**INDEXING :**

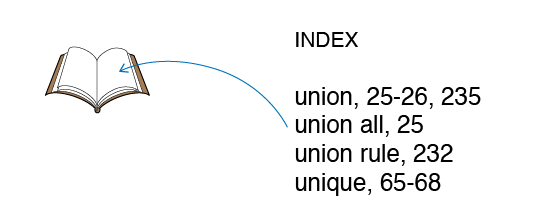
A database allows us to search values from tables based on any attribute present in the table. Say, we want to fetch the names of all employees who belong to the ETA department. How would the database go about providing the result?

| **ID** | **ENAME** | **DOJ** | **SALARY** | **DEPT** |
| --- | --- | --- | --- | --- |
| 1 | James Potter | 01-JUN-14 | 75000 | ICP |
| 2 | Ethan McCarty | 01-FEB-14 | 90000 | ETA |
| 3 | Emily Rayner | 01-JAN-14 | 25000 | ETA |
| 4 | Jack Abraham | 01-JUL-14 | 30000 | ETA |
| 5 | Ayaz Mohammad | 01-APR-14 | 40000 | ICP |

The database starts looking at each row of the table, checks for a match and includes the matched row into the result. Such an operation is called a **Table Scan** and would take a very long time especially if the table is very large.

**Indexing Concept :**

We have all used index in a book. Let us say we want to read about the UNION clause in a database book. We do not start reading the book from the beginning till we find the UNION topic. Instead, we first locate the topic in the index at the back of the book. The index is already sorted alphabetically on topics, so finding the entry in the index does not take much time. The index points us to the exact location (in this example page number 25) of the topic in the book. So we are able to find the topic with ease.



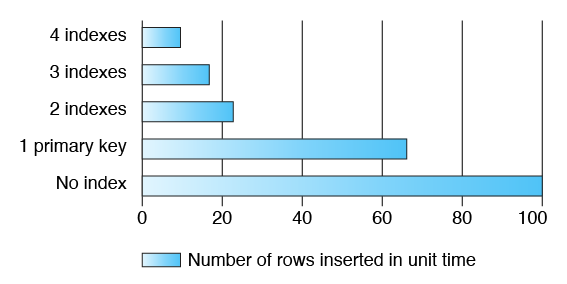
Index in databases function very much like an index in a book. Index is a database object that allows us to search data in tables quickly and efficiently.

**Disadvantages :**

While index increase the performance of SELECT statement they do come with a cost.

1. Index requires additional amount of space so the database size becomes bigger. Bigger size means more time for data backup and recovery.
2. Database Servers continually update indexes automatically with each UPDATE, DELETE and INSERT operations. These operations become slower as a result.

**Degradation of Insert Performance with addition of indexes**

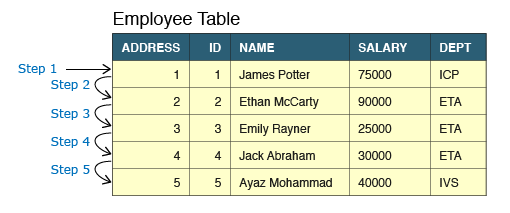


**Table Scan in Indexing :**

Data is retrieved from tables using **Table Scan**, **Index Unique Scan** or **Index Range Scan**. While the first process does not use an index at all, the other two leverage index created in database. Index is used only if sort order of data in the index can be leveraged for faster search.

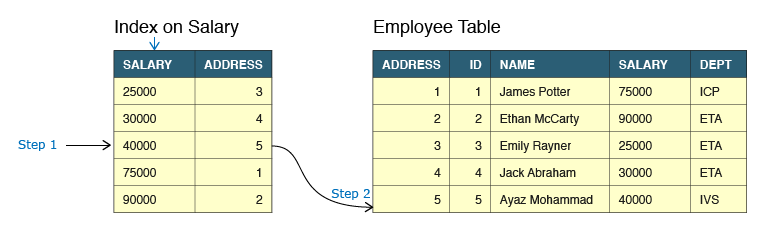
**Table Scan** reads all rows from a table and filters out those that do not meet the where clause predicates. A full table scan is selected when:

* A large portion of the rows in the table must be accessed
* Index does not exist on the column being searched
* Index exist but cannot be used



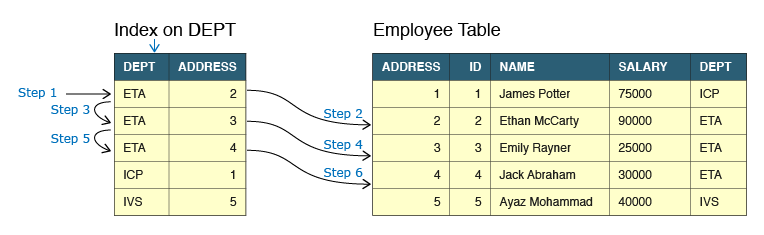
**Index Unique Scan :**

**Index Unique Scan** is used when there is an equality predicate on a unique index or an index created as a result of a primary key constraint. It always retrieves one row from the database.

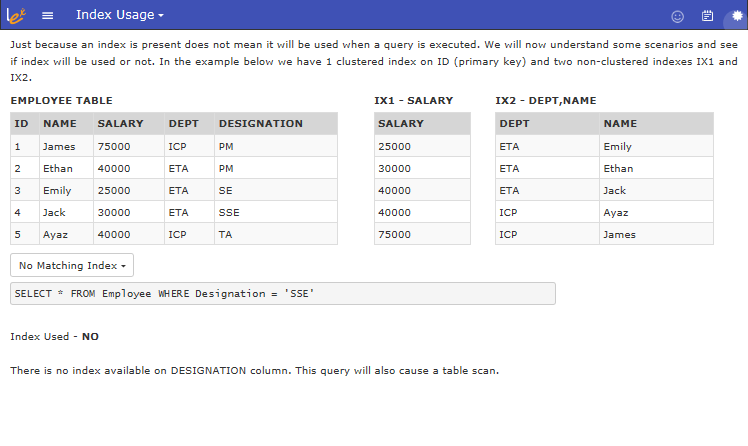


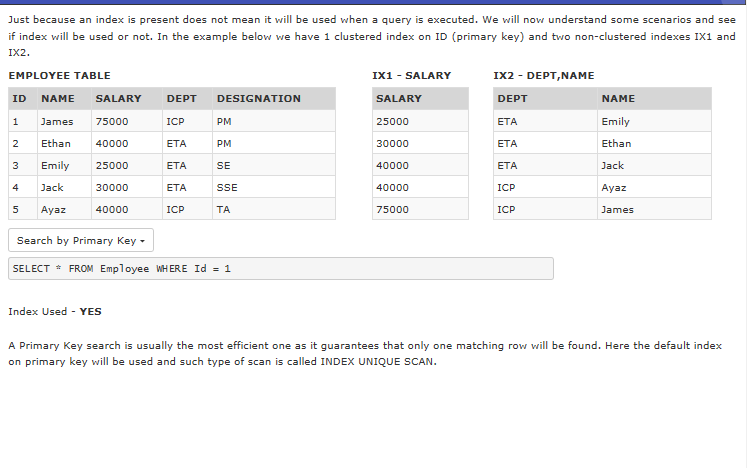
**Index Range Scan :**

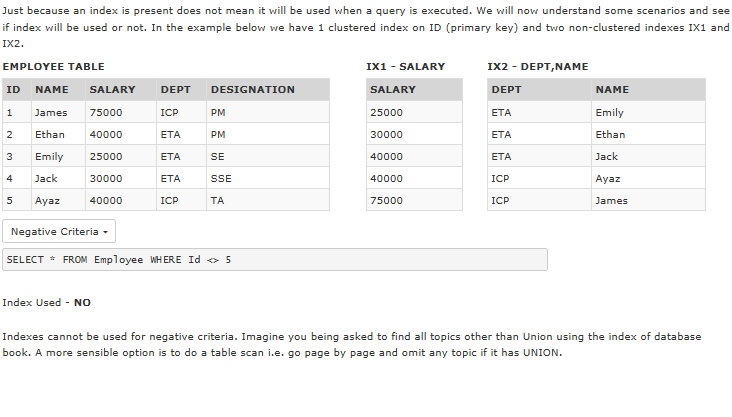
**Index Range Scan** is used when SQL statement has an equality or range predicate (>, <, between) on a non-unique index key, or a range predicate on a unique index key. data is returned in the ascending order of index columns.

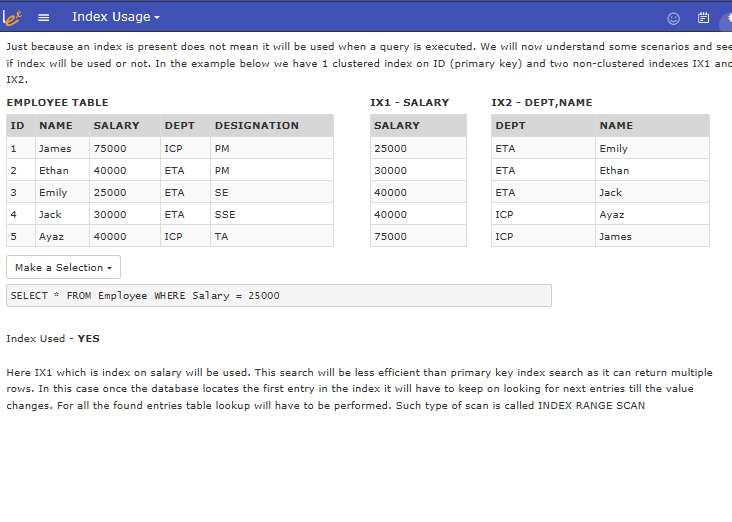


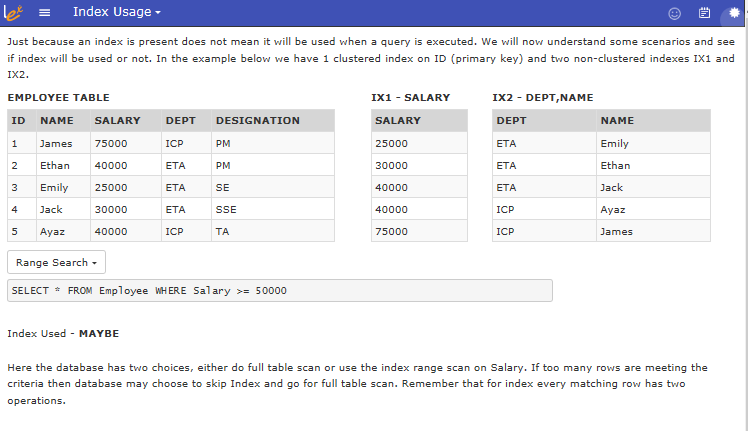
**Index Usages :**

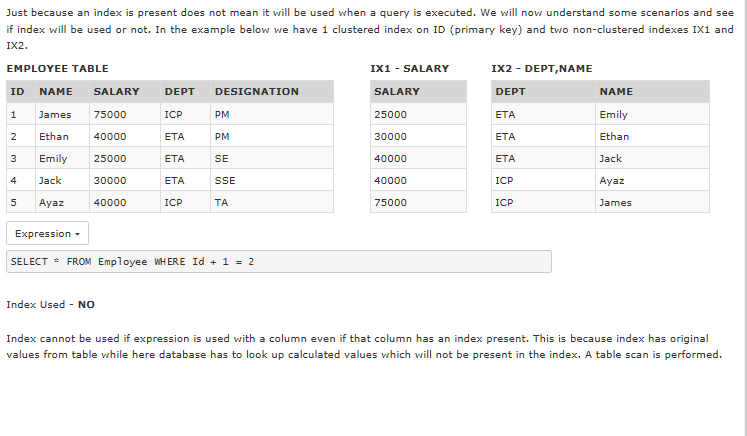


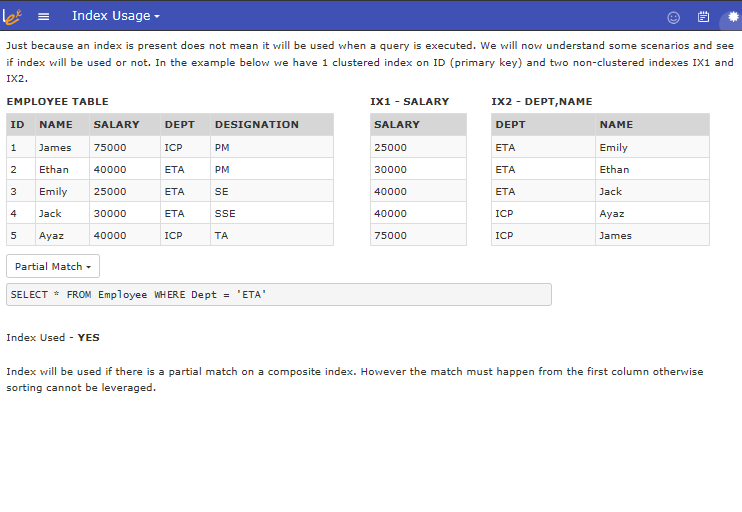












**Index Summary :**

Let us summarize what we have learnt about indexes in database systems.

* Indexes greatly improve performance of SELECT statements.
* Indexes slow down performance of Insert, Update and Delete statements.
* Indexes are generally created on columns that are frequently used in WHERE clause.
* Indexes are also used to enforce integrity constraints like Primary Key and Unique constraint.
* Index consumes additional space and as a result slow down database backup and recovery processes.
* As a rule of thumb index can be used only if
  + Index sort order can be leveraged i.e. data matching happens from first column.
  + there is significant filtering of data achieved
* Indexes should be created carefully targeting specific statements. Creating indexes indiscriminately hoping to improve performance will not work in most cases.
* Index can be located by querying the data dictionary view USER\_IND\_COLUMNS.
* SELECT INDEX\_NAME, COLUMN\_NAME FROM USER\_IND\_COLUMNS WHERE TABLE\_NAME = 'EMPLOYEE';