|  |  |
| --- | --- |
| Faculty of Computer & Information Sciences  Ain Shams University  Subject: DBA 271 File Organization  Year: (2nd year)undergraduate  Academic year: 2nd term 2019-2020 |  |

**Research Topic (6)**

**Title: Database Indexes**

**Introduction** (Ref. 1)

**INDEXING** is a data structure technique that permits you to rapidly retrieve records from a database file and optimize the performance of the database by limiting the number of disk accesses required when a query is processed. An Index is a little table having only two columns. The first column includes a copy of the primary or candidate key of a table. The second column of the table contains a set of pointers for holding the location of the disk block where that particular key value stored.

**1. Different Types of Database Indexes:**

* Primary Index
  + Dense Index
  + Sparse Index
* Secondary Index
* Clustering Index
* Other types

**2. Comparison Between Index Types:**

* **Primary Index:**

Primary Index is defined on an ordered and organized data file that contains the key fields of the table and a pointer to the non-key fields of the table. The primary index is created automatically when the table is created in the database and also considered as the key of the relation.

The primary indexing is divided into another two types:

* + **Dense Index:**

In dense index, there is an index record for each search key value in the database. This helps you to search quicker but requires more space to store index records. Index records contain search key value and a pointer to the actual record on the disk.

* + **Sparse Index:**

In sparse index, index records are not made for each search key. An index record here contains a search key and a real pointer to the data on the disk. To look through a record. Sparse index helps you to resolve the issues of dense indexing, we initially continue by index record and reach at the actual location of the data. It needs less space. If the data we are searching for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

* **Secondary Index:**

Secondary Index is generated by a field which is a candidate key and has a unique value in each record, or a non-key with duplicate values.

It’s also known as a non-clustering index.

* + **Non-clustered:**

A non clustered index is made using clustered index. Each record line in the non clustered index has non clustered key value and a row locator. It’s known that all locator positions to the data row in the clustered index that has a key value.

* + **Filtered:**

A non clustered index. Totally improved and optimized for query data from a well characterized subset of data. A filter is used to predicate a portion of rows in the table to be indexed.

* **Clustering Index:**

The importance of clustered index is to sort and store the rows data of a table based on the order of clustered index key. Clustered index key used to be implemented in B-tree index structure.

* **Other types:**
  + **Unique:** Unique index guarantees the availability and accessibility of only non-duplicate values and therefore, every row is unique.
  + **Full-text:** Its support is efficient in searching and looking through words in string data. This sort of indexes is utilized in certain database managers.
  + **Spatial:** It encourages the ability for performing operations in efficient way on spatial objects. To do this, the column ought to be of geometry type.

**3. Clustered vs. Non-Clustered Indexes:**

|  |  |
| --- | --- |
| **Clustered Index** | **Non-Clustered Index** |
| * Can be used only one per table | * Can be used many times per table |
| * Usually made on the primary key | * Usually made on any key |
| * Faster to read than non clustered as data is physically stored in index order | * Faster to insert and update operations than a clustered index |
| * Do not need extra space to store logical structure | * Use extra space to store logical structure |

The two kinds of the index will improve performance and execution when select data with fields that utilize the index but will hinder update and insert operations.

As a result of the slower insert and update clustered indexes ought to be set on a field that is ordinarily incremental i.e. Id or Timestamp

Clustered indexes physically order the data on the disk. This means no extra data is needed for the index, but there can be only one clustered index (obviously). Accessing data using a clustered index is fastest.

All other indexes must be non-clustered. A non-clustered index has a duplicate of the data from the indexed columns kept ordered together with pointers to the actual data rows (pointers to the clustered index if there is one). This means that accessing data through a non-clustered index has to go through an extra layer of indirection. However if you select only the data that's available in the indexed columns you can get the data back directly from the duplicated index data (that's why it's a good idea to SELECT only the columns that you need and not use \*)

**4. Description of the Implementation of my code:**

I’ll discuss the steps that I did to reach the final output:

1. I made a XML file called “people”, put all the data in it. The data are (ID, Name, City, Country) and wrote all the code that used to make the file and its data in the main function of the program. I wrote 40 records and assumed that city is unique for each record.
2. I sorted the data that in the XML file according to the “ID” in ascending order.
3. I divided the sorted data into 4 clustered (XML files), each file contains 10 records, and the files are “people1”, “people2”, “people3”, “people4”.
4. I made a XML file called “Entry\_Indices\_Table” that carries out the entry indices for the fourth files.

**5. Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | No of records in data | No of comparisons made to search for a record | |
| Search using ID | Search using City |
| 1 | 8 | ID:13  Num. of Iterations: **5** | City: Beni Ebeid  Num. of Iterations: **13** |
| 2 | 10 | ID: 15  Num. of Iterations: **7** | City: Cairo  Num. of Iterations: **15** |
| 3 | 39 | ID: 40  Num. of Iterations: **13** | City: 10th of Ramadan  Num. of Iterations: **40** |
| 4 | 15 | ID: 12  Num. of Iterations: **4** | City: Beni Mazar  Num. of Iterations: **12** |
| 5 | 26 | ID: 26  Num. of Iterations: **9** | City: El Husseiniya  Num. of Iterations: **26** |

1. **Conclusion**

**Search using id is faster than Searching using the name of the city.**

**References:**

**[1] Geeks for Geeks** (Website): [Link](https://www.geeksforgeeks.org/indexing-in-databases-set-1/#:~:text=Indexing%20is%20a%20way%20to,using%20a%20few%20database%20columns.)

[2]……………………………………………………………………………

[3]……………………………………………………………………………

[4]……………………………………………………………………………

[5]……………………………………………………………………………

[at least 5 references]

With My Best Regards,   
your signature