Index

An index can be compared to a book index enabling rapid search capabilities.

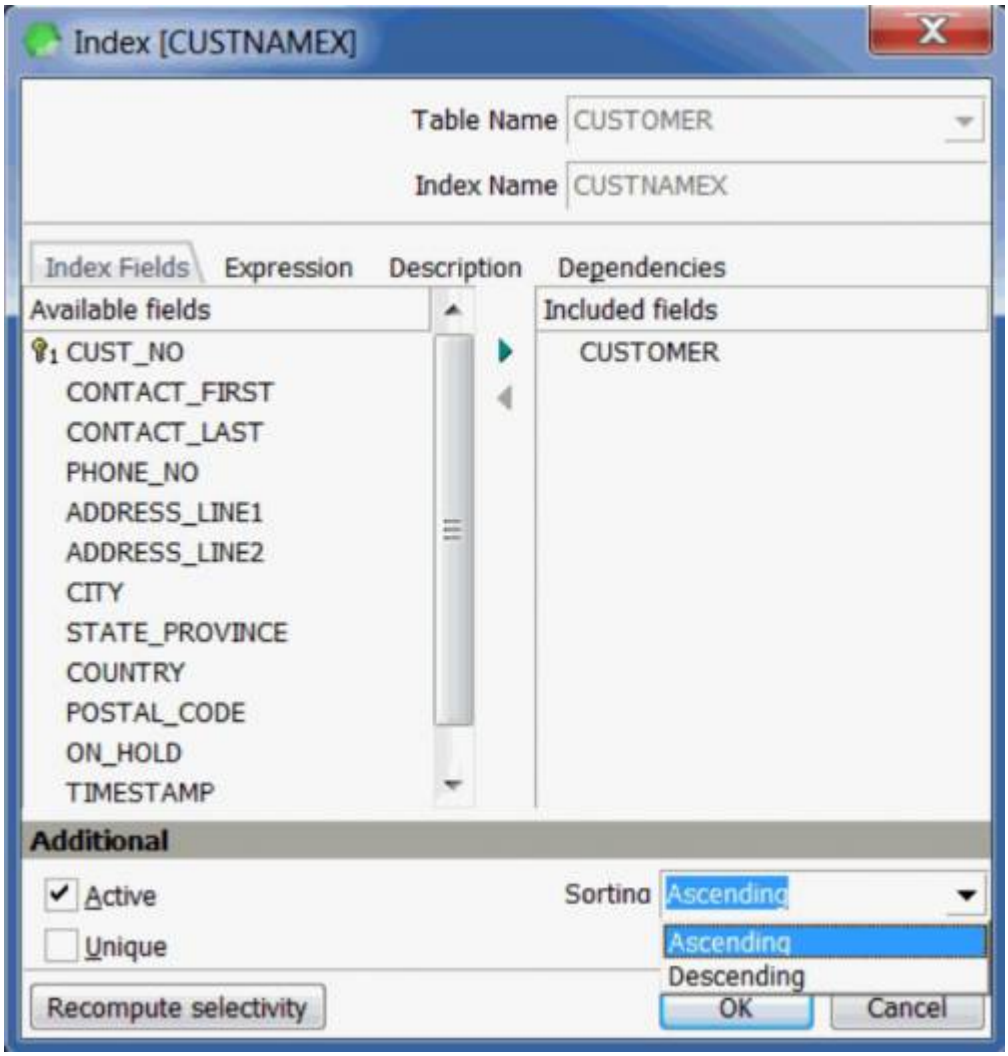
Indices are a sorted list of pointers into tables, to speed data access. They can be best described as an alphabetical directory with internal pointers, where what can be found. If the indexed field is unique there is only one pointer.

An index can be ascending or descending, and can also be defined as unique if wished.

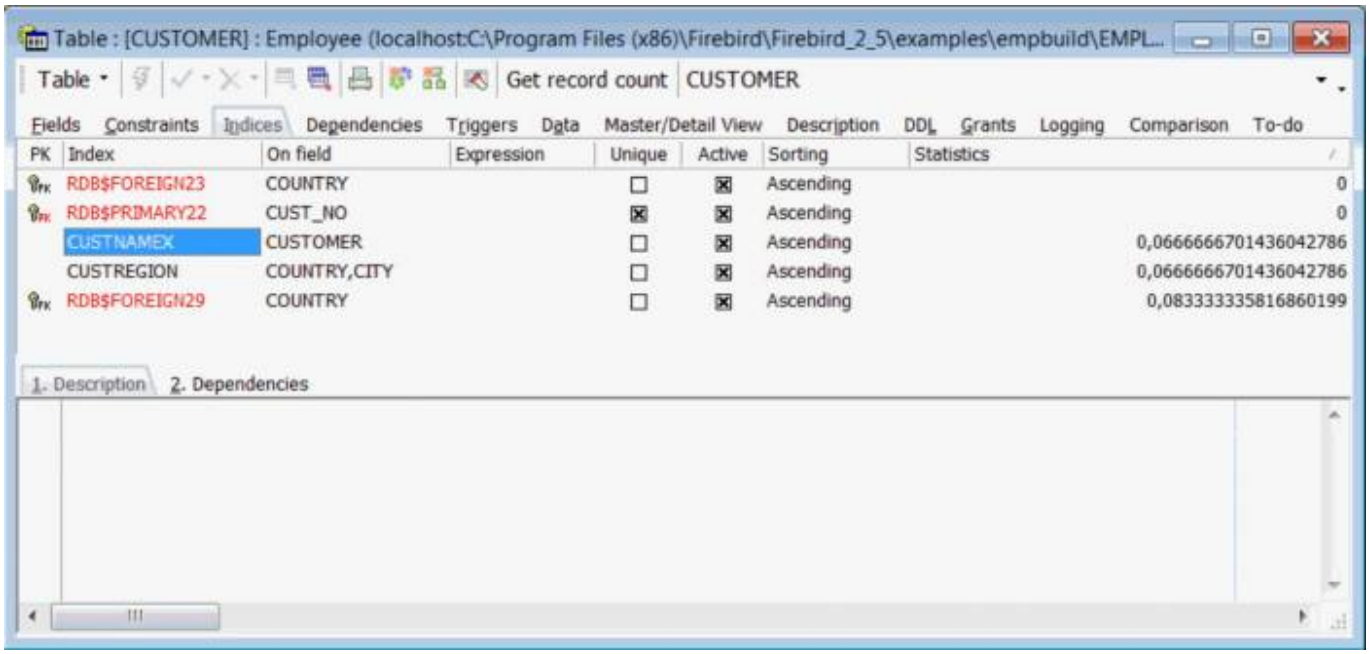
Indices should not be confused with keys. In the relational model, a key is used to organize data logically, so that specific rows can be identified. An index, however, is part of the table's physical structure on-disk, and is used to increase the performance of tables during queries. Indices are therefore not a part of the relational model. In spite of this indices are extremely important for relational database systems.

For columns defined with a primary key or a foreign key in a table, Firebird/InterBase® automatically generates a corresponding ascending index and enforces the uniqueness constraint demanded by the relational model.

An index can be defined in the IBExpert Index Editor:



or the IBExpert Table Editor (both editors are opened from the DB Explorer):



The *Dependencies* page displays any index dependencies that may exist.

The maximum number of 65 indices per table was been removed in Firebird 1.0.3, reintroduced at the higher level of 257 in Firebird 1.5, and removed once again in Firebird 2.0. Although there is no longer a “hard” ceiling, the number of indices attainable in practice is still limited by the database page size and the number of columns per index (please refer to the Firebird 2.0 Language Reference Update chapter, Maximum number of indices in different Firebird versions). However please be aware that under normal circumstances, even 50 indices is way too many and will drastically reduce mutation speeds. The maximum was raised to accommodate data-warehousing applications and the like, which perform lots of bulk operations during which indices are temporarily switched off.

If you wish to ascertain just how many indices already exist for individual tables in a database, query the following from the system table, RDB$INDICES:

```
select * from RDB$INDICES where RDB$INDICES.RDB$RELATION_NAME='MYTABLE'
```

or view the indices list under the Indices node in the DB Explorer.

System tables and indices can be viewed in the IBExpert DB Explorer by activating the *Show System Tables* and *Show System Indices* check options, found in the Database registration info on the Additional page.

Firebird 2.0 introduced indexing on expressions and increased the maximum length of index keys, which used to be fixed at 252 bytes, to 1/4 of the page size, i.e. varying from 256 to 4096. The maximum indexable string length in bytes is 9 less than the key length.

Indices are updated every time a new data set is inserted, or rather, the index-referenced field is updated. Firebird/InterBase® writes an additional second mini version of the data set in each index table. It may therefore make sense to temporarily deactivate indices on tables where there is a huge amount of data manipulation, if you find this is slowing the system down.

An index has a sequence e.g. when an ascending index is assigned to a field (default), and a descending select on this field is requested, Firebird/InterBase® does not sort using the ascending index. For this a second descending index needs to be specified for the same field.

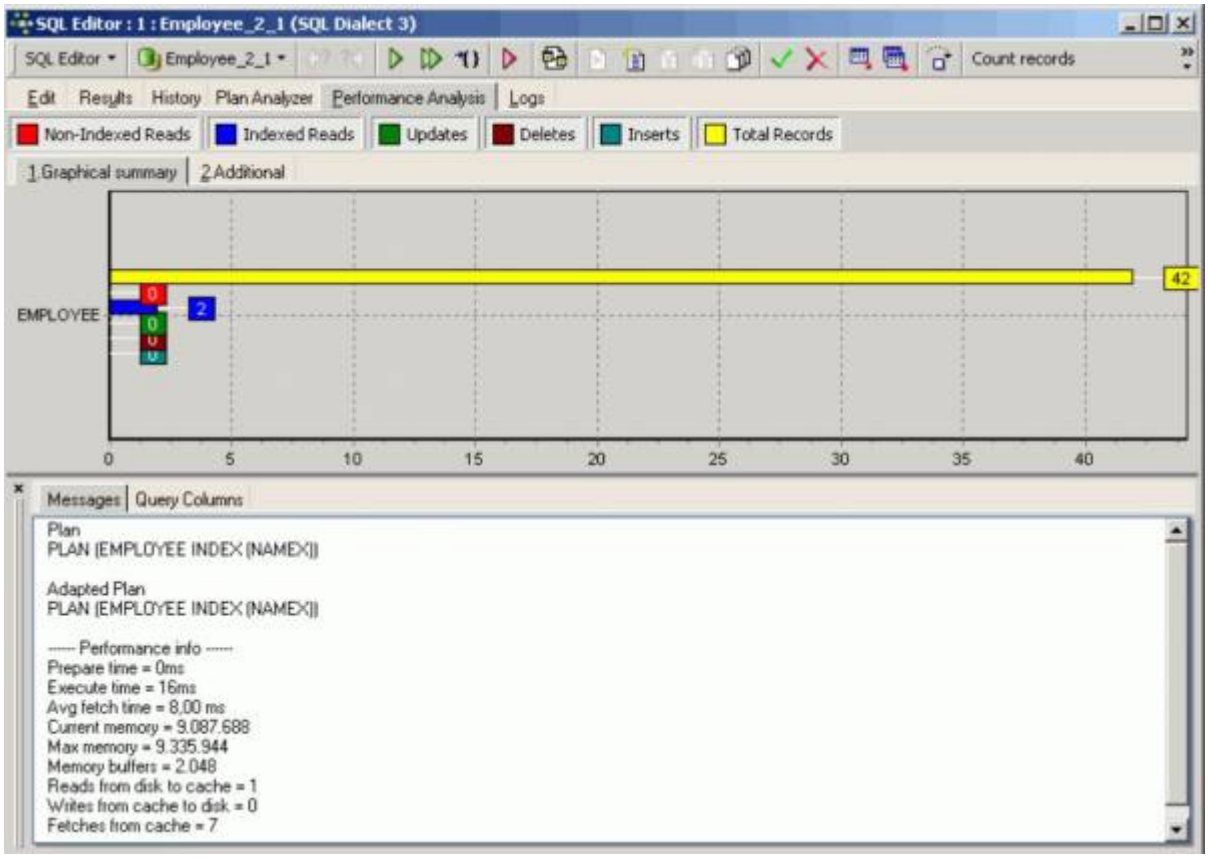
An index can be named as wished; consecutive numbers can even be used, as it is extremely rare that an index is named in SQL.

An index on two fields simultaneously only makes sense when both fields are to be sorted using ORDER BY, and this should only be used on relatively small quantities of results. An index on tables with few (<30) data sets is superfluous as it will make no difference to performance.

Firebird/InterBase® decides automatically which index it uses to carry out SELECT requests. On the Table Editor / Indices page under *Statistics*, it can be seen that the index with the lowest value has a higher uniqueness, and is therefore preferred by Firebird/InterBase® instead of other indices with a lower level of uniqueness. This is known as selectivity.

An index should only be used on fields which are really used frequently as sorting criteria (e.g. fields such as STREET and MALE/FEMALE are generally unimportant) or in a WHERE condition. In fact, on such fields as “Mr/Mrs” they can even be contraproductive. If a field is often used as a sorting criterion, a descending index should also be considered, e.g. in particular on DATE or TIMESTAMP fields. Care should also be taken that indexed CHAR fields are not larger than approximately 80 characters in length (with Firebird 1.5 the limit is somewhat higher). Indexing fields that are rarely queried is simply a waste of time; so define your indices wisely, on those fields where large data quantities are regularly filtered, sorted or grouped.

Indices can always be set after the database is actually in use, based on the performance requirements. For further details and examples please refer to Performance Analysis.



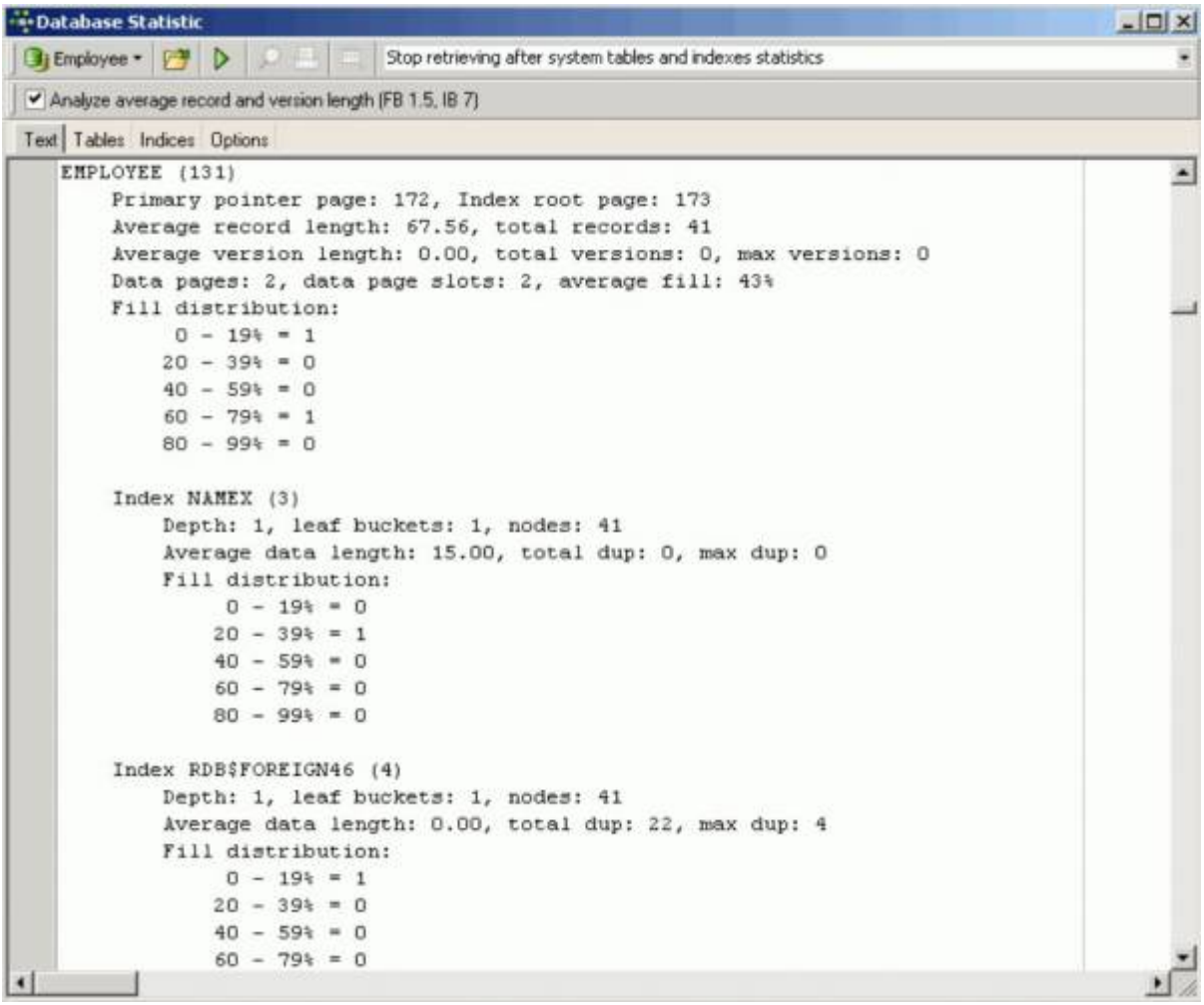
Index statistics and index selectivity

When a query is sent to the server, the Optimizer does not intuitively know how to process it. It needs further information to help it decide how to go about executing the query. For this it uses indices, and to decide which index is the best to use first, it relies on the index selectivity. The selectivity of an index is the best clue that the query plan has whether it should use a certain index or not. And when more than one index is available, it helps the Firebird server decide which index to use first. A good selectivity is close to 0 - it's the result of: 1/distinct values.

So the first thing the Optimizer does when it receives a query is to prepare the execution. It makes decisions regarding indices based solely upon their selectivity. Although the Optimizer only uses indices with a selectivity < 0.01 when there are no other appropriate indices available.

If you have an index on a field with only two distinct values (e.g. yes or no) in it, it will have a selectivity of 0.5. If your indexed field has 10 values, it will have a selectivity of 0.1. The higher the number of different values, the lower the selectivity number and the more suitable it is to be used as an index. Your benchmark is always your ID - the primary key, because that will always have complete unique values in it, and therefore the lowest selectivity.

Using the IBExpert Services menu item, Database Statistics the index statistics can be viewed. Leaf buckets display the number of registration leaves where Firebird/InterBase® can access immediately. An index depth of 2, for example, indicates that Firebird/InterBase® needs to perform two steps to obtain a result. Normally the value should not be higher than three.



The selectivity is only computed at the time of creation, or when the IBExpert menu item Recompute Selectivity or *Recompute All* is used (found directly in the Index Editor, IBExpert Services menu item, Database Statistics, in the Database menu, or in the right-click DB Explorer menu). Alternatively the

```
SET STATISTIC INDEX {INDEX_NAME}
```

command can be used in the SQL Editor to recompute individual indices. Only the creator of an index can use SET STATISTICS. Please note that SET STATISTICS does not rebuild an index; to rebuild an index, use ALTER INDEX.

The recalculation of selectivity can be automated to ensure the most efficient use of indices. Please refer to the Firebird administration using IBExpert chapter, Automating the recalculation of index statistics.

This is automatically performed during a database backup and restore, as it is not the index, but its definition that is saved, and so the index is therefore reconstructed when the database is restored.

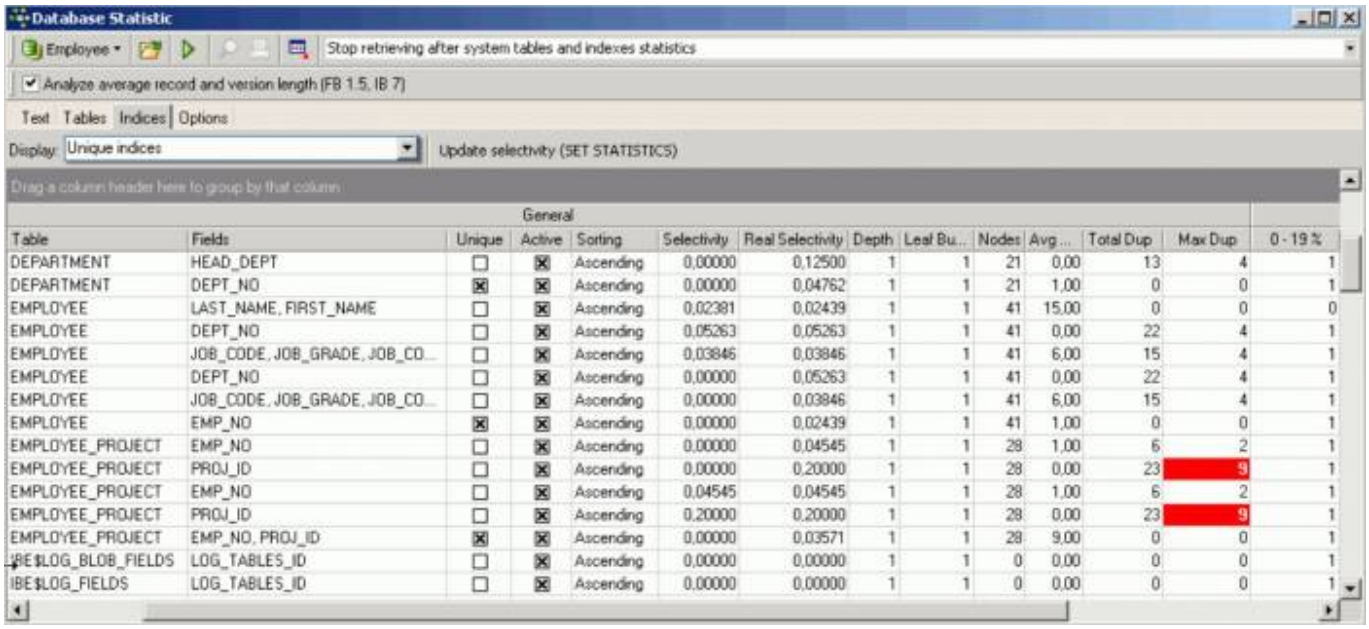


Table	Fields	Unique	Active	Sorting	Selectivity	Real Selectivity	Depth	Leaf Bu...	Nodes	Avg ...	Total Dup	Max Dup	0 - 19 %
DEPARTMENT	HEAD_DEPT	☐	☒	Ascending	0,00000	0,12500	1	1	21	0,00	13	4	1
DEPARTMENT	DEPT_NO	☒	☒	Ascending	0,00000	0,04762	1	1	21	1,00	0	0	1
EMPLOYEE	LAST_NAME, FIRST_NAME	☐	☒	Ascending	0,02381	0,02439	1	1	41	15,00	0	0	0
EMPLOYEE	DEPT_NO	☐	☒	Ascending	0,05263	0,05263	1	1	41	0,00	22	4	1
EMPLOYEE	JOB_CODE, JOB_GRADE, JOB_CO...	☐	☒	Ascending	0,03846	0,03846	1	1	41	6,00	15	4	1
EMPLOYEE	DEPT_NO	☐	☒	Ascending	0,00000	0,05263	1	1	41	0,00	22	4	1
EMPLOYEE	JOB_CODE, JOB_GRADE, JOB_CO...	☐	☒	Ascending	0,00000	0,03846	1	1	41	6,00	15	4	1
EMPLOYEE	EMP_NO	☒	☒	Ascending	0,00000	0,02439	1	1	41	1,00	0	0	1
EMPLOYEE_PROJECT	EMP_NO	☐	☒	Ascending	0,00000	0,04545	1	1	28	1,00	6	2	1
EMPLOYEE_PROJECT	PROJ_ID	☐	☒	Ascending	0,00000	0,20000	1	1	28	0,00	23	9	1
EMPLOYEE_PROJECT	EMP_NO	☐	☒	Ascending	0,04545	0,04545	1	1	28	1,00	6	2	1
EMPLOYEE_PROJECT	PROJ_ID	☐	☒	Ascending	0,20000	0,20000	1	1	28	0,00	23	9	1
EMPLOYEE_PROJECT	EMP_NO, PROJ_ID	☒	☒	Ascending	0,00000	0,03571	1	1	28	9,00	0	0	1
IBE$LOG_BLOB_FIELDS	LOG_TABLES_ID	☐	☒	Ascending	0,00000	0,00000	1	1	0	0,00	0	0	1
IBE$LOG_FIELDS	LOG_TABLES_ID	☐	☒	Ascending	0,00000	0,00000	1	1	0	0,00	0	0	1

The SQL plan used by the Firebird/InterBase® Optimizer merely shows how the server plans to execute the query.

If the developer wishes to override Firebird/InterBase®'s automatic index selection, and determine the index search sequence himself, this must be specified in SQL (please refer to Using the PLAN operator for further information).

For example, an index is created in the EMPLOYEE database:

```
CREATE INDEX EMPLOYEE_IDX1 ON EMPLOYEE(PHONE_EXT);
```

Then:

```
SELECT * FROM EMPLOYEE
   WHERE EMPLOYEE.PHONE_EXT='250'
   PLAN (EMPLOYEE INDEX (EMPLOYEE_IDX1));
```

Each index needs to be named and entered individually.

To eliminate an index from the plan +0 can be added in the query to the field where you wish the index to be ignored, thus denying the optimizer the ability to use that index for that particular query. This is much more powerful and flexible than deleting the index altogether, which prevents any use of it by the Optimizer in the future.

Indices should be prudently defined in a data structure, as not every index automatically leads to an acceleration in query performance. If in a table, for example, a column comprises data only with the value 0 or 1, an index could even slow performance down. A complex index structure can however have a huge influence upon insertion and alteration processes in the long run.

Please also refer to the *IBExpert documentation* chapter, Database Statistics, the Firebird 2.0.4 Release Notes chapter, Enhancements to indexing for improvements and new features in Firebird 2.0, the *Firebird 2.1 Release Notes* chapter, Indexing & optimizations and to the following subjects for further general information regarding indices.

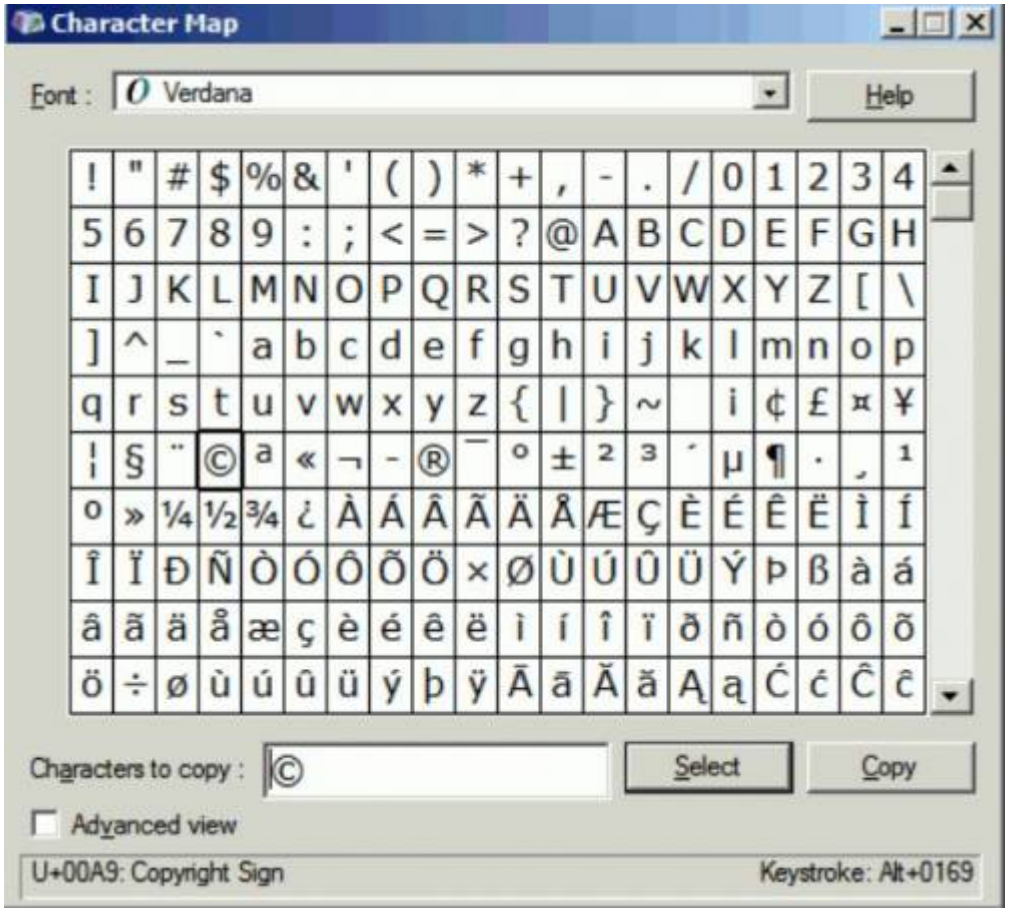
See also:

- Index Editor
- Table Editor / Index page
- Firebird administration using IBExpert: Index statistics
- SQL Editor / Plan Analyzer
- SQL Editor / Performance Analysis
- IBExpert Table Editor / Indices
- Firebird 2.0.4 Release Notes: Enhancements to Indexing
- Firebird 2.1 Release Notes: Indexing & optimizations
- CREATE INDEX
- Recompute selectivity of all indices
- Firebird for the database expert: Episode 1 - Indexes
- Recreating Indices 1
- Recreating Indices 2

back to top of page

Ascending index

An ascending index searches according to an ascending letter or numeric sequence, depending upon the defined character set (or, if no character set has been specified for the indexed field, the default character set).



Descending index

A descending index searches according to a descending letter or numeric sequence, depending upon the defined character set (or, if no character set has been specified for the indexed field, default character set).

UNIQUE indices allow NULLs

Since Firebird 1.5, in compliance with the SQL-99 standard, NULLs – even multiple – are allowed in columns that have a UNIQUE index defined on them. As far as NULLs are concerned, the rules for unique indices are exactly the same as those for unique keys.

back to top of page

Index Editor

Traditionally indices are created individually for tables in the IBExpert table editors. However the *Indices* node has the advantage of displaying all indices for a database, allowing you to, for example, quickly and securely deactivate or activate all or certain indices, without the toil of opening each object editor and searching the individual *Indices* pages.

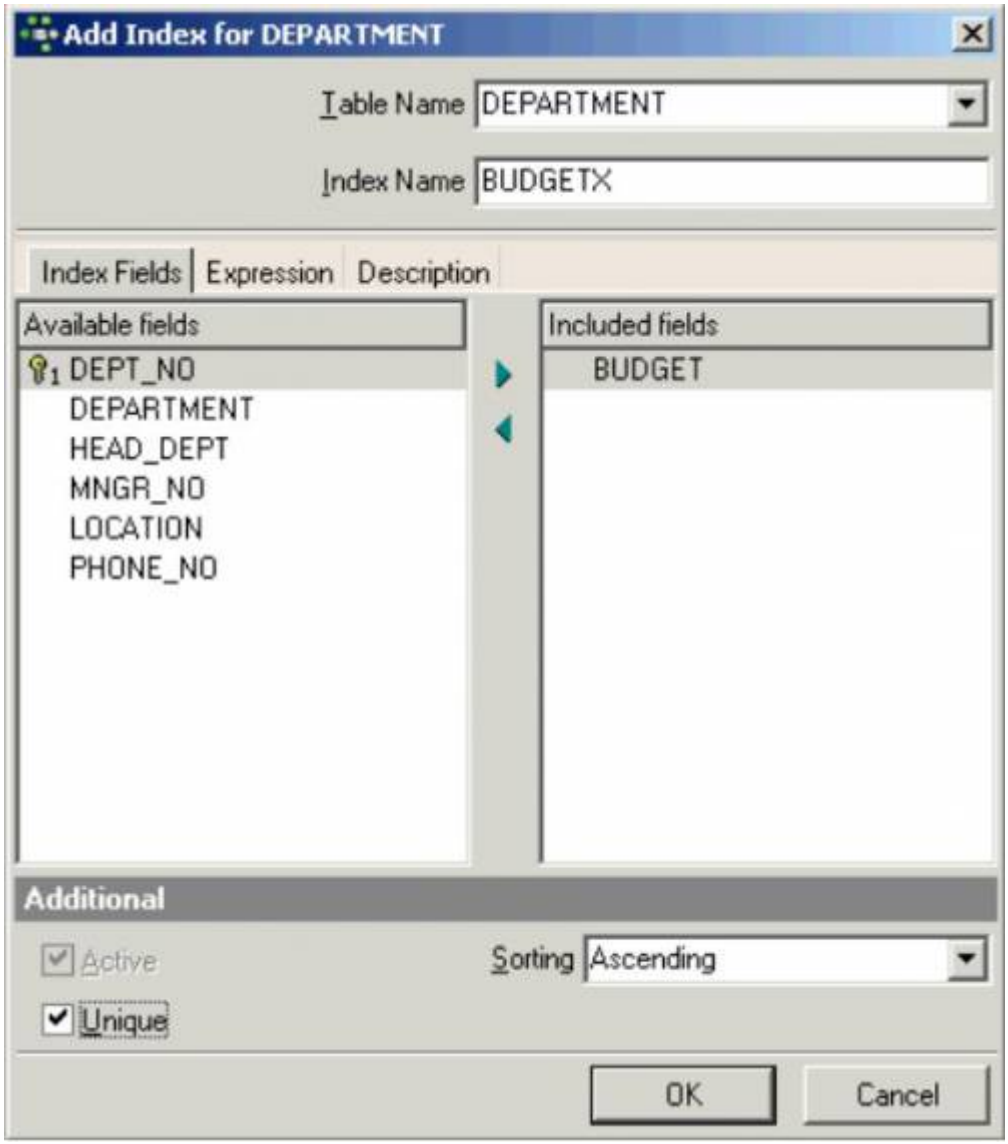
The Index node in the DB Explorer displays all indices in a database - both those manually created and system indices. System indices are displayed in red, if the system options have been flagged in the Register Database dialog (opened using the right mouse button Additional/DB Explorer). Firebird and InterBase® system indices always begin with the prefix RDB$.

back to top of page

New index/create index

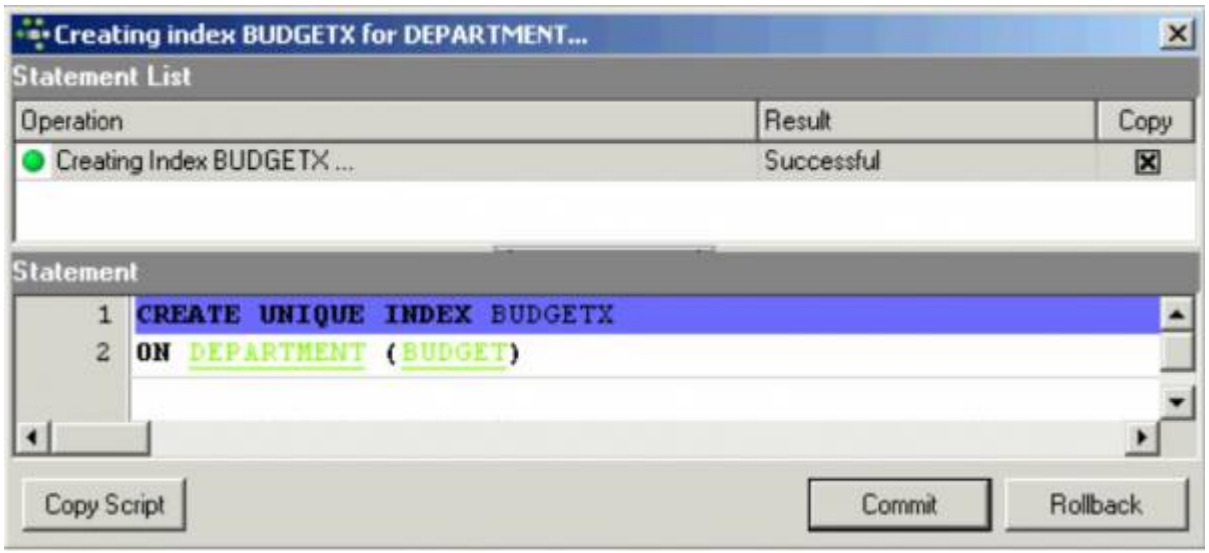
A new index can be created for a connected database using the DB Explorer right-click menu (or key combination [Ctrl + N], when the *Index* node or one of the indices of the relevant connected database is highlighted).

An *Add Index* for dialog appears.



Index Fields page

Select the table name from the drop-down list of the table you wish to place an index on, name the index and then select the field(s) you wish to index from the *Available fields* list on the left, using the blue arrow in the center panel to move to the right-hand *Included fields* list. Then specify the sorting order, check the *Unique* box if required, click the *OK* button and finally commit.



Expression page

New to Firebird 2.0: Instead of a column – or column list – you can now also specify a COMPUTED BY expression in an index definition. Expression indices will be used in appropriate queries, provided that the expression in the WHERE, ORDER BY or Data retrieval#GROUP BY|GROUP BY]] clause exactly matches the expression in the index definition.

Please refer to CREATE INDEX for further information and examples.

Description page

As with the majority of the IBExpert editors, the Index Editor's *Description* page can be used to insert, edit and delete text by the user as wished, enabling the database to be simply and quickly documented.

Don't forget to confirm and commit following any additions or amendments made on any of the Index Editor pages!

Those preferring hand-coding can of course create their indices in the SQL Editor using the following syntax:

```
CREATE [UNIQUE] [ASC[ENDING] | [DESC[ENDING]] INDEX indexname
   ON tablename
   { (colname [, colname ...]) | COMPUTED BY (expression) }
```

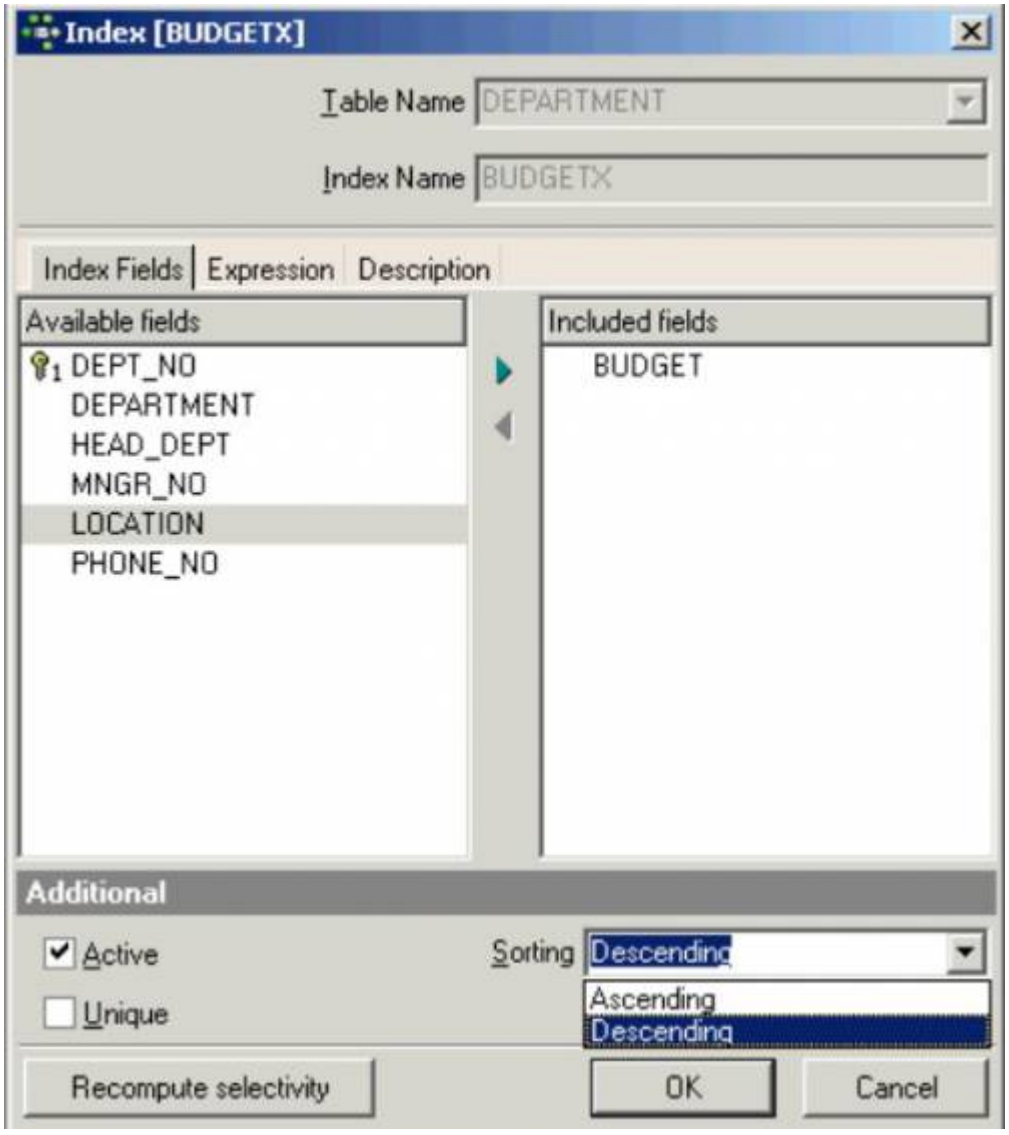
back to top of page

Alter index

An index can be altered in the Index Editor, opened by double-clicking on the index name in the DB Explorer. Alternatively use the DB Explorer´s right mouse-click menu item *Edit Index* or key combination [Ctrl + O].

The name of an index may not be altered. Should you wish to change an index name, you must drop and recreate the index. Attributes such as *Fields included*, *unique*, *sorting order* and *active* may be added, altered or deleted.

An index should be deactivated when, for example, a large number of data sets are to be added, as an active index would recompute the index each time a data set is input. By deactivating the index, and then reactivating after all the data has been input, the index is only recomputed once.



The *Recompute selectivity* button allows you to quickly and simply recompute the index's selectivity, maximizing the performance of any changes you have made. This feature is described in detail in the Recompute selectivity of all indices chapter.

This can also be done simply and directly on the Table Editor / Indices page, by checking or unchecking the relevant boxes in the Status column, then compiling, using the respective Editor icon

or [Ctrl + F9], and finally committing.

Using SQL it is only actually possible to alter the ACTIVE/INACTIVE status.

The SQL syntax is:

```
ALTER INDEX <index_name> ACTIVE | INACTIVE
```

If you paid attention to the IBExpert *Compile* dialog whilst compiling your index alterations in the Index Editor, you will have noticed that in order to enforce your desired changes,IBExpert does none other than drop the index and then recreate it incorporating the new properties specified.

If an index is in use, ALTER INDEX does not take effect until the index is no longer in use.

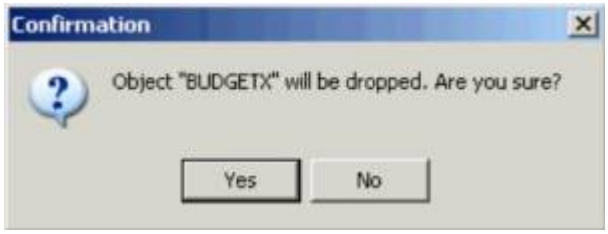
An index can be altered by its creator, the SYSDBA user, and any users with operating system root privileges.

back to top of page

Drop index/delete index

As indices can be quickly and simply deactivated by simply checking a box, it is hardly ever necessary to drop an index. However, should you ever feel the need to spring clean your database, it is possible. Only user-defined indices can be dropped. As the only alterations permitted on indices are activation and deactivation, when coding manually indices often need to be dropped and then subsequently recreated, in order to alter certain index information such as indexed columns, sort direction or uniqueness constraints. In IBExpert alterations can be quickly and easily carried out in the Index Editor.

To drop an index use the DB Explorer right-click menu and select the menu item *Drop Index* or [Ctrl + Del]. IBExpert asks for confirmation:



Alternatively when in the Table Editor / Indices page, simply mark the index to be dropped and then right-click and select the menu item *Drop Index <INDEXNAME>* or use the [DEL] key. Finally commit or roll back.

Using SQL the syntax is:

```
DROP INDEX Index_Name
```

An index in use is not dropped until it is no longer in use. DROP INDEX cannot be used for system-

generated indices on primary or foreign keys, or on columns with a uniqueness constraint in the table definition.

An index can be dropped by its creator, the SYSDBA user, or any user with operating system root privileges.

From:
http://ibexpert.com/docu/ - **IBExpert**

Permanent link:
http://ibexpert.com/docu/doku.php?id=01-documentation:01-13-miscellaneous:glossary:index

Last update: **2023/08/17 03:32**