Blob (Binary Large OBject)

A blob is a data type storing large binary information (Binary Large OBject).

Blobs can contain any binary or ASCII information, for example, large text files, documents for data processing, CAD program files, graphics and images, videos, music files etc.

Blobs are defined as table columns. Their memory size is almost unlimited as they can be stored across several pages. This assumes however that a sufficient database page size has been specified. For example, using a 1k page, the blob may not exceed 0.5 GB, using a 4k page size, the blob size is limited to 8GB.

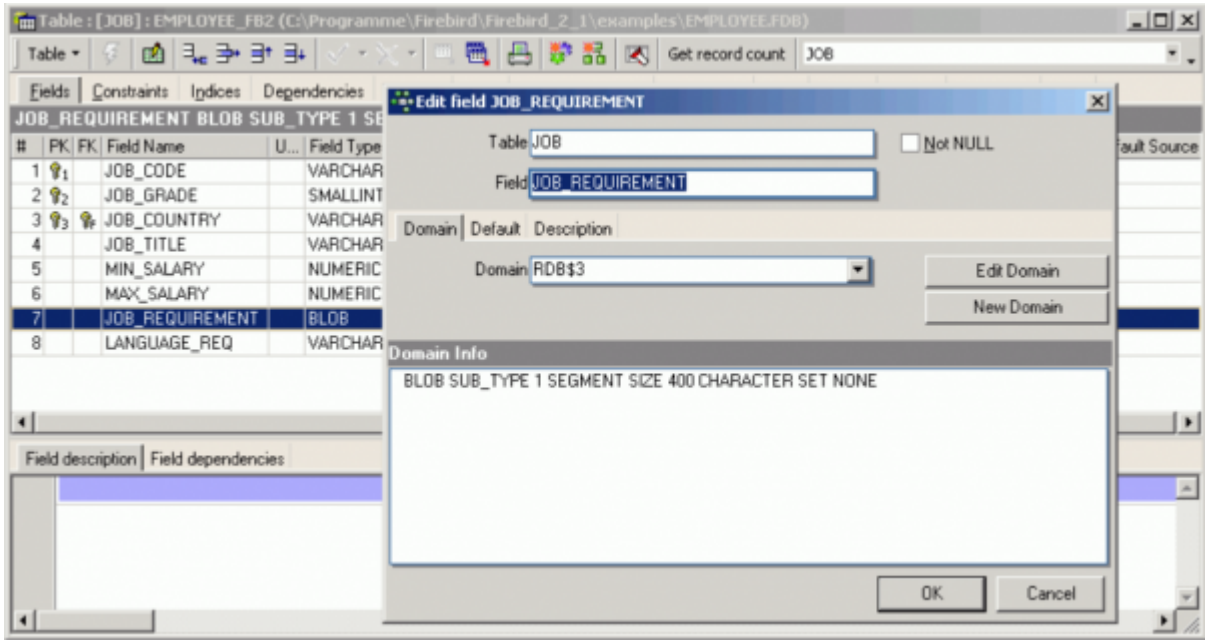
The ability to store such binary data in a database provides a high level of data security, data backup, version management, categorization and access control.

The advantage of blob text fields over VARCHAR fields (e.g. VARCHAR (32000)) is that a network protocol transfers all 32,000 VARCHAR characters when using an ISDN connection (analog lines compress the data to an extent). With a blob field, only the actual file size is transferred. Although - since Borland InterBase® version 6.5/7 this disadvantage with VARCHAR data type transfer has been solved, i.e. in these newer InterBase® versions the full VARCHAR length including spaces is no longer transferred each time across the network. However, even here, blobs are still more effective when working with such large data sizes.

Since Firebird 2.1 text blobs can masquerade as long VARCHARs. At various levels of evaluation, the Firebird engine now treats text blobs that are within the 32,765-byte size limit as though they were VARCHAR. String functions like CAST, LOWER, UPPER, TRIM and SUBSTRING will work with these blobs, as well as concatenation and assignment to string types. You can even access blob contents using CONTAINING and LIKE. ORDER BY however should not be used on blobs, as it sorts and displays the blob fields in the order that they were created und not according to content. Please refer to the Firebird 2.1 Release Notes for further information.

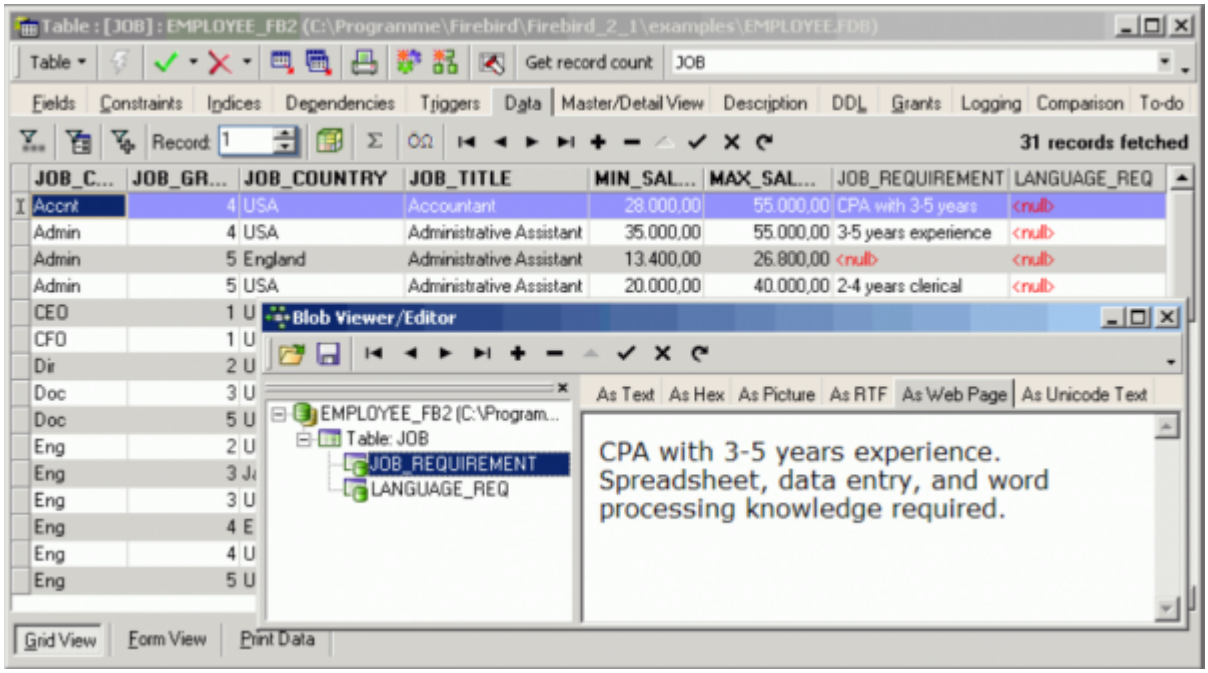
Firebird/InterBase® supports quick and efficient algorithms for reading, writing and updating blobs. The user can manipulate blob processing with blob routines - also called blob filters. These filters are ideal tools for the compression and translation of blobs, depending upon the application requirements.

Blobs can be specified using the IBExpert DB Explorer or the IBExpert SQL Editor.



Blob specification includes the subtype, segment size and, if wished, the character set.

When the Data View (i.e. Data page) in the Table Editor is selected, and the table shown contains a blob column, IBExpert can display the blob content of a selected data set as text (also as RTF), hex, images and web pages using the IBExpert menu item Tools / Blob Viewer/Editor.



It is important when using blobs in a database, to consider the database page size carefully. Blobs are created as part of a data row, but because a blob could be of unlimited length, what is actually stored with the data row is a BlobID, the data for the blob is stored separately on special blob pages elsewhere in the database.

The BlobID is an 8 byte value that allows Firebird/InterBase® to uniquely identify a blob and locate it. The BlobIDs can be either temporary or permanent; a temporary blob is one which has been created, but has not yet been stored as part of a table, permanent blobs have been stored in a table. The first 4 bytes represent the relation ID for the blob (like data rows, blobs are bound to a table), the second

four bytes represent the ID of the blob within the table. For temporary blobs the relation ID part is set to 0.

A blob page stores data for a blob. For large blobs, the blob page could actually be a blob pointer page, i.e. be used to store pointers to other blob pages. For each blob that is created a blob record is defined, the blob record contains the location of the blob data, and some information about the blob's contents that will be useful to the engine when it is trying to retrieve the blob. The blob data could be stored in three slightly different ways. The storage mechanism is determined by the size of the blob, and is identified by its level number (0, 1 or 2). All blobs are initially created as level 0, but will be transformed to level 1 or 2 as their size increases.

A level 0 blob, is a blob that can fit on the same page as the blob header record, for a data page of 4096 bytes, this would be a blob of approximately 4052 bytes (page overhead - slot - blob record header).

Although the documentation states that the segment length does not affect the performance of Firebird/InterBase®, the actual physical size of a blob, or its segment length can become useful in trying to improve I/O performance for the blob, especially if you can size the segment (typically) or blob to a page.

This is especially true if you plan to manipulate the blob using certain low level Firebird/InterBase® blob calls. When a blob is too large to fit on a single page (level 1), and the data will be stored on one or more blob data pages, then the initial page of the blob record will hold a vector of blob page numbers.

A level 2 blob occurs when the initial page of the blob record is not big enough to contain the vector of all the blob data page numbers. Then Firebird/InterBase® will create blob pointer pages, i.e. multiple vector pages that can be accessed from the initial blob header record, that now point to blob data pages.

The maximum size of a level 2 blob is a product of the maximum number of pointer pages, the number of data pages per pointer page, and the space available on each data page.

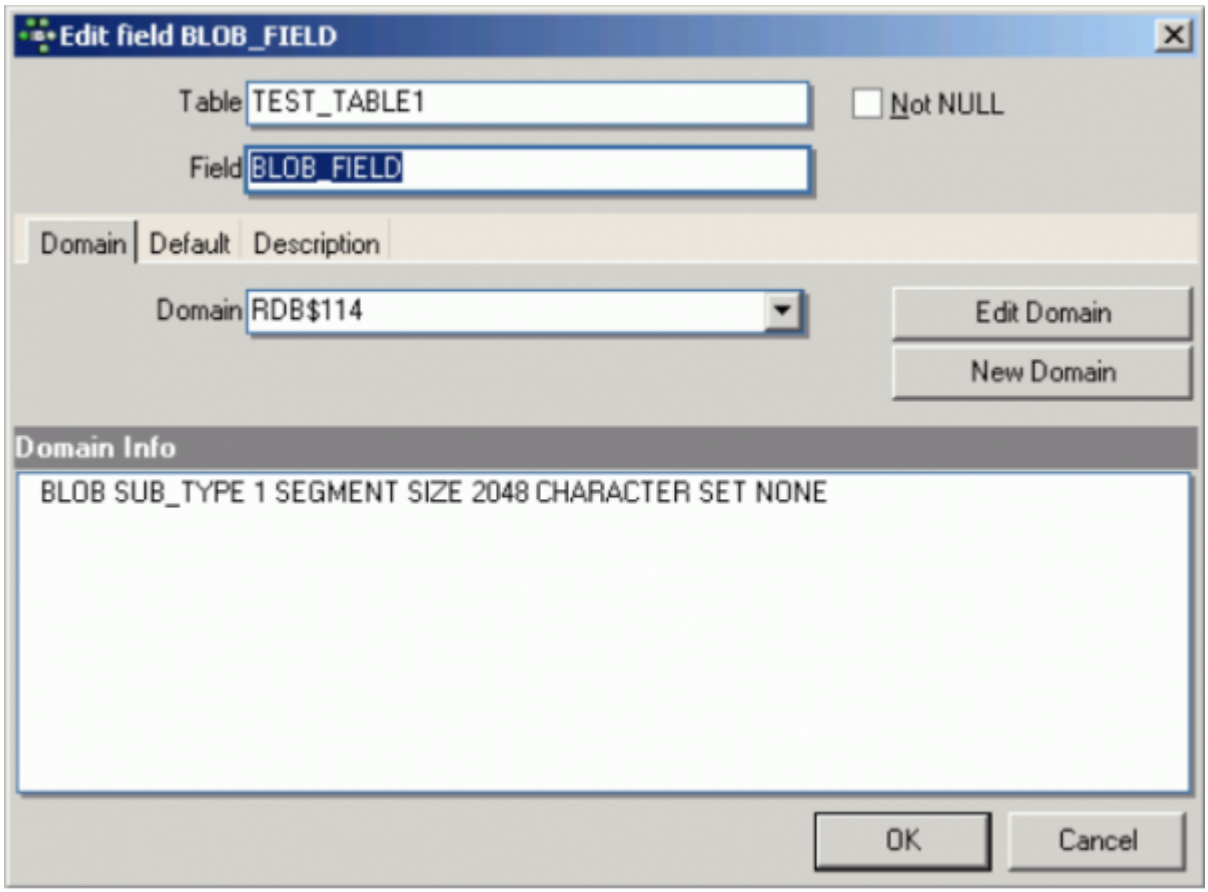
Max Blob Size:

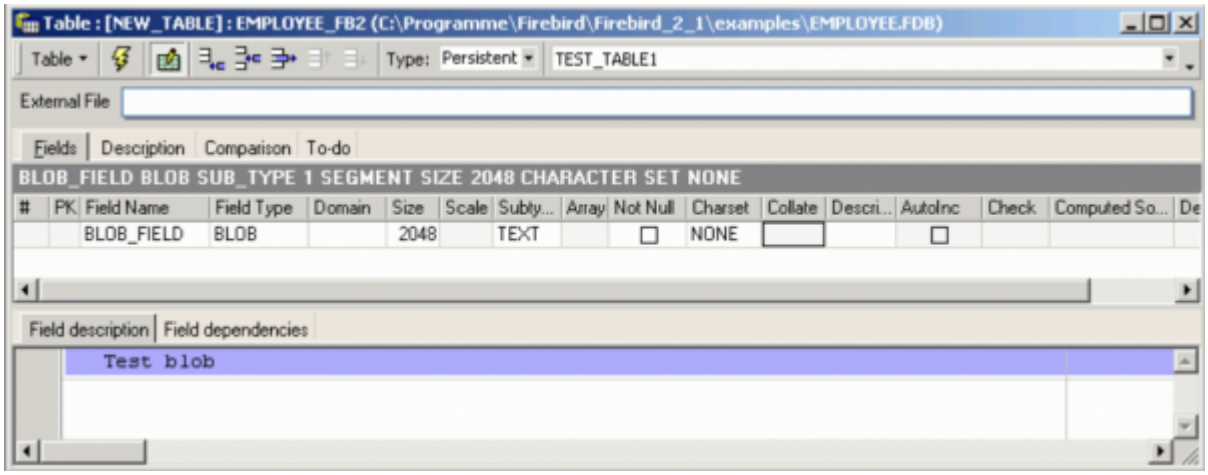
- 1Kb page size ⇒ 64 Mb
- 2Kb page size ⇒ 512 Mb
- 4Kb page size ⇒ 4 Gb
- 8Kb page size ⇒ 32 Gb
- 16kb page size ⇒ Big enough 🙂.

We would like to thank Paul Beach of IBPhoenix, for allowing us to reproduce excerpts of his session, *Using and Understanding Blobs*, held at the European Firebird Conference 2003.

Segment size

Segment sizes are specified for blob fields. This can be done using the Domain Editor or the Table Editor (started from the IBExpert DB Explorer).





A blob segment size can be defined, to increase the performance when inputting and outputting blob data. This should roughly correspond to the data type size. With a memo field, for example, for brief descriptions which could however, in individual cases, be considerably longer, the segment length could be defined as 100 bytes, whereby the blob data type is processed in 100 byte blocks.

When processing videos or large graphics in the database, a large segment length should be selected. The maximum length is 65536 bytes. This is because all blob contents are stored in blocks, and are fetched via these blocks. A typical segment size from the old days is 80 (because 80 characters fit onto one monitor line).

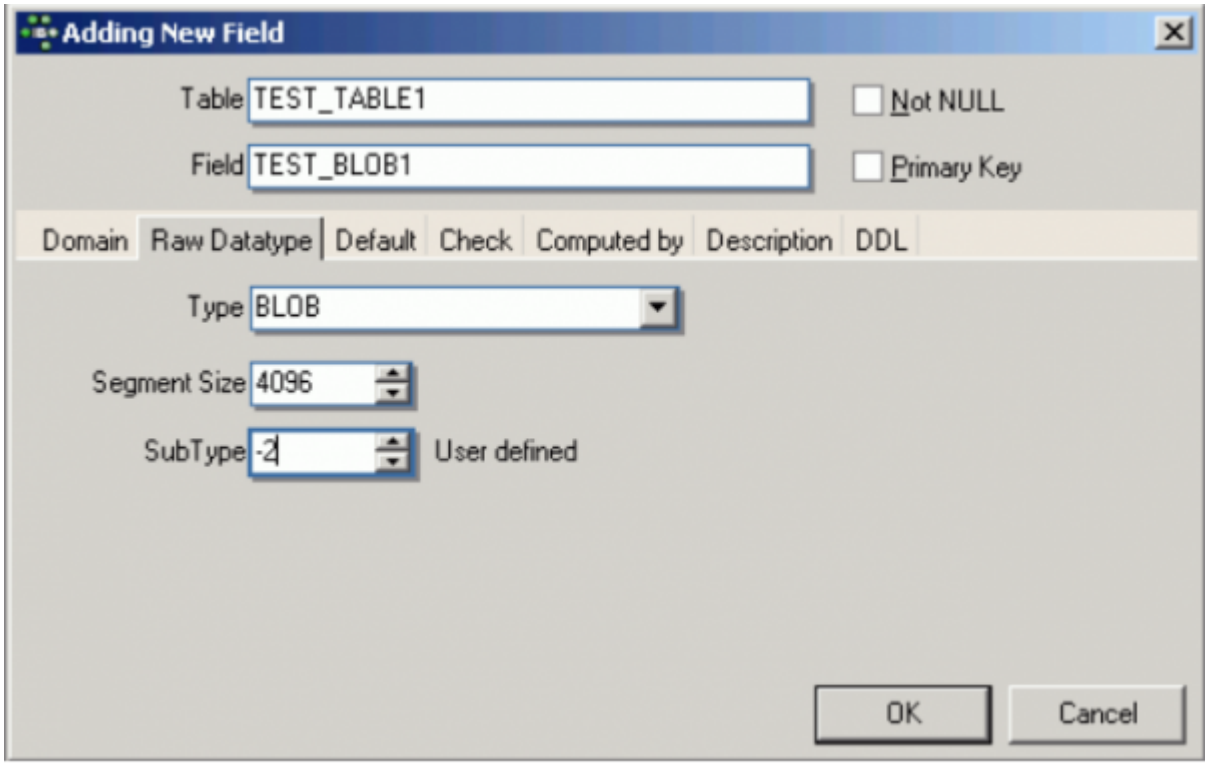
When a blob is extracted, the Firebird/InterBase® server reads the number of segments that the client has requested. As the server always selects complete blocks from the database, this value can in effect be ignored on modern powerful computers. 2048 is recommended as a standard since version InterBase® 6.

Subtype

Subtypes are specified for blobs. They are used to categorize the data type when defining blobs. A subtype is a positive or negative numerical value, which indicates the type of blob data. The following subtypes are predefined in Firebird/InterBase®:

Subtype	Meaning
0	Standard blob, non-specified binary data (image, video, audio, whatever)
1	Text blob, e.g. memo fields (basic character functions work)
2	BLR (used for definitions of Firebird procedures, triggers, etc.)
Text	Alternative for defining subtype 1
Positive value	Reserved for InterBase®
Negative value	User-defined blob subtypes

User applications should only use subtypes 0, 1 and negative values.



Blob fields can be specified using the Domain Editor or the Table Editor (started from the IBExpert DB Explorer).

The specification of a user-defined blob subtype has no effect upon Firebird/InterBase®, as the Firebird/InterBase® server treats all blob fields the same, i.e. it simply stores the data and delivers it to the client program when required.

The definitions are however required by the client programs in order to display the blob content correctly. For example, SUB_TYPE -200 could be defined as a subtype for GIF images and SUB_TYPE -201 as a subtype for JPG images.

Subtype specification is optional; if nothing is specified, Firebird/InterBase® assumes 0 = binary data.

As mentioned above, under the menu item Tools, the IBExpert Blob Viewer/Editor can display blob

contents as text, hex, images, RTF, Unicode and web pages.

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

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

Last update: **2023/08/13 20:30**