABF	64200	24.500	NYSE
3	AET	369000	72.625	NYSE
4	CPQ	8968800	51.375	NYSE
5	DEC	4461100	49.750	NYSE
6	DELL	2714400	89.750	NASDAQ
7	HWP	3009300	90.250	NYSE
8	IBM	7657700	92.500	NYSE
9	IFMX	3493600	33.000	NYSE
10	INTC	17694400	65.500	NASDAQ
11	LGNT	2600	47.250	NASDAQ
12	MSFT	18288600	91.125	NASDAQ

Stock Trading Example

- stocks are traded at two different stock exchanges at NASDAQ and at NYSE
- we see that our stock example comprises 12 different stocks which are uniquely identified by their record ID given in the first column.

Stock Trading Example

- We can represent Stocks and their corresponding trading places by the following simple bitmap:

Example:

- NASDAQ: (1 0 0 0 0 1 0 0 0 1 1 1)
- NYSE: (0 1 1 1 1 0 1 1 1 0 0 0)
- we have a straightforward way of describing the stock exchange by means of bitmaps.

Stock Trading Example

- **Question ?** How do we retrieve data from such a bitmap index?
- If we make a simple modification of our example and suppose that some stocks are traded at both stock exchanges
- NASDAQ: (1 0 1 1 0 1 0 0 0 1 1 1)
- NYSE: (0 1 1 1 1 0 1 1 1 0 1 0)
- We notice that the **3rd**, **4th** and **11th** bit in our example are traded at both stock exchange.

Stock Trading Example

- We simply **AND** both bitmaps together so that to retrieve this information from our database.

- NASDAQ: (1 0 1 1 0 1 0 0 0 1 1 1)
- NYSE: (0 1 1 1 1 0 1 1 1 0 1 0) **AND**
- (0 0 **1** **1** 0 0 0 0 0 0 **1** 0)

- Given that the **3rd**, the **4th** and the **11th** bits of the resulting bitmap are set to 1, we can know that these stocks are traded at both stock exchanges.

Projection Index

- A Projection Index on an indexed column A in a table T stores all values of A in the same order as they appear in T.
- it is simply a sequence of column values from any table where the ordinal row number of table gives the order of the bitmap index.

Figure 2.1 : Projection Index Example

Col1	Col2	Col3	Col4
	v1		
	v2		
	.		
	.		
	.		
	v_k		

Col2
v1
v2
.
.
.
v_k

An other Example

PRODUCT TABLE

Product ID	Weight	Size	Package_Type
P10	10	10	A
P11	50	10	B
P12	50	10	A
P13	50	10	C
P14	30	10	A
P15	50	10	B
P16	50	10	D
P17	5	10	H
P18	50	10	I
P19	50	10	E
P21	40	10	I
P22	50	10	F
P23	50	10	J
P24	50	10	G
P25	10	10	F
P26	50	10	F
P27	50	10	J
P28	20	10	H
P29	50	10	G
P30	53	10	D

CUSTOMER TABLE

Customer_ID	Gender	City	State
C101	F	Norman	OK
C102	F	Norman	OK
C103	M	OKC	OK
C104	M	Norman	OK
C105	F	Roncake	VA
C106	F	OKC	OK
C107	M	Norman	OK
C108	F	Dallas	TX
C109	M	Norman	OK
C110	F	Moore	OK

SALE TABLE

Product_ID	Customer_ID	Total_Sale
P10	C105	100
P11	C102	100
P15	C105	500
P10	C107	10
P10	C106	100
P10	C101	900
P11	C105	100
P10	C109	20
P11	C109	100
P10	C102	400
P13	C105	100

Figure 2.2: An example of the PRODUCT, CUSTOMER and SALE table.

Projection Index Example

Package_Type
A
B
L
C
:
G
D

(a) Projection

B_A	B_B	B_C	B_D	B_E	B_F	B_G	B_H	B_I	B_J	B_K	B_L
1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0
:	:	:	:	:	:	:	:	:	:	:	:
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0

(b) Pure Bitmap Index

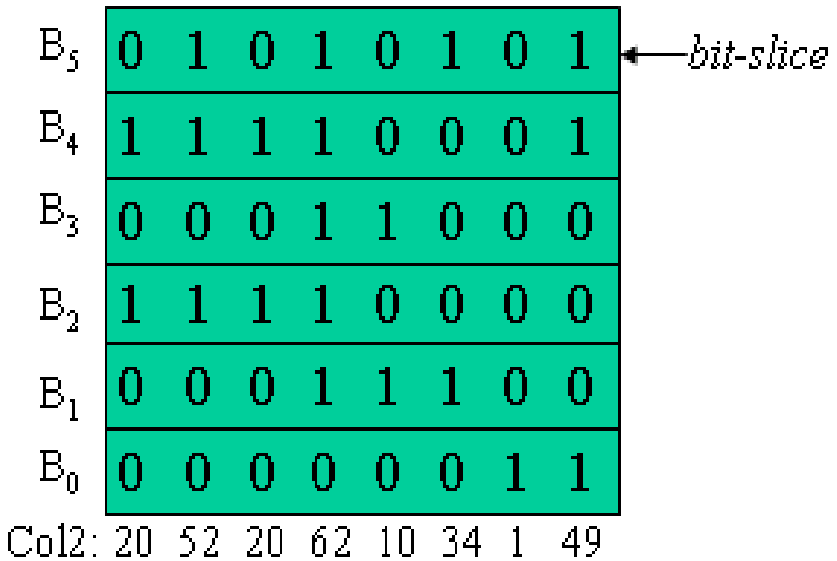
Figure 3: An example of the Projection Index and Pure Bitmap Index on the package_type column of PRODUCT table.

Projection Index

- having the Projection Index on these columns reduces extremely the cost of querying
- because a single I/O operation may bring more values into memory.

Bit-Sliced Index

- Bit-Sliced Index is a set of bitmap slices which are orthogonal to the data held in a projection index.



Bit-Sliced Index

- A bit-sliced index based on converting integer values to binary values in order to perform fast logical operations on them since that hardware support directly.

- We should choose an optimal number of bits per bit-vector in order to represent the whole attribute domain and to occupy minimum space.

Range-Based Indexes

- The space complexity of the Simple Bitmap index is low for low cardinality attributes but large for high cardinality attributes.
- Range-Based Index is a simple modification of the bitmap index that handles to some extent this clear weakness
- The variation is that the bitmap vector is used to represent a range rather than a distinct attribute value as we saw it in our previous example for the attribute Exchange.
-

Range-Based Indexes

- The most important idea of Range-Based Indexes is to reduce storage overhead
- by partitioning, That is, attribute values are split into smaller number of ranges and represented by bitmap vectors.
- Indeed, a bit is set to 1 if a record falls into specified range; otherwise this bit is set to 0.

Range-Based Indexes Example

- We suppose that a maximum trading volume per day is 20.000.000 shares.
- Then we divide the attribute Trading Volume into two equal ranges:

[10.000.000, 20.000.000]: (0 0 0 0 0 0 0 0 0 **1** 0 **1**)
[0, 10.000.000): (1 1 1 1 1 1 1 1 1 0 1 0)

- For example, the 10th and 12th stock are traded in a volume greater than 10.000.000 stocks per day

Range-Based Indexes Example

- The great advantage of a Range-Based index over the Simple Bitmap index is that only a lower number of bitmap vectors need to be stored.

- Nevertheless, the resulting query process might be longer.

But how are data retrieved?

- We suppose that we are interested in all stocks at NYSE that have a trading volume of more than 4 millions shares.
- Therefore, the two bitmap vectors for the attribute Exchange and the range [0, 10.000.000) are ANDed together:

- [0, 10.000.000): (1 1 1 1 1 1 1 1 1 0 1 0)
- NYSE: (0 1 1 1 1 0 1 1 1 0 0 0) **AND**
- Candidates (0 **1 1 1 1** 0 **1 1 1** 0 0 0)

But how are data retrieved?

- There are 7 candidates which are represented by the 1-bit, but we still need to check the value either larger than 4 millions or not.
- Range-Based index needs two search steps instead of only one which is true for Simple Bitmap index.
- But, one of the great difficulties with this index is to find an optimal partitioning of the range in order to lower the processing time in step 2.

Encoded Bitmap Indexes

- The weaknesses of SBI for high cardinality attributes lead to the suggestion of encoded bitmap indexing which provides the advantage of a drastic reduction in space requirements
- The main idea of EBI is to encode the attribute domain.

Encoded Bitmap Indexes Example

We will see the following **example**:

- We assume that we have a fact table SALES with N tuples and a dimension table PRODUCT with 12.000 different products.
- If we build a simple bitmap index on PRODUCT, It will require 12.000 bitmap vectors of N bits in length.
- However, if we use encoded bitmap indexing we only need ceil($\log_2$ 12.000)=14 bitmap vectors **plus** a **mapping table** which is a very significant reduction of the space complexity.

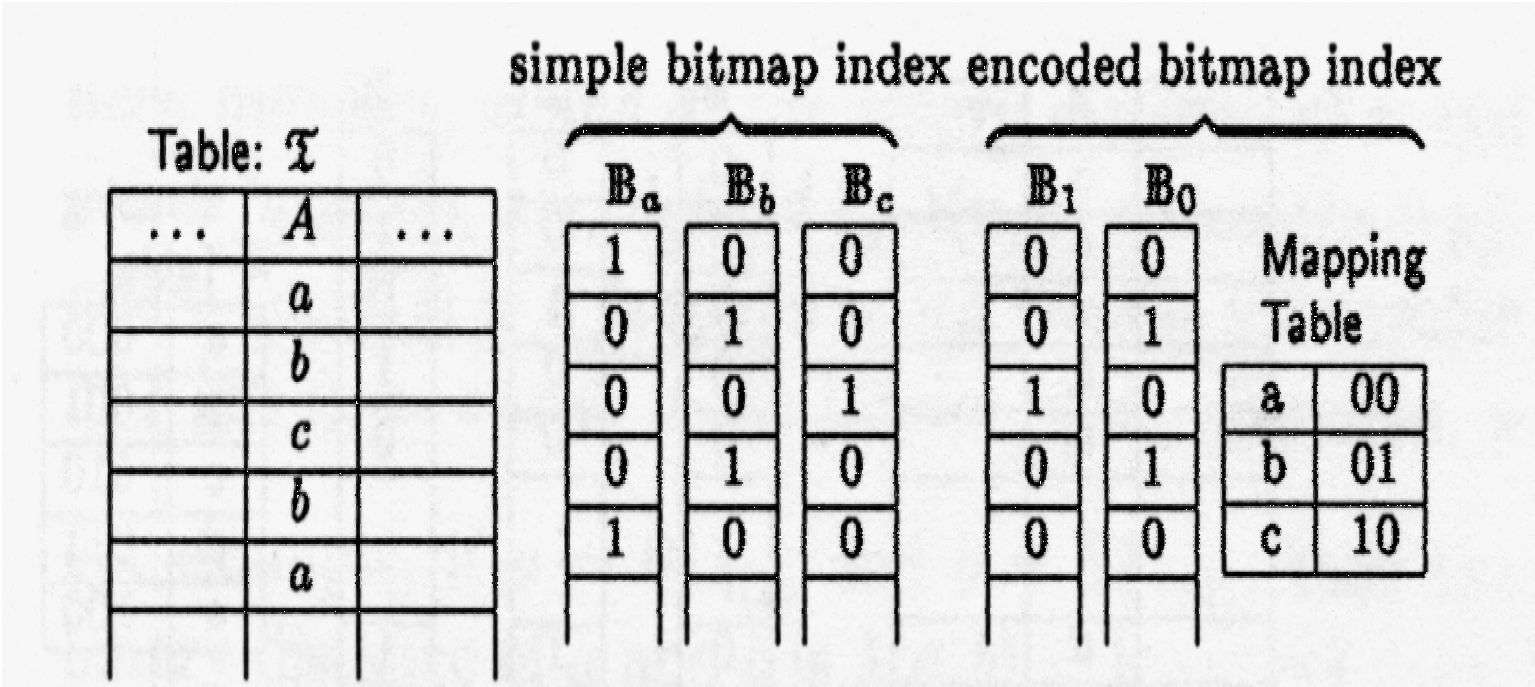
Encoded Bitmap Indexes Example

- In this new example we will show how **Huffman encoding** used for reducing the space complexity of bitmap indexes:
- We assume that our attribute domain is given by the table T is {a,b,c}.
- The encoding schema of EBI is stored in a separate table called mapping table and simply encodes the values from a SBI by means of Huffman encoding
- **therefore** reduces the number of bitmaps vectors. In particular, we use only ceil($\log^2 3$)=2 Encoded Bitmap vectors instead of 3 simple bitmap vectors.

Encoded Bitmap Indexes Example

- This means that 2 bits are used to encode the domain {a,b,c}.
- For example, the attribute value of a is represented by the bit string 100 in the table of the SBI but in the table of EBI the attribute value a is encoded as 00.

Figure3: Huffman encoded bitmap index



Advantages and disadvantages of Simple Bitmap Indexes

- **Advantages:**
  - One of the main advantages of bitmap indexes is that logical operations are very well supported by hardware and, thus, the operations are executed quite fast.
  - In addition, both the cost for constructing bitmap indexes and the processing costs are very low.

- **Disadvantages:**
  - For high cardinality attributes the space complexity becomes so large that this technique might not be very space efficient.

Comparison of the different Indexes techniques

Indexing Techniques	Characteristics	Advantages	Disadvantages	Implementing Commercial Systems
B-Tree Index	Two representations (row id and bitmap) are implemented at the leaves of the index depending on the cardinality of the data.	• It speeds up known queries. • It is well suited for high cardinality. • The space requirement is independent of the cardinality of the indexed column. • It is relatively inexpensive when we update the indexed column since individual rows are locked.	• It performs inefficiently with low cardinality data • It does not support ad hoc queries. More I/O operations are needed for a wide range of queries. • The indexes can not be combined before fetching the data.	• Most of commercia products (Oracle, Informix, Red Brick

Comparison of the different Indexes techniques

Pure Bitmap Index	An array of bits is utilized to represent each unique column value of each row in a table, setting the bits corresponding to the row either *ON*(valued 1) or *OFF*(valued 0). The equality encoding scheme is used.	• It is well suited for low-cardinality columns. • It utilizes bitwise operations. • The indexes can be combined before fetching raw data. • It uses low space • It works well with parallel machine. • It is easy to build. • It performs efficiently with columns involving scalar functions (e.g., COUNT). • It is easy to add new indexed value. • It is suitable for OLAP.	• It performs inefficiently with high cardinality data. • It is very expensive when we update index column. The whole bitmap segment of the updated row is locked so the other row can not be updated until the lock is released. • It does not handle spare data well.	• Oracle • Informix • Sybase • Informix • Red Brick • DB2

Comparison of the different Indexes techniques

Encoded Bitmap Index	The index is the binary Bit-Sliced Index built on the attribute domain	• It uses space efficiently. • It performs efficiently with wide range query.	• It performs inefficiently with equality queries. • It is very difficult to find a good encoding scheme. • It is rebuilt every time when a new indexed value for which we run out of bit to represent is added.	• DB2

Comparison of the different Indexes techniques

Projection Index	The index is built by storing actual values of column(s) of indexed table.	• It speeds up the performance when a few columns in the table are retrieved.	• It can be used only to retrieve raw data (i.e., column list in selection).	• Sybase

Conclusion

- There is no basic index that is best suited for all applications. Each application has its own specificities.
- The Bitmap indexing works well in low cardinality but not for high cardinalities.
- Compressed Bitmap is a promising technique to overcome this problem.
- we need some other efficient encoding techniques to lower the number of logical operations.

References

- **[1]** Mag. Kurt Stockinger. Optimization of DB-Access. Literaturseminar SS 1999
- **[2]** Ming-Chuan Wu, Alejandro P. Buchmann. Encoded Bitmap Indexing for data warehouses. DVS1, Computer Science Department, Technische Universitat Darmstadt, Germany.
- **[3]** Sirirut Vanichayobon Le Gruenwald. Indexing techniques for Data Warehouses' queries. The University of Oklahoma.
- **[4]** Chee-Yong Chan and Yannis E. loannidis. Bitmap Index Design and Evaluation. University of Wisconsin-Madison
- **[5]** Sihem Amer-Yahia and Theodore Johnson. Optimizing queries on compressed Bitmaps. AT&T Labs-Research
- **[6]** Marcus Jurgens, Hans-J Lenz. Tree Based Indexes vs Bitmap Indexes: A performance Study. Institute of Statistics and Econometrics.