**Information Technology Institution**

**Intensive Training Program**

**Data Analysis Track**

**Index Types In DWH**

Submitted By:

**Salma Samir**

**Mohammed Adel Eltohamy Alsheikh**

**abdelrahman aly youssef**

**Kareem Fouad Hamouda**

# Indexing Techniques in Data Warehouses

In data warehouse systems, there are many indexing techniques. Each existing indexing technique is suitable for a particular situation. In this report we describe several indexing techniques being studied/used in both academic research and industrial applications.

## The B-Tree Index

The B-Tree Index is the default index for most relational database systems. The topmost level of the index is called the root. The lowest level is called the leaf node. All other levels in between are called branches. Both the root and branch contain entries that point to the next level in the index. Leaf nodes consisting of the index key and pointers pointing to the physical location in which the corresponding records are stored.

A screenshot of a computer

Description automatically generated with medium confidence

Figure PRODUCT, CUSTOMER and SALE table.

A screenshot of a computer

Description automatically generated with medium confidence

Figure B-Tree Index on the package\_type column of the PRODUCT table.

## 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**. Each row of the Projection Index stores one value of **A**. The row order of value ***x*** in the index is the same as the row order of value ***x*** in **T**. Normally, the queries against a data warehouse retrieve only a few of the table’s columns, so having the Projection Index on these columns reduces tremendously the cost of querying because a single **I/O** **operation** may bring more values into memory. Sybase builds a Projection Index under the name of **Fast Projection Index** on every column of a table.

## Bitmap Index

The bitmap representation is an alternate method of the row ids representation. It is simple to represent and uses less space- and CPU-efficient than row ids when the number of distinct values of the indexed column is low. The indexes improve complex query performance by applying low-cost Boolean operations such as OR, AND, and NOT in the selection predicate on multiple indexes at one time to reduce search space before going to the primary source data. Many variations of the Bitmap Index such as (Pure Bitmap Index, Encoded Bitmap) have been introduced, aiming to reduce space requirement as well as improve query performance.

## Join Index

Graphical user interface, text, application

Description automatically generatedA Join Index is built by translating restrictions on the column value of a dimension table (ex: gender column) to restrictions on a large fact table. The index is implemented using one of the two representations: row id or bitmap, depending on the cardinality of the indexed column. A bitmap representation, which is called Bitmap Join Index, is used with the low cardinality data while a row id representation is used with a high cardinality. In DW, there are many join operations involved; so building Join Indexes on the joining columns improves query-processing time. For example in Figure.1, Bitmap Join Indexes on the **gender column** in the **SALE** table can be built by using the gender column in the **CUSTOMER** table and the foreign key customer id in the SALES table. Note that the Sales table does not contain the gender column. The Bitmap Join Index for gender equal to male is created by setting a bit corresponding to a row for customer\_id whose gender is ‘M’ to 1 in the Sales Table.