



# **Firebird 2.1 Beta 2 Release Notes**

Helen Borrie (Collator/Editor)

1 Oct 2007 - Document v. 210\_20 - for Firebird 2.1 Beta 2

---

## **Firebird 2.1 Beta 2 Release Notes**

1 Oct 2007 - Document v. 210\_20 - for Firebird 2.1 Beta 2  
Helen Borrie (Collator/Editor)

---

---

---

## Table of Contents

1. General Notes .....	1
Bug Reporting .....	1
Documentation .....	2
2. New in Firebird 2.1 .....	3
New Features Implemented .....	3
Database Triggers .....	3
SQL and Objects .....	3
Procedural SQL .....	4
Security .....	4
International Language Support .....	4
Platform Support .....	5
Administrative .....	5
Remote Interface .....	5
3. Global Improvements in Firebird 2.1 .....	6
Remote Interface Improvements .....	6
API Changes .....	7
XSQLVAR .....	7
Optimization .....	7
Optimization for Multiple Index Scans .....	7
Optimize sparse bitmap operations .....	7
Configuration and Tuning .....	7
Increased Lock Manager Limits & Defaults .....	7
Page sizes of 1K and 2K Deprecated .....	8
Enlarge Disk Allocation Chunks .....	8
Bypass Filesystem Caching on Superserver .....	8
Other Global Improvements .....	9
Garbage Collector Rationalisation .....	9
Immediate Release of External Files .....	9
Synchronization of DSQL metadata cache objects in Classic server .....	9
BLOB Improvements .....	9
Type Flag for Stored Procedures .....	9
4. New Configuration Parameters and Changes .....	11
MaxFileSystemCache .....	11
5. Administrative Features .....	12
Monitoring Tables .....	12
The Concept .....	12
Scope and Security .....	12
Metadata .....	13
Usage .....	15
Cancel a Running Query .....	16
More Context Information .....	17
6. SQL Language Enhancements .....	18
Data Definition Language (DDL) .....	18
Database Triggers .....	18
Global Temporary Tables .....	19
BLOB Subtype 1 Compatibility with VarChar .....	21
Views Enhancements .....	21
SQL2003 compliance for CREATE TRIGGER .....	22

SQL2003 Compliant Alternative for Computed Fields .....	22
Data Manipulation Language (DML) .....	22
Common Table Expressions .....	23
The LIST Function .....	26
The RETURNING Clause .....	27
UPDATE OR INSERT Statement .....	28
New JOIN Types .....	29
INSERT with defaults .....	30
Make RDB\$DB_KEY in outer joins return NULL when appropriate .....	31
Data Type of an Aggregation Result .....	31
Built-in Functions .....	32
Changes in DSQL Parsing .....	33
Procedural Language Extensions (PSQL) .....	34
Domains in PSQL .....	34
COLLATE in Stored Procedures and Parameters .....	35
Optimization .....	35
7. International Language Support (INTL) .....	36
The CREATE COLLATION Command .....	36
ICU Character Sets .....	37
Registering an ICU Character Set Module .....	37
The UNICODE Collations .....	38
Specific Attributes for Collations .....	38
Collation Changes .....	39
Supported Character Sets .....	40
Character Sets Added .....	40
Metadata Text Conversion .....	40
Repairing Your Metadata Text .....	41
8. Utility Programs .....	43
New Command-line Utility fbvcmgr .....	43
Using fbvcmgr .....	43
Improvements to Utilities .....	46
Utilities Support for Database Triggers .....	46
gbak .....	46
isql .....	46
Services Manager .....	47
Builds and Installs .....	47
9. Security .....	48
Using Windows Security to Authenticate Users .....	48
SQL Privileges .....	48
Administrators .....	48
Configuration Parameter “Authentication” .....	49
Forcing Trusted Authentication .....	49
Other Security Improvements .....	50
isc_service_query() wrongly reveals full database file spec .....	50
Any user could view the server log through the Services API .....	50
10. Bugs Fixed .....	51
Firebird 2.1 Beta 2 .....	51
Core Engine/DSQL .....	51
Server Crashes .....	52
Windows-Specific .....	53
Data Definition Language (DDL) .....	53
Data Manipulation Language (DML) .....	53

---

Procedural Language (PSQL) .....	54
Remote Interface .....	55
API .....	55
International Language Support (INTL) .....	56
Database Monitoring/Admin .....	56
Security .....	57
Command-line Utilities .....	57
Firebird 2.1 Beta 1 .....	58
Core Engine/DSQL .....	59
Server Crashes .....	59
Win32-Specific .....	59
POSIX-Specific .....	60
Data Definition Language (DDL) .....	60
Data Manipulation Language (DML) .....	60
Procedural Language (PSQL) .....	61
Remote Interface .....	61
Security .....	61
Utilities .....	62
Building/Installers .....	63
Fixed Regressions .....	63
Not Fixed .....	63
Appendix A: .....	64
New Built-in Functions, Firebird 2.1 .....	64
Appendix B: .....	72
INTL Character Sets .....	72
Narrow Character Sets .....	72
ICU Character Sets .....	72

---

## Chapter 1

# General Notes

This is the second Beta release of Firebird 2.1. Thanks to all who have field-tested Beta 1 and reported problems. As you can tell from the bug-fix list, you found plenty for us to do!

As with all alphas and betas, we ask you to test this one rigorously in the field on **dispensable copies** of your favourite databases! Do not be tempted to put Firebird 2.1 Beta builds into your production environment, however sweet it looks at first sight.

### Important

#### From the QA Team

With Beta 2, we're entering final phase of Firebird 2.1 development. This version is feature-complete and, all being well, the next release is hoped to be the first Release Candidate. Developers are now focused on optimization and bug fixing and your feedback is needed even more than before.

The 2.1 release has many interesting new features that you can play with, like database triggers, temporary and monitoring tables, common table expressions, recursive queries and dozens of new inbuilt functions. We encourage you to see what you can achieve with these new features and let us know about any deficiency.

You are enthusiastically invited to post to the firebird-devel list good descriptions of any bugs or beasts you encounter, or post bug reports directly to our [Issue Tracker](#).

To help smoothe the transition from older versions, we encourage you to test this release with your applications and stress it with real-world data and loads, enabling as many hidden regressions or performance issues as possible to surface and be fixed before final release.

#### Tell us it's OK, too!

We're interested in your feedback even if you don't find any issues. "Positive" feedback helps to shorten the release cycle because, without it, it is hard to gauge how much the build is being tested in the field, in terms of both scale and functionality. The "quality index" is estimated using download count, direct feedback, hearsay, development stage, and such. If the positive feedback doesn't come in, the only way we can compensate for it is to lengthen the release cycle.

So don't hesitate to share your successful test results with us. You can send your comments and findings to *pcisar AT users DOT sourceforge DOT net*

Within reason, you can also ask support questions in firebird-devel but please restrict the questions to matters concerning the new v.2.1 features. The firebird-devel list is *not* a support forum for Firebird newbies!

On the other hand, support questions about Firebird 2.1 Betas (or any other alpha or beta builds) are not welcome in the firebird-support list, so we ask you to respect our forum rules.

## Bug Reporting

- If you think you have discovered a new bug in this release, please make a point of reading the instructions for bug reporting in the article [How to Report Bugs Effectively](#), at the Firebird Project website.

- If you think a bug fix hasn't worked, or has caused a regression, please locate the original bug report in the Tracker, reopen it if necessary, and follow the instructions below.

Follow these guidelines as you attempt to analyse your bug:

1. Write detailed bug reports, supplying the exact build number of your Firebird kit. Also provide details of the OS platform. Include reproducible test data in your report and post it to our [Tracker](#).
2. You are warmly encouraged to make yourself known as a field-tester of this alpha by subscribing to the [field-testers' list](#) and posting the best possible bug description you can.
3. If you want to start a discussion thread about a bug or an implementation, please do so by subscribing to the [firebird-devel list](#). In that forum you might also see feedback about any tracker ticket you post regarding this alpha.

## Documentation

You will find README documents for many of the new features in both this Beta and for Firebird v.2.0.x. in the doc sub-directory of your Firebird 2.1 Beta 1 installation.

An automated "Release Notes" page in the Tracker provides lists and links for all of the Tracker tickets associated with this alpha. [Use this link](#).

--*The Firebird Project*



# New in Firebird 2.1

## New Features Implemented

The following new features have been implemented since the 2.0.x releases.-

### *Database Triggers*

Newly implemented “database triggers” are user-defined PSQL modules that can be designed to fire in various connection-level and transaction- level events. See [Database Triggers](#).

### *SQL and Objects*

#### *Global Temporary Tables*

SQL standards-compliant global temporary tables have been implemented. These pre-defined tables are instantiated on request for connection-specific or transaction-specific use with non-persistent data, which the Firebird engine stores in temporary files. See [Global Temporary Tables](#).

#### *Common Table Expressions, Recursive DSQL Queries*

Standards-compliant common table expressions, which make dynamic recursive queries possible, are introduced. See [Common Table Expressions](#).

#### *RETURNING Clause*

Optional RETURNING clause for all singleton operations update, insert and delete operations. See [RETURNING Clause](#).

#### *UPDATE OR INSERT Statements for MERGE Functionality*

Now you can write a statement that is capable of performing either an update to an existing record or an insert, depending on whether the targeted record exists. See [UPDATE OR INSERT Statement](#).

#### *LIST() function*

A new aggregate function LIST(<SOMETHING>) retrieves all of the SOMETHINGs in a group and aggregates them into a comma-separated list. See [LIST Function](#).

## ***Lots of New Built-in Functions***

Built-in functions replacing many of the UDFs from the Firebird- distributed UDF libraries. For a full list with examples, see [Built-in Functions](#).

At various levels of evaluation, the engine now treats text BLOBs that are within the 32,765-byte size limit as though they were varchars. Now functions like cast, lower, upper, trim and substring will work with these BLOBs, as well as concatenation and assignment to string types. See [Text BLOB Compatibility](#).

## ***Procedural SQL***

### ***Domains for Defining PSQL Variables and Arguments***

PSQL local variables and input and output arguments for stored procedures can now be declared using domains in lieu of canonical data types. See [Domains in PSQL](#).

### ***COLLATE in Stored Procedures and Parameters***

Collations can now be applied to PSQL variables and arguments. See [COLLATE in Stored Procedures](#).

### ***Enhancement to PSQL error stack trace***

V. Horsun

[Feature request CORE-970](#)

A PSQL error stack trace now shows line and column numbers.

## ***Security***

### ***Windows Security to Authenticate Users***

From Firebird 2.1 onward, Windows “Trusted User” security can be applied for authenticating Firebird users on a Windows host. See [Windows Trusted User Security](#).

## ***International Language Support***

### ***The CREATE COLLATION Command***

The DDL command `CREATE COLLATION` has been introduced for implementing a collation, obviating the need to use the script for it. See [CREATE COLLATION Command](#).

## ***Unicode Collations Anywhere***

Two new Unicode collations can be applied to any character set using a new mechanism. See [UNICODE Collations](#).

## ***Platform Support***

### ***Ports to Windows 2003 64-bit***

D. Yemanov

[Feature request CORE-819](#) and [CORE-682](#)

64-bit Windows platform (AMD64 and Intel EM64T) ports of Classic, Superserver and Embedded models.

## ***Administrative***

### ***Database Monitoring via SQL***

Implementation of run-time database snapshot monitoring (transactions, tables, etc.) via SQL over some new system tables that use the new global temporary tables. See [Monitoring Tables](#).

Included in the set of tables is one named MON\$DATABASE that provides a lot of the database header information that could not be obtained previously via SQL: such details as the on-disk structure (ODS) version, SQL dialect, sweep interval, OIT and OAT and so on.

It is possible to use the information from the monitoring tables to cancel a rogue query. See [Cancel a Running Query](#).

### ***More Context Information***

Context information providing the server engine version has been added, for retrieving via SELECT calls to the RDB\$GET\_CONTEXT function. See [More Context Information](#).

## ***Remote Interface***

The remote protocol has been slightly improved to perform better in slow networks once drivers are updated to utilise the changes. Testing showed that API round trips were reduced by about 50 percent, resulting in about 40 per cent fewer TCP round trips. See [Remote Interface Improvement](#).

# Global Improvements in Firebird 2.1

Some global improvements and changes have been implemented in Firebird 2.1, as engine development moves towards the architectural changes planned for Firebird 3.

## Remote Interface Improvements

V. Horsun, D. Yemanov

### [Feature request CORE-971](#)

The remote protocol has been slightly improved to perform better in slow networks. In order to achieve this, more advanced packets batching is now performed, along with some buffer transmission optimizations. In a real world test scenario, these changes showed about 50 per cent fewer API round trips, thus incurring about 40 per cent fewer TCP roundtrips.

In Firebird 2.1 the remote interface limits the packet size of the response to various `isc_XXX_info` calls to the real used length of the contained data, whereas before it sent the full specified buffer back to the client buffer, even if only 10 bytes were actually filled. Firebird 2.1 remote interface sends back only 10 bytes in this case.

Some of our users should see a benefit from the changes, especially two-tier clients accessing databases over the Internet.

The changes can be summarised as

- a. Batched packets delivery. Requires both server and client of version v2.1, enabled upon a successful protocol handshake. Delays sending packets of certain types which can be deferred for batched transfer with the next packet. (Allocate/deallocate statement operations come into this category, for example.)
- b. Pre-fetching some pieces of information about a statement or request and caching them on the client side for (probable) following API calls. Implemented on the client side only, but relies partly on the benefits of reduced round trips described in (a).

It works with any server version, even possibly providing a small benefit for badly written client applications, although best performance is not to be expected if the client is communicating with a pre-V.2.1 server.

- c. Reduced information responses from the engine (no trailing zeroes). As the implementation is server-side only, it requires a V.2.1 server and any client. Even old clients will work with Firebird 2.1 and see some benefit from the reduction of round trips, although the old remote interface, unlike the new, will still send back big packets for `isc_dsql_prepare()`.

The changes work with either TCP/IP or NetBEUI. They are backward-compatible, so existing client code will not be broken. However, existing code will not enable the enhancements unless drivers are updated.

## API Changes

### *XSQLVAR*

A. dos Santos Fernandes

The identifier of the connection character set or, when the connection character set is NONE, the BLOB character set, is now passed in the `XSQLVAR::sqlscale` item of text BLOBs.

## Optimization

### *Optimization for Multiple Index Scans*

V. Horsun

[Feature request CORE-1069](#)

An optimization was done for index scanning when more than one index is to be scanned with AND conjunctions.

### *Optimize sparse bitmap operations*

V. Horsun

[Feature request CORE-1070](#)

Optimization was done for sparse bitmap operations (set, test and clear) when values are mostly consecutive.

## Configuration and Tuning

### *Increased Lock Manager Limits & Defaults*

D. Yemanov

[Feature requests CORE-958](#) and [CORE-937](#)

- the **maximum number of hash slots** is raised from 2048 to 65,536. Because the actual setting should be a prime number, the exact supported maximum is 65,521 (the biggest prime number below 65,536). The minimum is 101.
- the new **default number of hash slots** is 1009
- the default **lock table size** has been increased to 1 Mb on all platforms

## ***Page sizes of 1K and 2K Deprecated***

D. Yemanov

[Feature request CORE-969](#)

Page sizes of 1K and 2K are deprecated as inefficient.

### **Note**

The small page restriction applies to new databases only. Old ones can be attached to regardless of their page size.

## ***Enlarge Disk Allocation Chunks***

V. Horsun

[Feature request CORE-1229](#)

Allocate disk space in chunks larger than one page. [More info required.]

## ***Bypass Filesystem Caching on Superserver***

V. Horsun

[Feature requests CORE-1381](#) and [CORE-1480](#)

Firebird uses and maintains its own cache in memory for page buffers. The operating system, in turn, may re-cache Firebird's cache in its own filesystem cache. If Firebird is configured to use a cache that is large relative to the available RAM and Forced Writes is on, this cache duplication drains resources for little or no benefit.

Often, when the operating system tries to cache a big file, it moves the Firebird page cache to the swap, causing intensive, unnecessary paging. In practice, if the Firebird page cache size for Superserver is set to more than 80 per cent of the available RAM, resource problems will be extreme.

### **Note**

Filesystem caching is of some benefit on file writes, but only if Forced Writes is OFF, which is not recommended for most conditions.

Now, Superserver on both Windows and POSIX can be configured by a new configuration parameter, **MaxFileSystemCache**, to prevent or enable filesystem caching. It may provide the benefit of freeing more memory for other operations such as sorting and, where there are multiple databases, reduce the demands made on host resources.

### **Note**

For Classic, there is no escaping filesystem caching.

For details of the **MaxFileSystemCache** parameter, see [MaxFileSystemCache](#).

## Other Global Improvements

### ***Garbage Collector Rationalisation***

V. Horsun

[Feature request CORE-1071](#)

The background garbage collector process was reading all back versions of records on a page, including those created by active transactions. Since back versions of active records cannot be considered for garbage collection, it was wasteful to read them.

### ***Immediate Release of External Files***

D. Yemanov

[Feature request CORE-969](#)

The engine will now release external table files as soon as they are no longer in use by user requests.

### ***Synchronization of DSQL metadata cache objects in Classic server***

A. dos Santos Fernandes

[Feature request CORE-976](#)

No details.

### ***BLOB Improvements***

A. dos Santos Fernandes

[Feature request CORE-1169](#)

Conversion of temporary blobs to the destination blob type now occurs when materializing.

### ***Type Flag for Stored Procedures***

D. Yemanov

[Feature request CORE-779](#)

Introduced a type flag for stored procedures, adding column RDB\$PROCEDURE\_TYPE to the table RDB\$PROCEDURES. Possible values are:

- 0 or NULL -

legacy procedure (no validation checks are performed)

- 1 -

selectable procedure (one that contains a SUSPEND statement)

- 2 -

executable procedure (no SUSPEND statement, cannot be selected from)



# New Configuration Parameters and Changes

## MaxFileSystemCache

V. Horsun

Sets a threshold determining whether Firebird will allow the page cache to be duplicated to the filesystem cache or not. If this parameter is set to any (integer) value greater than zero, its effect depends on the current default size of the page cache: if the default page cache (in pages) is less than the value of MaxFileSystemCache (in pages) then filesystem caching is enabled, otherwise it is disabled.

### Note

This applies both when the page cache buffer size is set implicitly by the DefaultDBCACHEPages setting or explicitly as a database header attribute.

Thus,

- To disable filesystem caching always, set MaxFileSystemCache to zero
- To enable filesystem caching always, set MaxFileSystemCache an integer value that is sufficiently large to exceed the size of the database page cache. Remember that the effect of this value will be affected by subsequent changes to the page cache size.

### Important

The default setting for MaxFileSystemCache is 65536 pages, i.e. filesystem caching is enabled.

---

## Chapter 5

# Administrative Features

Firebird is gradually adding new features to assist in the administration of databases. Firebird 2.1 sees the introduction of a new set of system tables through which administrators can monitor transactions and statements that are active in a database. These facilities employ a new v.2.1 DDL feature, *global temporary tables* to provide snapshots.

## Monitoring Tables

Dmitry Yemanov

Firebird 2.1 introduces the ability to monitor server-side activity happening inside a particular database. The engine offers a set of so-called “virtual” tables that provides the user with a snapshot of the current activity within the given database.

The word “virtual” means that the table data is not materialised until explicitly asked for. However, the metadata of the virtual table is stable and can be retrieved from the schema.

### Note

Virtual monitoring tables exist only in ODS 11.1 (and higher) databases, so a migration via backup/restore is required in order to use this feature.

## The Concept

The key term of the monitoring feature is an *activity snapshot*. It represents the current state of the database, comprising a variety of information about the database itself, active attachments and users, transactions, prepared and running statements, and more.

A snapshot is created the first time any of the monitoring tables is being selected from in the given transaction and it is preserved until the transaction ends, in order that multiple-table queries (e.g., master-detail ones) will always return a consistent view of the data.

In other words, the monitoring tables always behave like a snapshot table stability (“consistency”) transaction, even if the host transaction has been started with a lower isolation level.

To refresh the snapshot, the current transaction should be finished and the monitoring tables should be queried in a new transaction context.

## Scope and Security

- Access to the monitoring tables is available in both DSQL and PSQL.
- Complete database monitoring is available to SYSDBA and the database owner.
- Regular users are restricted to the information about their own attachments only—other attachments are invisible to them.

## Metadata

### *MON\$DATABASE (connected database)*

- MON\$DATABASE\_NAME (database pathname or alias)
- MON\$PAGE\_SIZE (page size)
- MON\$ODS\_MAJOR (major ODS version)
- MON\$ODS\_MINOR (minor ODS version)
- MON\$OLDEST\_TRANSACTION (OIT number)
- MON\$OLDEST\_ACTIVE (OAT number)
- MON\$OLDEST\_SNAPSHOT (OST number)
- MON\$NEXT\_TRANSACTION (next transaction number)
- MON\$PAGE\_BUFFERS (number of pages allocated in the cache)
- MON\$SQL\_DIALECT (SQL dialect of the database)
- MON\$SHUTDOWN\_MODE (current shutdown mode)
  - 0: online
  - 1: multi-user shutdown
  - 2: single-user shutdown
  - 3: full shutdown
- MON\$SWEEP\_INTERVAL (sweep interval)
- MON\$READ\_ONLY (read-only flag)
- MON\$FORCED\_WRITES (sync writes flag)
- MON\$RESERVE\_SPACE (reserve space flag)
- MON\$CREATION\_DATE (creation date/time)
- MON\$PAGES (number of pages allocated on disk)
- MON\$BACKUP\_STATE (current physical backup state)
  - 0: normal
  - 1: stalled
  - 2: merge
- MON\$STAT\_ID (statistics ID)

### *MON\$ATTACHMENTS (connected attachments)*

- MON\$ATTACHMENT\_ID (attachment ID)
- MON\$SERVER\_PID (server process ID)
- MON\$STATE (attachment state)
  - 0: idle
  - 1: active
- MON\$ATTACHMENT\_NAME (connection string)
- MON\$USER (user name)
- MON\$ROLE (role name)
- MON\$REMOTE\_PROTOCOL (remote protocol name)
- MON\$REMOTE\_ADDRESS (remote address)
- MON\$REMOTE\_PID (remote client process ID)
- MON\$REMOTE\_PROCESS (remote client process pathname)
- MON\$CHARACTER\_SET\_ID (attachment character set)
- MON\$TIMESTAMP (connection date/time)
- MON\$GARBAGE\_COLLECTION (garbage collection flag)
- MON\$STAT\_ID (statistics ID)

- columns MON\$REMOTE\_PID and MON\$REMOTE\_PROCESS contains non-NULL values only if the client library is version 2.1 or higher
- column MON\$REMOTE\_PROCESS can contain a non-pathname value if an application has specified a custom process name via DPB

*MON\$TRANSACTIONS (started transactions)*

- MON\$TRANSACTION\_ID (transaction ID)
- MON\$ATTACHMENT\_ID (attachment ID)
- MON\$STATE (transaction state)
  - 0: idle
  - 1: active
- MON\$TIMESTAMP (transaction start date/time)
- MON\$TOP\_TRANSACTION (top transaction)
- MON\$OLDEST\_TRANSACTION (local OIT number)
- MON\$OLDEST\_ACTIVE (local OAT number)
- MON\$ISOLATION\_MODE (isolation mode)
  - 0: consistency
  - 1: concurrency
  - 2: read committed record version
  - 3: read committed no record version
- MON\$LOCK\_TIMEOUT (lock timeout)
  - 1: infinite wait
  - 0: no wait
  - N: timeout N
- MON\$READ\_ONLY (read-only flag)
- MON\$AUTO\_COMMIT (auto-commit flag)
- MON\$AUTO\_UNDO (auto-undo flag)
- MON\$STAT\_ID (statistics ID)

*MON\$STATEMENTS (prepared statements)*

- MON\$STATEMENT\_ID (statement ID)
  - MON\$ATTACHMENT\_ID (attachment ID)
  - MON\$TRANSACTION\_ID (transaction ID)
  - MON\$STATE (statement state)
    - 0: idle
    - 1: active
  - MON\$TIMESTAMP (statement start date/time)
  - MON\$SQL\_TEXT (statement text, if appropriate)
  - MON\$STAT\_ID (statistics ID)
- 
- column MON\$SQL\_TEXT contains NULL for GDML statements
  - columns MON\$TRANSACTION\_ID and MON\$TIMESTAMP contain valid values for active statements only

*MON\$CALL\_STACK (call stack of active PSQL requests)*

- MON\$CALL\_ID (request ID)
  - MON\$STATEMENT\_ID (top-level DSQL statement ID)
  - MON\$CALLER\_ID (caller request ID)
  - MON\$OBJECT\_NAME (PSQL object name)
  - MON\$OBJECT\_TYPE (PSQL object type)
  - MON\$TIMESTAMP (request start date/time)
  - MON\$SOURCE\_LINE (SQL source line number)
  - MON\$SOURCE\_COLUMN (SQL source column number)
  - MON\$STAT\_ID (statistics ID)
- column MON\$STATEMENT\_ID groups call stacks by the top-level DSQL statement that initiated the call chain. This ID represents an active statement record in the table MON\$STATEMENTS.
  - columns MON\$SOURCE\_LINE and MON\$SOURCE\_COLUMN contain line/column information related to the PSQL statement currently being executed

#### Note

Textual descriptions of all “state” and “mode” values can be found in the system table RDB\$TYPES.

## Usage

Creation of a snapshot is usually quite a fast operation, but some delay could be expected under high load (especially in the Classic Server).

A valid database connection is required in order to retrieve the monitoring data. The monitoring tables return information about the attached database only. If multiple databases are being accessed on the server, each of them has to be connected to and monitored separately.

The system variables CURRENT\_CONNECTION and CURRENT\_TRANSACTION could be used to select data about the caller's current connection and transaction respectively. These variables correspond to the ID columns of the appropriate monitoring tables.

## Examples

1. Retrieve IDs of all CS processes loading CPU at the moment

```
SELECT MON$SERVER_PID
FROM MON$ATTACHMENTS
WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
AND MON$STATE = 1
```

2. Retrieve information about client applications

```
SELECT MON$USER, MON$REMOTE_ADDRESS,
MON$REMOTE_PID,
MON$TIMESTAMP
FROM MON$ATTACHMENTS
WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

### 3. Get isolation level of the current transaction

```
SELECT MON$ISOLATION_MODE
FROM MON$TRANSACTIONS
WHERE MON$TRANSACTION_ID = CURRENT_TRANSACTION
```

### 4. Get statements that are currently active

```
SELECT ATT.MON$USER,
       ATT.MON$REMOTE_ADDRESS,
       STMT.MON$SQL_TEXT,
       STMT.MON$TIMESTAMP
FROM MON$ATTACHMENTS ATT
JOIN MON$STATEMENTS STMT
  ON ATT.MON$ATTACHMENT_ID = STMT.MON$ATTACHMENT_ID
WHERE ATT.MON$ATTACHMENT_ID <> CURRENT_CONNECTION
AND STMT.MON$STATE = 1
```

### 5. Retrieve call stacks for all connections

```
WITH RECURSIVE HEAD AS
(
  SELECT CALL.MON$STATEMENT_ID,
         CALL.MON$CALL_ID,
         CALL.MON$OBJECT_NAME,
         CALL.MON$OBJECT_TYPE
  FROM MON$CALL_STACK CALL
  WHERE CALL.MON$CALLER_ID IS NULL
  UNION ALL
  SELECT CALL.MON$STATEMENT_ID,
         CALL.MON$CALL_ID,
         CALL.MON$OBJECT_NAME,
         CALL.MON$OBJECT_TYPE
  FROM MON$CALL_STACK CALL
  JOIN HEAD
    ON CALL.MON$CALLER_ID = HEAD.MON$CALL_ID
)
SELECT MON$ATTACHMENT_ID,
       MON$OBJECT_NAME,
       MON$OBJECT_TYPE
FROM HEAD
JOIN MON$STATEMENTS STMT
  ON STMT.MON$STATEMENT_ID = HEAD.MON$STATEMENT_ID
WHERE STMT.MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

## Cancel a Running Query

Runaway and long-running queries can now be cancelled *from a separate connection*.

There is no API function call directed at this feature. It will be up to the SysAdmin (SYSDBA or Owner) to make use of the data available in the monitoring tables and devise an appropriate mechanism for reining in the rogue statements.

### Example

As a very rough example, the following statement will kill all statements currently running in the database, other than any that belong to the *separate* connection that the SysAdmin is using himself:

```
delete from mon$statements
  where mon$attachment_id <> current_connection
```

## More Context Information

More context information about the server and database ('SYSTEM') is available via SELECT calls to the RDB \$GET\_CONTEXT function, including the engine version.

### Example

```
SELECT RDB$GET_CONTEXT('SYSTEM', 'ENGINE_VERSION')
FROM RDB$DATABASE
```

For detailed information about using these context calls, refer to the v.2.0.1 release notes.

---

## Chapter 6

# SQL Language Enhancements

In this chapter are the additions and improvements that have been added to the SQL language in the v.2.1 development phase.

## Data Definition Language (DDL)

Several new features and improvements have been added to the DDL language set.

### Database Triggers

Adriano dos Santos Fernandes

A *database trigger* is a PSQL module that is executed when a connection or transaction event occurs. The events and the timings of their triggers are as follows.-

#### *CONNECT*

- Database connection is established
- A transaction is started
- Triggers are fired; uncaught exceptions roll back the transaction, disconnect the attachment and are returned to the client
- The transaction is committed

#### *DISCONNECT*

- A transaction is started
- Triggers are fired; uncaught exceptions roll back the transaction, disconnect the attachment and are swallowed
- The transaction is committed
- The attachment is disconnected

#### *TRANSACTION START*

Triggers are fired in the newly-created user transaction; uncaught exceptions are returned to the client and the transaction is rolled back.

#### *TRANSACTION COMMIT*

Triggers are fired in the committing transaction; uncaught exceptions roll back the trigger's savepoint, the commit command is aborted and the exception is returned to the client.



**Note**

For two-phase transactions, the triggers are fired in the “prepare”, not in the commit.

**TRANSACTION ROLLBACK**

Triggers are fired during the roll-back of the transaction. Changes done will be rolled back with the transaction. Exceptions are swallowed

**Syntax**

```
<database-trigger> ::=
  {CREATE | RECREATE | CREATE OR ALTER}
  TRIGGER <name>
  [ACTIVE | INACTIVE]
  ON <event>
  [POSITION <n>]
AS
  BEGIN
    ...
  END

<event> ::=
  CONNECT
  | DISCONNECT
  | TRANSACTION START
  | TRANSACTION COMMIT
  | TRANSACTION ROLLBACK
```

**Rules and Restrictions**

1. Database triggers type cannot be changed.
2. Permission to create, recreate, create or alter, or drop database triggers is restricted to the database owner and SYSDBA.

**Utilities Support for Database Triggers**

A new parameter was added to gbak, nbackup and isql to suppress database triggers from running. It is available only to the database owner and SYSDBA:

```
gbak -no_dbtriggers
isql -nodbtriggers
nbackup -T
```

**Global Temporary Tables**

Vlad Horsun

Global temporary tables (GTTs) are tables that are stored in the system catalogue with permanent metadata, but with temporary data. Data from different connections (or transactions, depending on the scope) are isolated from each other, but the metadata of the GTT are shared among all connections and transactions.

There are two kinds of GTT:

- with data that persists for the lifetime of connection in which the specified GTT was referenced; and
- with data that persists only for the lifetime of the referencing transaction.

## Syntax and Rules for GTTs

```
CREATE GLOBAL TEMPORARY TABLE
...
[ON COMMIT <DELETE | PRESERVE> ROWS]
```

Creates the metadata for the temporary table in the system catalogue.

The clause ON COMMIT sets the kind of temporary table:

### *ON COMMIT PRESERVE ROWS*

Data left in the given table after the end of the transaction remain in database until the connection ends.

### *ON COMMIT DELETE ROWS*

Data in the given table are deleted from the database immediately after the end of the transaction. ON COMMIT DELETE ROWS is used by default if the optional clause ON COMMIT is not specified.

### *CREATE GLOBAL TEMPORARY TABLE*

is a regular DDL statement that is processed by the engine the same way as a CREATE TABLE statement is processed. Accordingly, it not possible to create or drop a GTT within a stored procedure or trigger.

## Relation Type

GTT definitions are distinguished in the system catalogue from one another and from permanent tables by the value of `RDB$RELATIONS.RDB$RELATION_TYPE`:

- A GTT with ON COMMIT PRESERVE ROWS option has `RDB$RELATION_TYPE = 4`

A GTT with ON COMMIT DELETE ROWS option has `RDB$RELATION_TYPE = 5`.

### Note

For the full list of values, see `RDB$TYPES`.

## Structural Feature Support

The same structural features that you can apply to regular tables (indexes, triggers, field-level and table level constraints) are also available to a GTT, with certain restrictions on how GTTs and regular tables can interrelate.-

- a. references between persistent and temporary tables are forbidden
- b. A GTT with ON COMMIT PRESERVE ROWS cannot have a reference on a GTT with ON COMMIT DELETE ROWS
- c. A domain constraint cannot have a reference to any GTT.

## Implementation Notes

An instance of a GTT—a set of data rows created by and visible within the given connection or transaction—is created when the GTT is referenced for the first time, usually at statement prepare time. Each instance has its own private set of pages on which data and indexes are stored. The data rows and indexes have the same physical storage layout as permanent tables.

When the connection or transaction ends, all pages of a GTT instance are released immediately. It is similar to what happens when a DROP TABLE is performed, except that the metadata definition is retained, of course. This is much quicker than the traditional row-by-row delete + garbage collection of deleted record versions.

### Note

This method of deletion does not cause DELETE triggers to fire, so do not be tempted to define Before or After Delete triggers on the false assumption that you can incorporate some kind of “last rites” that will be execute just as your temporary data breathes its last!

The data and index pages of all GTT instances are placed in separate temporary files. Each connection has its own temporary file created the first time the connection references some GTT.

### Note

These temporary files are always opened with Forced Writes = OFF, regardless of the database setting for Forced Writes.

No limit is placed on the number of GTT instances that can coexist. If you have N transactions active simultaneously and each transaction has referenced some GTT then you will have N instances of the GTT.

## BLOB Subtype 1 Compatibility with VarChar

A. dos Santos Fernandes

At various levels of evaluation, the engine now treats text BLOBs that are within the 32,765- byte string size limit as though they were varchars. Operations that now allow text BLOBs to behave like strings are assignments, conversions and concatenations, as well as the functions CAST, LOWER, UPPER, TRIM and SUBSTRING.

## Views Enhancements

D. Yemanov

A couple of enhancements were made to view definitions in v.2.1.-

### Use Column Aliases in CREATE VIEW

Feature request [CORE-831](#)

Column aliases can now be processed as column names in the view definition.

### Example

```
CREATE VIEW V_TEST AS
  SELECT ID,
         COL1 AS CODE,
         COL2 AS NAME
  FROM TAB;
```

### ***CURRENT OF Now Allowed for Views***

[Feature request CORE-1213](#)

. [Need some doc and an example]

### ***SQL2003 compliance for CREATE TRIGGER***

A. dos Santos Fernandes

[Feature request CORE-711](#)

Syntax for CREATE TRIGGER now complies with SQL2003. Pattern? Examples?

### ***SQL2003 Compliant Alternative for Computed Fields***

D. Yemanov

[Feature request CORE-1386](#)

SQL-compliant alternative syntax **GENERATED ALWAYS AS** was implemented for defining a computed field in CREATE/ALTER TABLE.

#### **Syntax Pattern**

```
<column name> [<type>] GENERATED ALWAYS AS ( <expr> )
```

It is fully equivalent semantically with the legacy form:

```
<column name> [<type>] COMPUTED [BY] ( <expr> )
```

#### **Example**

```
CREATE TABLE T (PK INT, EXPR GENERATED ALWAYS AS (PK + 1))
```

## **Data Manipulation Language (DML)**

Several new features and improvements have been added to the DML language set.

## Common Table Expressions

Vlad Horsun

Based on work by Paul Ruizendaal for Fyracle project

A *common table expression* (CTE) is like a view that is defined locally within a main query. The engine treats a CTE like a derived table and no intermediate materialisation of the data is performed.

### Benefits of CTEs

Using CTEs allows you to specify dynamic queries that are recursive:

- The engine begins execution from a non-recursive member.
- For each row evaluated, it starts executing each recursive member one-by-one, using the current values from the outer row as parameters.
- If the currently executing instance of a recursive member produces no rows, execution loops back one level and gets the next row from the outer result set.

The memory and CPU overhead of a recursive CTE is much less than that of an equivalent recursive stored procedure.

### Recursion Limit

Currently the recursion depth is limited to a hard-coded value of 1024.

### Syntax and Rules for CTEs

```
select :
  select_expr for_update_clause lock_clause
select_expr :
  with_clause select_expr_body order_clause rows_clause
  | select_expr_body order_clause rows_clause
with_clause :
  WITH RECURSIVE with_list | WITH with_list
with_list :
  with_item | with_item ',' with_list
with_item :
  symbol_table_alias_name derived_column_list
  AS '(' select_expr ')'
select_expr_body :
  query_term
  | select_expr_body UNION distinct_noise query_term
  | select_expr_body UNION ALL query_term
```

A less formal representation:

```
WITH [RECURSIVE]
  CTE_A [(a1, a2, ...)]
  AS ( SELECT ... ),

  CTE_B [(b1, b2, ...)]
  AS ( SELECT ... ),
...
SELECT ...
  FROM CTE_A, CTE_B, TAB1, TAB2 ...
  WHERE ...
```

### **Rules for Non-Recursive CTEs**

- Multiple table expressions can be defined in one query
- Any clause legal in a SELECT specification is legal in table expressions
- Table expressions can reference one another
- References between expressions should not have loops
- Table expressions can be used within any part of the main query or another table expression
- The same table expression can be used more than once in the main query
- Table expressions (as subqueries) can be used in INSERT, UPDATE and DELETE statements
- Table expressions are legal in PSQL code
- WITH statements can not be nested

### **Example of a non-recursive CTE**

```
WITH
  DEPT_YEAR_BUDGET AS (
    SELECT FISCAL_YEAR, DEPT_NO,
           SUM(PROJECTED_BUDGET) AS BUDGET
    FROM PROJ_DEPT_BUDGET
    GROUP BY FISCAL_YEAR, DEPT_NO
  )
SELECT D.DEPT_NO, D.DEPARTMENT,
       B_1993.BUDGET AS B_1993, B_1994.BUDGET AS B_1994,
       B_1995.BUDGET AS B_1995, B_1996.BUDGET AS B_1996
FROM DEPARTMENT D
  LEFT JOIN DEPT_YEAR_BUDGET B_1993
    ON D.DEPT_NO = B_1993.DEPT_NO
   AND B_1993.FISCAL_YEAR = 1993
  LEFT JOIN DEPT_YEAR_BUDGET B_1994
    ON D.DEPT_NO = B_1994.DEPT_NO
   AND B_1994.FISCAL_YEAR = 1994
  LEFT JOIN DEPT_YEAR_BUDGET B_1995
    ON D.DEPT_NO = B_1995.DEPT_NO
   AND B_1995.FISCAL_YEAR = 1995
  LEFT JOIN DEPT_YEAR_BUDGET B_1996
    ON D.DEPT_NO = B_1996.DEPT_NO
   AND B_1996.FISCAL_YEAR = 1996
```

```
WHERE EXISTS (  
  SELECT * FROM PROJ_DEPT_BUDGET B  
  WHERE D.DEPT_NO = B.DEPT_NO)
```

### **Rules for Recursive CTEs**

- A recursive CTE is self-referencing (has a reference to itself)
- A recursive CTE is a UNION of recursive and non-recursive members:
  - At least one non-recursive member (anchor) must be present
  - Non-recursive members are placed first in the UNION
  - Recursive members are separated from anchor members and from one another with a UNION ALL clause [Ed. note: author, please clarify]
- References between CTEs should not have loops
- Aggregates (DISTINCT, GROUP BY, HAVING) and aggregate functions (SUM, COUNT, MAX etc) are not allowed in recursive members
- A recursive member can have only one reference to itself and only in a FROM clause
- A recursive reference cannot participate in an outer join

### **Example of a recursive CTE**

```
WITH RECURSIVE  
  DEPT_YEAR_BUDGET AS  
  (  
    SELECT FISCAL_YEAR, DEPT_NO,  
           SUM(PROJECTED_BUDGET) AS BUDGET  
    FROM PROJ_DEPT_BUDGET  
    GROUP BY FISCAL_YEAR, DEPT_NO  
  ),  
  
  DEPT_TREE AS  
  (  
    SELECT DEPT_NO, HEAD_DEPT, DEPARTMENT,  
           CAST(' ' AS VARCHAR(255)) AS INDENT  
    FROM DEPARTMENT  
    WHERE HEAD_DEPT IS NULL  
  
    UNION ALL  
  
    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
           H.INDENT || ' ' AS INDENT  
    FROM DEPARTMENT D  
    JOIN DEPT_TREE H  
      ON D.HEAD_DEPT = H.DEPT_NO  
  )  
  
  SELECT D.DEPT_NO,  
         D.INDENT || D.DEPARTMENT AS DEPARTMENT,  
         B_1993.BUDGET AS B_1993,  
         B_1994.BUDGET AS B_1994,
```

```
B_1995.BUDGET AS B_1995,  
B_1996.BUDGET AS B_1996  
  
FROM DEPT_TREE D  
  LEFT JOIN DEPT_YEAR_BUDGET B_1993  
    ON D.DEPT_NO = B_1993.DEPT_NO  
    AND B_1993.FISCAL_YEAR = 1993  
  
  LEFT JOIN DEPT_YEAR_BUDGET B_1994  
    ON D.DEPT_NO = B_1994.DEPT_NO  
    AND B_1994.FISCAL_YEAR = 1994  
  
  LEFT JOIN DEPT_YEAR_BUDGET B_1995  
    ON D.DEPT_NO = B_1995.DEPT_NO  
    AND B_1995.FISCAL_YEAR = 1995  
  
  LEFT JOIN DEPT_YEAR_BUDGET B_1996  
    ON D.DEPT_NO = B_1996.DEPT_NO  
    AND B_1996.FISCAL_YEAR = 1996
```

## The LIST Function

Oleg Loa  
Dmitry Yemanov

This function returns a string result with the concatenated non-NULL values from a group. It returns NULL if there are no non-NULL values.

### Format

```
<list function> ::=  
  LIST '(' [ {ALL | DISTINCT} ] <value expression> [ ',' <delimiter value>  
    ] ')'  
  
<delimiter value> ::=  
  { <string literal> | <parameter> | <variable> }
```

### Syntax Rules

1. If neither ALL nor DISTINCT is specified, ALL is implied.
2. If <delimiter value> is omitted, a comma is used to separate the concatenated values.

### Other Notes

1. Numeric and date/time values are implicitly converted to strings during evaluation.
2. The result value is of type BLOB with SUB\_TYPE TEXT for all cases except list of BLOB with different subtype.
3. Ordering of values within a group is implementation-defined.

### Examples



```
/* A */
SELECT LIST(ID, ':')
FROM MY_TABLE

/* B */
SELECT TAG_TYPE, LIST(TAG_VALUE)
FROM TAGS
GROUP BY TAG_TYPE
```

## The RETURNING Clause

Dmitry Yemanov  
Adriano dos Santos Fernandes

The purpose of this SQL enhancement is to enable the column values stored into a table as a result of the INSERT, UPDATE OR INSERT, UPDATE and DELETE statements to be returned to the client.

The most likely usage is for retrieving the value generated for a primary key inside a BEFORE-trigger. The RETURNING clause is optional and is available in both DSQL and PSQL, although the rules differ slightly.

In DSQL, the execution of the operation itself and the return of the set occur in a single protocol round trip.

Because the RETURNING clause is designed to return a singleton set in response to completing an operation on a single record, it is not valid to specify the clause in a statement that inserts, updates or deletes multiple records.

### Note

In DSQL, the statement always returns the set, even if the operation has no effect on any record. Hence, at this stage of implementation, the potential exists to return an “empty” set. (This may be changed in future.)

## Syntax Patterns

```
INSERT INTO ... VALUES (...)
    [RETURNING <column_list> [INTO <variable_list>]]

INSERT INTO ... SELECT ...
    [RETURNING <column_list> [INTO <variable_list>]]

UPDATE OR INSERT INTO ... VALUES (...) ...
    [RETURNING <column_list> [INTO <variable_list>]]

UPDATE ... [RETURNING <column_list> [INTO <variable_list>]]

DELETE FROM ...
    [RETURNING <column_list> [INTO <variable_list>]]
```

## Rules for Using a RETURNING Clause

1. The INTO part (i.e. the variable list) is allowed in PSQL only, for assigning the output set to local variables. It is rejected in DSQL.

2. The presence of the RETURNING clause causes an INSERT statement to be described by the API as `isc_info_sql_stmt_exec_procedure` rather than `isc_info_sql_stmt_insert`. Existing connectivity drivers should already be capable of supporting this feature without special alterations.
3. The RETURNING clause ignores any explicit record change (update or delete) that occurs as a result of the execution of an AFTER trigger.
4. OLD and NEW context variables can be used in the RETURNING clause of UPDATE and INSERT OR UPDATE statements.
5. In UPDATE and INSERT OR UPDATE statements, field references that are unqualified or qualified by table name or relation alias are resolved to the value of the corresponding NEW context variable.

### Examples

1.

```
INSERT INTO T1 (F1, F2)
VALUES (:F1, :F2)
RETURNING F1, F2 INTO :V1, :V2;
```

2.

```
INSERT INTO T2 (F1, F2)
VALUES (1, 2)
RETURNING ID INTO :PK;
```

3.

```
DELETE FROM T1
WHERE F1 = 1
RETURNING F2;
```

4.

```
UPDATE T1
SET F2 = F2 * 10
RETURNING OLD.F2, NEW.F2;
```

## UPDATE OR INSERT Statement

Adriano dos Santos Fernandes

This syntax has been introduced to enable a record to be either updated or inserted, according to whether or not it already exists (checked with IS NOT DISTINCT). The statement is available in both DSQL and PSQL.

### Syntax Pattern

```
UPDATE OR INSERT INTO <table or view> [( <column_list> )]
VALUES ( <value_list> )
[ MATCHING <column_list> ]
[ RETURNING <column_list> [ INTO <variable_list> ] ]
```

## Examples

1.

```
UPDATE OR INSERT INTO T1 (F1, F2)
VALUES (:F1, :F2);
```

2.

```
UPDATE OR INSERT INTO EMPLOYEE (ID, NAME)
VALUES (:ID, :NAME)
RETURNING ID;
```

3.

```
UPDATE OR INSERT INTO T1 (F1, F2)
VALUES (:F1, :F2)
MATCHING (F1);
```

4.

```
UPDATE OR INSERT INTO EMPLOYEE (ID, NAME)
VALUES (:ID, :NAME)
RETURNING OLD.NAME;
```

## Usage Notes

1. When MATCHING is omitted, the existence of a primary key is required.
2. INSERT and UPDATE permissions are needed on <table or view>.
3. If the RETURNING clause is present, then the statement is described as `isc_info_sql_stmt_exec_procedure` by the API; otherwise, it is described as `isc_info_sql_stmt_insert`.

### Note

A “multiple rows in singleton select” error will be raised if the RETURNING clause is present and more than one record matches the search condition.

## New JOIN Types

Adriano dos Santos Fernandes

Two new JOIN types are introduced: the NAMED COLUMNS join and its close relative, the NATURAL join.

## Syntax and Rules

```
<named columns join> ::=  
  <table reference> <join type> JOIN <table reference>  
    USING ( <column list> )  
  
<natural join> ::=  
  <table reference> NATURAL <join type> JOIN <table primary>
```

### **Named columns join**

1. All columns specified in <column list> should exist in the tables at both sides.
2. An equi-join (<left table>.<column> = <right table>.<column>) is automatically created for all columns (ANDed).
3. The USING columns can be accessed without qualifiers—in this case, the result is equivalent to COALESCE(<left table>.<column>, <right table>.<column>).
4. In “SELECT \*”, USING columns are expanded once, using the above rule.

### **Natural join**

1. A “named columns join” is automatically created with all columns common to the left and right tables.
2. If there is no common column, a CROSS JOIN is created.

### **Examples**

```
/* 1 */  
select * from employee  
  join department  
    using (dept_no);  
  
/* 2 */  
select * from employee_project  
  natural join employee  
  natural join project;
```

## **INSERT with defaults**

D. Yemanov

### [Feature request](#)

It is now possible to INSERT without supplying values, if Before Insert triggers and/or declared defaults are available for every column and none is dependent on the presence of any supplied 'NEW' value.

### **Example**

```
INSERT INTO <table>  
  DEFAULT VALUES
```

[RETURNING <values>]

## ***Make RDB\$DB\_KEY in outer joins return NULL when appropriate***

A. dos Santos Fernandes

[Feature request CORE-979](#)

[ Details needed. ]

## ***Data Type of an Aggregation Result***

Arno Brinkman

When aggregations, CASE evaluations and UNIONs for output columns are performed over a mix of comparable data types, the engine has to choose one data type for the result. The developer often has to prepare a variable or buffer for such results and is mystified when a request returns a data type exception. The rules followed by the engine in determining the data type for an output column under these conditions are explained here.

1. Let DTS be the set of data types over which we must determine the final result data type.
2. All of the data types in DTS shall be comparable.
3. In the case that
  - a. any of the data types in DTS is character string
    - i. If all data types in DTS are fixed-length character strings, then the result is also a fixed-length character string; otherwise the result is a variable-length character string.

The resulting string length, in characters, is equal to the maximum of the lengths, in characters, of the data types in DTS.
    - ii. The character set and collation used are taken from the data type of the first character string in DTS.
  - b. all of the data types in DTS are exact numeric

the result data type is exact numeric with scale equal to the maximum of the scales of the data types in DTS and precision equal to the maximum precision of all data types in DTS.
  - c. any data type in DTS is approximate numeric

each data type in DTS must be numeric, otherwise an error is thrown.
  - d. any data type in DTS is a date/time data type

every data type in DTS must be a date/time type having the *same date/time type*, otherwise an error is thrown.
  - e. any data type in DTS is BLOB

each data type in DTS must be BLOB and all with the same sub-type.

## Built-in Functions

Some existing built-in functions have been enhanced, while a large number of new ones has been added.

### New Built-in Functions

Adriano dos Santos Fernandes  
Oleg Loa  
Alexey Karyakin

A number of built-in functions has been implemented in V.2.1 to replace common UDFs with the same names. The built-in functions will not be used if the UDF of the same name is declared in the database.

#### Note

The choice between UDF and built-in function is decided when compiling the statement. If the statement is compiled in a PSQL module whilst the UDF is available in the database, then the module will continue to require the UDF declaration to be present until it is next recompiled.

The new built-in function *DECODE()* does not have an equivalent UDF in the libraries that are distributed with Firebird.

The functions are detailed in [Appendix A](#).

## Enhancements to Functions

A. dos Santos Fernandes

### *EXTRACT(WEEK FROM DATE)*

Feature request [CORE-663](#)

The *EXTRACT()* function is extended to support the ISO-8601 ordinal week numbers. For example:

```
EXTRACT (WEEK FROM date '30.09.2007')
```

returns 39

### *Specify the Scale for TRUNC()*

Feature request [CORE-1340](#)

In Beta 1 the implementation of the *TRUNC()* function supported only one argument, the value to be truncated. From Beta 2, an optional second argument can be supplied to specify the scale of the truncation. For example:

```
select
  trunc(987.65, 1),
  trunc(987.65, -1)
from rdb$database;
```

returns 987.60, 980.00

For other examples of using TRUNC() with and without the optional scale argument, refer to the alphabetical listing of functions in Appendix A.

#### *Milliseconds Handling for EXTRACT(), DATEADD() and DATEDIFF()*

Feature request [CORE-1387](#)

From v.2.1 Beta 2, EXTRACT(), DATEADD() and DATEDIFF() can operate with milliseconds (represented as an integer number). For example:

```
EXTRACT ( MILLISECOND FROM timestamp '01.01.2000 01:00:00.1234' )
```

returns 123

```
DATEADD ( MILLISECOND, 100, timestamp '01.01.2000 01:00:00.0000' )
DATEDIFF ( MILLISECOND, timestamp '01.01.2000 02:00:00.0000', timestamp '01.01.2000 01:00:00.
```

For more explanatory examples of using DATEADD() and DATEDIFF(), refer to the alphabetical listing of functions in Appendix A.

#### *DATEADD and DATEDIFF Expanded Form Semantics*

Improvement request [CORE-1490](#)

The semantics used in the choice of keywords for the expanded form of the DATEDIFF() function syntax will be changed in the next beta or release candidate. In the pattern:

```
DATEDIFF( <timestamp_part> FROM <date_time> FOR <date_time> )
```

the keywords FROM and FOR will be changed to:

```
DATEDIFF( <timestamp_part> FROM <date_time> TO <date_time> )
```

The contracted form is not affected.

Similarly, the FOR keyword in the expanded form of DATEADD() will be changed to TO.

## **Changes in DSQL Parsing**

### **Sorting on BLOB and ARRAY Columns is Now Disallowed**

Dmitry Yemanov

Previous versions of the engine allowed BLOBs and arrays to be sorted on. While both GROUP BY and ORDER BY on BLOBs could be prepared successfully, a run-time error was thrown for GROUP BY, due to inability to convert a blob to a sortable type. ORDER BY "worked" (after a fashion) in all versions, with the sort operating on the BLOB\_ID.

Such a level of tolerance produces wrong or undesirable results. Due to the sorter restrictions, re-implementing it in any way that makes sense in practice is not possible. Hence, from v.2.1 onwards, any sorting operation that involves a BLOB or ARRAY type column is rejected at the PREPARE stage.

## Procedural Language Extensions (PSQL)

The highlight of PSQL changes in v.2.1 is the ability to use domains when declaring variables and arguments in procedures and triggers. A handful of other improvements have been added to the PSQL extensions.

### *Domains in PSQL*

Adriano dos Santos Fernandes

It is now possible to use a domain when declaring the data types of arguments and variables in PSQL modules. Depending on your requirements, you can declare the argument or variable using

- the domain identifier alone, in lieu of the native data type identifier, to have the variable inherit all of the attributes of the domain; or
- the data type of the domain, without inheriting CHECK constraints and the DEFAULT value (if declared in the domain), by including the TYPE OF keyword in the declaration (see the syntax below)

#### Syntax

```
data_type ::=
    <builtin_data_type>
    | <domain_name>
    | TYPE OF <domain_name>
```

#### Examples

```
CREATE DOMAIN DOM AS INTEGER;

CREATE PROCEDURE SP (
    I1 TYPE OF DOM,
    I2 DOM)
RETURNS (
    O1 TYPE OF DOM,
    O2 DOM)
AS
    DECLARE VARIABLE V1 TYPE OF DOM;
    DECLARE VARIABLE V2 DOM;

BEGIN
    ...
END
```

#### Note

A new field RDB\$VALID\_BLR was added in RDB\$RELATIONS and RDB\$TRIGGERS to indicate whether the procedure/trigger is valid after an ALTER DOMAIN operation. The value of RDB\$VALID\_BLR is shown in the ISQL commands SHOW PROCEDURE or SHOW TRIGGER.



## ***COLLATE in Stored Procedures and Parameters***

A. dos Santos Fernandes

Collations can now be applied to PSQL variables, including stored procedure parameters.

## **Optimization**

Optimization improvements in v.2.1 include:

*Economising on Indexed Reads for MIN() and MAX()*

Indexed MIN/MAX aggregates would produce three indexed reads instead of the expected single read. So, with an ASC index on the non-nullable COL, the query

**SELECT MIN(COL) FROM TAB**

should be completely equivalent, to

**SELECT FIRST 1 COL FROM TAB  
ORDER BY 1 ASC**

with both performing a single record read. However, formerly, the first query required three indexed reads while the second one required just the expected single read. Now, they both resolve to a single read.

The same optimization applies to the MAX() function when mapped to a DESC index.

---

## Chapter 7

# International Language Support (INTL)

Adriano dos Santos Fernandes

In this chapter are the additions and improvements that have been added to the v.2.0.x international language support system (INTL) in the v.2.1 development phase. Most notably, it is no longer necessary to use the script `misc/intl.sql` for implementing a collation, since the DDL command `CREATE COLLATION` has been introduced for this task.

Further capabilities have been implemented for

1. using ICU charsets through `fbintl`
2. UNICODE collation (`charset_UNICODE`) being available for all `fbintl` charsets
3. using collation attributes
4. `CREATE/DROP COLLATION` statements
5. `SHOW COLLATION` and collation extraction in ISQL
6. Verifying that text blobs are well-formed
7. Transliterating text blobs automatically

## The CREATE COLLATION Command

### Syntax for CREATE COLLATION

```
CREATE COLLATION <name>
  FOR <charset>
  [ FROM <base> | FROM EXTERNAL ( '<name>' ) ]
  [ NO PAD | PAD SPACE ]
  [ CASE SENSITIVE | CASE INSENSITIVE ]
  [ ACCENT SENSITIVE | ACCENT INSENSITIVE ]
  [ '<specific-attributes>' ]
```

#### Note

Specific attributes should be separated by semicolon and are case sensitive.

### Examples

```

/* 1 */
CREATE COLLATION UNICODE_ENUS_CI
    FOR UTF8
    FROM UNICODE
    CASE INSENSITIVE
    'LOCALE=en_US';
/* 2 */
CREATE COLLATION NEW_COLLATION
    FOR WIN1252
    PAD SPACE;

/* NEW_COLLATION should be declared in .conf file
   in $root/intl directory */

```

## ICU Character Sets

All non-wide and ASCII-based character sets present in ICU can be used by Firebird. To reduce the size of the distribution kit, we customize ICU to include only essential character sets and any for which there was a specific feature request.

If the character set you need is not included, you can replace the ICU libraries with another complete module, found at our site or already installed in your operating system.

### *Registering an ICU Character Set Module*

To use an alternative module, you first need to register it in `intl/fbintl.conf`, as follows.-

```

<charset          NAME>
    intl_module    fbintl
    collation      NAME [REAL-NAME]
</charset>

```

To register the module in databases, run the procedure `sp_register_character_set`, the source for which can be found in `misc/intl.sql`.

#### **Note**

You need to know how many bytes a single character can occupy in the encoding.

#### **Example**

```

<charset          GB>
    intl_module    fbintl
    collation      GB GB18030
</charset>

execute procedure sp_register_character_set ('GB', 4);

```

## The UNICODE Collations

The UNICODE collations (case sensitive and case insensitive) can be applied to any character set that is present in `fbintl.conf`. They are already registered in `fbintl.conf`, but you need to register them in the databases, with the desired associations and attributes.

### Naming Conventions

The naming convention you should use is `charset_collation`. For example,

```
create collation win1252_unicode
  for win1252;

create collation win1252_unicode_ci
  for win1252
  from win1252_unicode
  case insensitive;
```

#### Note

The character set name should be as in `fbintl.conf` (i.e. `ISO8859_1` instead of `ISO88591`, for example).

## Specific Attributes for Collations

#### Note

Some attributes may not work with some collations, even though they do not report an error.

### *DISABLE-COMPRESSIONS*

Disable compressions (aka contractions) changing the order of a group of characters.

Valid for collations of narrow character sets.

Format: `DISABLE-COMPRESSIONS={0 | 1}`

#### Example

`DISABLE-COMPRESSIONS=1`

### *DISABLE-EXPANSIONS*

Disable expansions changing the order of a character to sort as a group of characters.

Valid for collations of narrow character sets.

Format: `DISABLE-EXPANSIONS={0 | 1}`

#### Example

`DISABLE-EXPANSIONS=1`

### *ICU-VERSION*

Specify what version of ICU library will be used. Valid values are the ones defined in the config file (intl/fbintl.conf) in entry intl\_module/icu\_versions.

Valid for UNICODE and UNICODE\_CI.

Format: ICU-VERSION={default | major.minor}

#### **Example**

ICU-VERSION=3.0

### *LOCALE*

Specify the collation locale.

Valid for UNICODE and UNICODE\_CI. Requires complete version of ICU libraries.

Format: LOCALE=xx\_XX

#### **Example**

LOCALE=en\_US

### *MULTI-LEVEL*

Uses more than one level for ordering purposes.

Valid for collations of narrow character sets.

Format: MULTI-LEVEL={0 | 1}

#### **Example**

MULTI-LEVEL=1

### *SPECIALS-FIRST*

Order special characters (spaces, symbols, etc) before alphanumeric characters.

Valid for collations of narrow character sets.

Format: SPECIALS-FIRST={0 | 1}

#### **Example**

SPECIALS-FIRST=1

## Collation Changes

### *Spanish*

ES\_ES (as well as the new ES\_ES\_CI\_AI) collation automatically uses attributes DISABLE-COMPRESSIONS=1;SPECIALS-FIRST=1.

**Note**

The attributes are stored at database creation time, so the changes do not apply to databases with ODS < 11.1.

The ES\_ES\_CI\_AI collation was standardised to current usage.

*UTF-8*

Case-insensitive collation for UTF-8. See [feature request CORE-972](#)

## Supported Character Sets

See [Appendix B](#) at the end of these notes, for a listing the supported character sets.

### *Character Sets Added*

The following character sets and/or collations have been added to the main manifest:

*Case-insensitive collation for French*

[Feature request CORE-1366](#)

French case-insensitive collation FR\_FR\_CI\_AI, contributed by A. dos Santos Fernandes

-----

*CP943C for Japanese*

[Feature request CORE-1324](#)

Japanese character set CP943C, contributed by A. dos Santos Fernandes

-----

## Metadata Text Conversion

Firebird versions 2.0.x had two problems related to character sets and metadata extraction:

1. When creating or altering objects, text associated with metadata was not transliterated from the client character set to the system (UNICODE\_FSS) character set of these BLOB columns. Instead, raw bytes were stored there.

The types of text affected were PSQL sources, descriptions, text associated with constraints and defaults, and so on.

**Note**

Even in the current version (2.1 Beta 1) the problem can still occur if CREATE or ALTER operations are performed with the connection character set as NONE or UNICODE\_FSS and you are using non-UNICODE\_FSS data.

2. In reads from text BLOBs, transliteration from the BLOB character set to the client character set was not being performed.

## ***Repairing Your Metadata Text***

If your metadata text was created with non-ASCII encoding, you need to repair your database in order to read the metadata correctly after upgrading it to v.2.1.

**Important**

The procedure involves multiple passes through the database, using scripts. It is strongly recommended that you disconnect and reconnect before each pass.

The database should already have been converted to ODS11.1 by way of a gbak backup and restore.

Before doing anything, make a copy of the database.

## ***Create the procedures in the database***

```
[1] isql /path/to/your/database.fdb
[2] SQL> input 'misc/upgrade/metadata_charset_create.sql';
```

## ***Check your database***

```
[1] isql /path/to/your/database.fdb
[2] SQL> select * from rdb$check_metadata;
```

The `rdb$check_metadata` procedure will return all objects that are touched by it.

- If no exception is raised, your metadata is OK and you can go to the section “[Remove the upgrade procedures](#)”.
- Otherwise, the first bad object is the last one listed before the exception.

## ***Fixing the metadata***

To fix the metadata, you need to know in what character set the objects were created. The upgrade script will work correctly only if all your metadata was created using the same character set.

```
[1] isql /path/to/your/database.fdb
```

```
[2] SQL> input 'misc/upgrade/metadata_charset_create.sql';  
[3] SQL> select * from rdb$fix_metadata('WIN1252'); -- replace WIN1252 by your charset  
[4] SQL> commit;
```

The *rdb\$fix\_metadata* procedure will return the same data as *rdb\$check\_metadata*, but it will change the metadata texts.

**Important**

It should be run once!

After this, you can remove the upgrade procedures.

### ***Remove the upgrade procedures***

```
[1] isql /path/to/your/database.fdb  
[2] SQL> input 'misc/upgrade/metadata_charset_drop.sql';
```



---

## Chapter 8

# Utility Programs

In this chapter are the additions and improvements that have been added to Firebird's set of command-line utilities in the v.2.1 development phase. Of note is the new *fbsvcmgr* utility that enables access to the Services API from a command shell.

## New Command-line Utility *fbsvcmgr*

Alex Peshkov

The new utility *fbsvcmgr* provides a command-line interface to the Services API, providing access to any service that is implemented in Firebird.

Although there are numerous database administration tools around that surface the Services API through graphical interfaces, the new tool addresses the problem for admins needing to access remote Unix servers in broad networks through a text-only connection. Previously, meeting such a requirement needed a programmer.

## Using *fbsvcmgr*

*fbsvcmgr* does not emulate the switches implemented in the traditional “g\*” utilities. Rather, it is just a front-end through which the Services API functions and parameters can pass. Users therefore need to be familiar with the Services API as it stands currently. The API header file—*ibase.h*, in the *../include* directory of your Firebird installation—should be regarded as the primary source of information about what is available, backed up by the InterBase 6.0 beta API Guide.

## Parameters

### *Specify the Services Manager*

The first required parameter for a command line call is the Services Manager you want to connect to:

- For a local connection use the simple symbol `service_mgr`
- To attach to a remote host, use the format `hostname:service_mgr`

### *Specify subsequent service parameter blocks (SPBs)*

Subsequent SPBs, with values if required, follow. Any SPB can be optionally prefixed with a single `'` symbol. For the long command lines that are typical for *fbsvcmgr*, use of the `'` improves the readability of the command line. Compare, for example, the following (each a single command line despite the line breaks printed here):

```
# fbsvcmgr service_mgr user sysdba password masterke  
    action_db_stats dbname employee sts_hdr_pages
```

and

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -action_db_stats -dbname employee -sts_hdr_pages
```

## SPB Syntax

The SPB syntax that fbsvcmgr understands closely matches with what you would encounter in the `ibase.h` include file or the InterBase 6.0 API documentation, except that a slightly abbreviated form is used to reduce typing and shorten the command lines a little. Here's how it works.

All SPB parameters have one of two forms: (1) `isc_spb_VALUE` or (2) `isc_VALUE1_svc_VALUE2`. For fbsvcmgr you just need to pick out the `VALUE`, `VALUE1` or `VALUE2` part[s] when you supply your parameter.

Accordingly, for (1) you would type simply `VALUE`, while for (2) you would type `VALUE1_VALUE2`. For example:

```
isc_spb_dbname => dbname
isc_action_svc_backup => action_backup
isc_spb_sec_username => sec_username
isc_info_svc_get_env_lock => info_get_env_lock
```

and so on.

### Note

An exception is `isc_spb_user_name`: it can be specified as either `user_name` or simply `user`.

It is not realistic to attempt to describe all of the SPB parameters in release notes. In the InterBase 6.0 beta documentation it takes about 40 pages! The next section highlights some known differences between the operation of fbsvcmgr and what you might otherwise infer from the old beta documentation.

## fbsvcmgr Syntax Specifics

### “Do's and Don'ts”

With fbsvcmgr you can perform a single action—and get its results if applicable—or you can use it to retrieve multiple information items from the Services Manager. You cannot do both in a single command.

For example,

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -action_display_user
```

will list all current users on the local firebird server:

SYSDBA	Sql Server Administrator	0	0
QA_USER1		0	0
QA_USER2		0	0
QA_USER3		0	0
QA_USER4		0	0
QA_USER5		0	0
GUEST		0	0
SHUT1		0	0

SHUT2	0	0
QATEST	0	0

...and...

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -info_server_version -info_implementation
```

will report both the server version and its implementation:

```
Server version: LI-T2.1.0.15740 Firebird 2.1 Alpha 1
Server implementation: Firebird/linux AMD64
```

But an attempt to mix all of this in single command line:

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -action_display_user -info_server_version -info_implementation
```

raises an error:

```
Unknown switch "-info_server_version"
```

#### *Undocumented Items*

The function `isc_spb_rpr_list_limbo_trans` was omitted from the IB6 beta documentation. It is supported in `fbsvcmgr`.

#### *New Services API Items in v.2.1*

Two new items were added to Firebird 2.1 and are supported by `fbsvcmgr`:

- `isc_spb_trusted_auth` (type it as `trusted_auth`) applies only to Windows. It forces Firebird to use Windows trusted authentication.
- Ability to set a database name parameter (`[isc_spb_]dbname`) in all actions related to the security database, equivalent to supplying the `-database` switch to the `gsec` utility.

#### **Note**

For `gsec` the `-database` switch is mostly used to specify a remote server you want to administer. In `fbsvcmgr`, the name of the server is already given in the first parameter (via the `service_mgr` symbol) so the `[isc_spb_]dbname` parameter is mostly unnecessary.

#### *Documentation Bugs*

The forms supplied for some parameters in InterBase 6 beta documentation are buggy. When in trouble, treat `ibase.h` as the primary source for the correct form.

#### *Unsupported functions*

- Everything to do with licensing was removed from the original InterBase 6 open source code and is therefore not supported either in Firebird or by `fbsvcmgr`.

- The old Config file view/modification functions have been unsupported since Firebird 1.5 and are not implemented by fbsvcmgr.

## Improvements to Utilities

A number of improvements were made to several utilities.

### *Utilities Support for Database Triggers*

A new parameter was added to gbak, nbackup and isql to suppress [Database Triggers](#) from running. It is available only to the database owner and SYSDBA:

```
gbak -no_dbtriggers
isql -nodbtriggers
nbackup -T
```

### *gbak*

#### *gbak Made More Version-friendly*

C. Valderrama

V.2.1 gbak can be used to restore a database on any version of Firebird.

#### *Hide User Name & Password in Shell*

A. Peshkov

[Feature request CORE-867](#)

GBAK now changes param0 to prevent the user name and password from being displayed in `ps -axf`.

### *isql*

#### *Ctrl-C to cancel query output*

M. Kubecek

A. dos Santos Fernandes

[Feature request CORE-704](#)

Output from a SELECT in an interactive isql session can now be stopped using Ctrl-C. Note, this merely stops fetching rows from the buffer, it does not cancel the query.

### ***Extension of isql SHOW SYSTEM command***

A. dos Santos Fernandes

[Feature request CORE-978](#)

See v.2.0.1 Release Notes.

## ***Services Manager***

### ***Fixed Some Misbehaviour***

A. Peshkov

[Feature request CORE-1232](#)

Fixed some misbehaviour of the Services Manager during backup/restore operations.

### ***Disable Non-SYSDBA Use***

A. Peshkov

[Feature request CORE-787](#)

Optionally disable non-SYSDBA use of Services API. [How?]

## ***Builds and Installs***

### ***Parameter for Instance name added to instsvc.exe***

D. Yemanov

[Feature request CORE-673](#)

instsvc.exe now supports multi-instance installations. More guff?

### ***Revised Win32 Installer Docs***

P. Reeves

See install\_windows\_manually.txt. More guff?

### ***Gentoo/FreeBSD detection during install***

A. Peshkov

[Feature request CORE-1047](#)

More details in v.2.0.1 release notes.

---

## Chapter 9

# Security

In this chapter are the additions and improvements that have been added to Firebird's security since v.2.0.x.

## Using Windows Security to Authenticate Users

Alex Peshkov

From Firebird 2.1 onward, Windows “Trusted User” security can be applied for authenticating Firebird users on a Windows host. The Trusted User's security context is passed to the Firebird server and, if it succeeds, it is used to determine the Firebird security user name.

Simply omitting the user and password parameters from the DPB/SPB will automatically cause Windows Trusted User authentication to be applied, in almost all cases. See the Environment section, below, for exceptions.

### Illustration

Suppose you have logged in to the Windows server SRV as user 'John'. If you connect to server SRV with *isql*, without specifying a Firebird user name and password:

```
isql srv:employee
```

and do:

```
SQL> select CURRENT_USER from rdb$database;
```

you will get something like:

```
USER
=====
SRV\John
```

## SQL Privileges

Windows users can be granted rights to access database objects and roles in the same way as regular Firebird users, emulating the capability that has been always been available users of Unix and Linux hosted Firebird databases.

## Administrators

If a member of the built-in Domain Admins group connects to Firebird using trusted authentication, he/she will be connected as SYSDBA.

## Configuration Parameter “Authentication”

The new parameter `Authentication` has been added to `firebird.conf` for configuring the authentication method on Windows. Possible values are:-

*Authentication = Native*

Provides full compatibility with previous Firebird versions, avoiding trusted authentication.

*Authentication = Trusted*

The Security database is ignored and only Windows authentication is used. In some respects, on Windows this is more secure than Native, in the sense that it is no less and no more secure than the security of the host operating system.

*Authentication = Mixed*

This is the default setting.

To retain the legacy behaviour, when the `ISC_USER` and `ISC_PASSWORD` variables are set in the environment, they are picked and used instead of trusted authentication.

### Note

Trusted authentication can be coerced to override the environment variables if they are set—refer to the notes below.

## Forcing Trusted Authentication

For the situation where trusted authentication is needed and there is a likelihood that `ISC_USER` and `ISC_PASSWORD` are set, there is a new DPB parameter that you can add to the DPB—`isc_dpb_trusted_auth`.

Most of the Firebird command-line utilities support parameter by means of the switch `-tru[sted]` (the abbreviated form is available, according to the usual rules for abbreviating switches).

### Note

The *qli* and *nbackup* utilities do not follow the pattern: they use single-letter switches that are somewhat arcane. The switch of interest for *qli* is `-K`). For *nbackup*, watch this space. The facility to force trusted authentication is yet to be implemented for it.

## Example

```
C:\Pr~\bin>isql srv:db          -- log in using trusted authentication
C:\Pr~\bin>set ISC_USER=user1
C:\Pr~\bin>set ISC_PASSWORD=12345
C:\Pr~\bin>isql srv:db          -- log in as 'user1' from environment
C:\Pr~\bin>isql -trust srv:db   -- log in using trusted authentication
```

## Other Security Improvements

### *isc\_service\_query() wrongly reveals full database file spec*

[Feature request CORE-1091](#)

When the server is configured "DatabaseAccess = None", `isc_service_query()` should return an alias name instead of a full database file name. This seems more like "fixing a bug of omission" than introducing an improvement.

### *Any user could view the server log through the Services API*

[Feature request CORE-1148](#)

This was a minor security vulnerability. Regular users are now blocked from retrieving the server log using the Services API. Requests are explicitly checked to ensure that the authenticated user is SYSDBA.



# Bugs Fixed

## Firebird 2.1 Beta 2

The following section details the bug fixes that have been applied since the Beta 1 release:

### Core Engine/DSQL

([CORE-1476](#)) Forced writes have never actually worked on Linux, leaving open the potential for system trauma to break databases even with FW=ON. It has actually been known to happen on Linux.

*fixed by A. Peshkov*

~ ~ ~

([CORE-1468](#)) Database corruption was possible when database file expansion and read/write activity were being performed simultaneously.

*fixed by V. Horsun*

~ ~ ~

([CORE-1440](#)) Transaction options were dangerously lacking in validation.

*fixed by C. Valderrama*

~ ~ ~

([CORE-1418](#)) Rapidly starting and shutting down could cause a race condition in the blocking AST thread due to poor synchronization.

*fixed by A. Peshkov*

~ ~ ~

([CORE-1401](#)) Instances of a global temporary table were not always picking up all indices.

*fixed by V. Horsun*

~ ~ ~

([CORE-1361](#)) Index operations for global temporary tables were not visible to the active connection.

*fixed by V. Horsun*

~ ~ ~

([CORE-1380](#)) Changing the Forced Writes setting for a database would cause I/O errors if the database had existing attachments.

*fixed by V. Horsun*

~ ~ ~

([CORE-1408](#)) UDF names using reserved words were being extracted with the double quotes missing.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1379](#)) Invalid parameter type ("Data type unknown" error) when passing the argument to the CHAR\_LENGTH function as a parameter.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1347](#)) Unexpected "cannot transliterate" error.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1332](#)) The SQLSCALE member of a text BLOB column can carry the BLOB's character set. In some documentation it wrongly says it should always be there. Text BLOBs needed to be brought into line with character types, i.e., if the connection character set is other than NONE and the BLOB's character set is not NONE or OCTETS, then it should be the character set of the connection.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Server Crashes

([CORE-1470](#)) The server would crash if a secondary file name was longer than 127 characters.

*fixed by C. Valderrama*

~ ~ ~

([CORE-1457](#)) The server would crash when attempting to deliver events for a session that had just disconnected.

*fixed by V. Horsun, D. Yemanov*

~ ~ ~

([CORE-1451](#)) Using RDB\$DB\_KEY in the **WHERE** clause of a SELECT from a stored procedure would crash the server.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1338](#)) Connection lost (error 335544721, Unable to complete network request to host ...) when selecting from a view having a derived field defined with ROUND().

*fixed by D. Yemanov*

~ ~ ~

([CORE-1334](#)) Joins with a NULL RDB\$DB\_KEY would crash the server.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Windows-Specific

([CORE-1456](#)) Wrong events delivery could occur where there were concurrent XNET connections.

*fixed by V. Horsun, D. Yemanov*

~ ~ ~

([CORE-1443](#)) On 64-bit Windows 2003 Server, the embedded engine could cause an application to hang on exit if no database access was performed.

*fixed by V. Horsun*

~ ~ ~

([CORE-1403](#)) The server under Windows would crash if multiple events were being registered simultaneously by a client connected via the XNET protocol.

*fixed by D. Yemanov*

~ ~ ~

## Data Definition Language (DDL)

([CORE-1395](#)) CHECK constraints on domains were demonstrating a few problems.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1378](#)) A number of issues were reported regarding domain names and character sets.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Data Manipulation Language (DML)

([CORE-1466](#)) The SUBSTRING() function could return a truncated substring for some multi-byte BLOBs.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1428](#))      Timestamp subtraction in dialect 3 was incorrect if the calculation would result in a negative number.

*fixed by V. Horsun*

~ ~ ~

([CORE-1417](#))      Error “Invalid BLOB ID” error could occur when performing an insert using InterBaseXpress.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1373](#))      A recursive CTE query would produce incorrect results when the recursive member's SELECT list contained an expression involving self-referencing fields.

*fixed by V. Horsun*

~ ~ ~

## **Procedural Language (PSQL)**

([CORE-1434](#))      EXECUTE STATEMENT was truncating the last two bytes of VARCHAR columns.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1419](#))      CURRENT\_TIMESTAMP was being wrongly evaluated during the execution of selectable procedures.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1371](#))      An EXECUTE BLOCK sequence would fail if it was passed within an EXECUTE STATEMENT string.

*fixed by A. Peshkov*

~ ~ ~

([CORE-1370](#))      Use of CTE within procedures was causing memory leaks.

*fixed by V. Horsun*

~ ~ ~

([CORE-1331](#))      Character set transliterations would not work with EXECUTE STATEMENT.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Remote Interface

([CORE-1455](#)) Crash in fbclient after an unsuccessful user management API call.

*fixed by A. Peshkov*

~ ~ ~

([CORE-1452](#)) The client library would crash when attempting to process an event received just before disconnection. (Did not affect libfbembed.so.)

*fixed by D. Yemanov, V. Horsun*

~ ~ ~

([CORE-1430](#)) Access Violation in fbclient.dll if a statement was prepared and executed right after events were registered.

*fixed by V. Horsun*

~ ~ ~

([CORE-1388](#)) It was not possible to attach to the Service Manager remotely if the remote engine version was less than 2.0.

*fixed by V. Horsun*

~ ~ ~

([CORE-1349](#)) The remote interface was not validating the client-supplied message length against the message format length.

*fixed by V. Horsun*

~ ~ ~

## API

([CORE-1485](#)) Fixed a very ancient bug whereby the **sqlen** field in the xsqlvar contained length of data in a varying structure, not its total size. In any OS environment it could cause an access violation when loading messages in msg.fdb

*fixed by A. Peshkov*

~ ~ ~

([CORE-1416](#)) Incorrect parameter order in a TPB was being accepted without returning an error.

*fixed by C. Valderrama*

~ ~ ~

([CORE-1372](#)) If `isc_dsqli_fetch()` is called after `isc_commit_transaction()` an exception should be raised. That was not happening.

*fixed by V. Horsun*

~ ~ ~

## International Language Support (INTL)

([CORE-1484](#)) INTL modules compiled in the Linux, gcc 4.1.2, amd64 environment would cause an access violation in Superserver, due to use of the standard operator **new** but the overloaded operator **delete**.

*fixed by A. Peshkov*

~ ~ ~

([CORE-1446](#)) Problem with UNICODE collations from fbintl when using system ICU.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1431](#)) There were some inherent issues with uppercasing certain Greek characters in cp1251.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1384](#)) LIKE would not work correctly with collations using SPECIALS-FIRST=1.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1339](#)) The metadata character set upgrade script was generating garbage in descriptions.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Database Monitoring/Admin

([CORE-1467](#)) A database attachment would go into some kind of invalid state after its long-running statement was canceled via MON\$STATEMENTS, returning a 'database shutdown' error.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1441](#)) Query cancellation feature could not interrupt a long fetch.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1436](#)) Outer joins would not work properly with the MON\$ tables.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1359](#)) The server would crash at the first operation with the monitoring tables if the filesystem lacked the necessary permissions for the shared-memory file.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1358](#)) Operations with MON\$STATEMENTS were throwing "cannot transliterate" errors.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1330](#)) Semaphores were being double-locked when the monitoring tables were queried during long fetches.

*fixed by D. Yemanov*

~ ~ ~

## Security

([CORE-1447](#)) Querying for database info on very long path through an isc\_database\_info() API call could cause a buffer overrun.

*fixed by C. Valderrama*

~ ~ ~

([CORE-1397](#)) A possible vulnerability was discovered in the remote server attachment.

*fixed by V. Horsun*

~ ~ ~

([CORE-1312](#)) A remote attacker could check for the presence of a file on a system running the Firebird server.

*fixed by A. Peshkov*

~ ~ ~

## Command-line Utilities

### gstat

([CORE-1400](#)) GSTAT did not support infixing the port number in the connection string.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1399](#)) GSTAT would not use the RemoteServicePort configured in firebird.conf.

*fixed by D. Yemanov*

~ ~ ~

([CORE-1398](#)) GSTAT was treating 'localhost' as case-sensitive in Windows.

*fixed by D. Yemanov*

~ ~ ~

### **gbak**

([CORE-1369](#)) Default values of procedure parameters were not being caught when downgrading a database from ODS11.1.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1344](#)) Error "request depth exceeded" when restoring complex metadata.

*fixed by D. Yemanov*

~ ~ ~

### **isql**

([CORE-1465](#)) ISQL would ignore an explicit constraint name when it was confused with an internal, automatic name.

*fixed by C. Valderrama*

~ ~ ~

([CORE-1261](#)) isql would ignore the index and ordering in a UNIQUE CONSTRAINT when generating a metadata script.

*fixed by C. Valderrama*

~ ~ ~

## **Firebird 2.1 Beta 1**

The following are rough groupings to help you find specific bug fixes that you want to check up on. In general, expect these to be fixes that were deferred at the 2.0 release or showed up as regressions after a 2.0.x or 2.1 Alpha release.



## Core Engine/DSQL

( [CORE-1248](#))      Incorrect timestamp arithmetic was performed when one of the operands was a negative number.

*fixed by V. Horsun*

~ ~ ~

( [CORE-1228](#))      Reports of database corruption after an out-of-disk-space condition.

*fixed by V. Horsun*

~ ~ ~

( [CORE-1227](#))      LIST() function would seem to fail if used twice or more in a query.

*fixed by A. dos Santos Fernandes*

~ ~ ~

( [CORE-1215](#))      Wrong SELECT query results using index to evaluate >= condition

*fixed by V. Horsun*

~ ~ ~

( [CORE-1175](#))      Error “Data type unknown” when any UDF argument was a built-in function containing a DSQL parameter reference.

*fixed by D. Yemanov*

~ ~ ~

## Server Crashes

( [CORE-1244](#))      Server crash on “select \* from <recursive CTE>”.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## Win32-Specific

( [CORE-1207](#))      FB embedded would not load without extra OS privileges.

*fixed by V. Horsun*

~ ~ ~

## POSIX-Specific

( [CORE-1240](#)) Any task on Darwin PPC that used libfbclient would hang on exit.

*fixed by A. Peshkov*

~ ~ ~

( [CORE-1223](#)) Wrong message in firebird.log on Open SuSe Linux 10.2 : Open file limit increased from 1024 to 0.

*fixed by V. Horsun*

~ ~ ~

## Data Definition Language (DDL)

( [CORE-1292](#)) CREATE TABLE failed if using long username and UTF8 as attachment charset.

*fixed by A. dos Santos Fernandes*

~ ~ ~

( [CORE-1271](#)) Engine was allowing creation of invalid procedures and triggers.

*fixed by A. dos Santos Fernandes*

~ ~ ~

( [CORE-1183](#)) View could not be created if its WHERE clause contained IN <subquery> with a procedure reference.

*fixed by D. Yemanov*

~ ~ ~

( [CORE-1162](#)) Problem altering numeric field type.

*fixed by C. Valderrama*

~ ~ ~

## Data Manipulation Language (DML)

( [CORE-1253](#)) LIST(DISTINCT) was concatenating VARCHAR values as CHAR

*fixed by A. dos Santos Fernandes*

~ ~ ~

( [CORE-1153](#)) Activating an index change would cause “STARTING” to work as “LIKE” in a join condition.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## **Procedural Language (PSQL)**

([CORE-1267](#)) Small bug with default value for domains in PSQL

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1256](#)) Table columns were hiding the destination variables for RETURNING INTO.

*fixed by A. dos Santos Fernandes*

~ ~ ~

([CORE-1165](#)) WHEN <list of exceptions> was tracking dependencies only on the first exception in PSQL.

*fixed by C. Valderrama*

~ ~ ~

## **Remote Interface**

([CORE-1218](#)) isc\_dsql\_info(isc\_info\_sql\_stmt\_type) did not set isc\_info\_end at the end of the passed user's buffer

*fixed by V. Horsun*

~ ~ ~

([CORE-1196](#)) Long SQL statements were breaking the TCP/IP connection.

*fixed by V. Horsun, A. Peshkov, D. Yemanov*

~ ~ ~

## **Security**

([CORE-885](#)) It was impossible to revoke rights on update of a column.

*fixed by A. Peshkov*

~ ~ ~

([CORE-856](#)) Could not set FName, MName, LName fields in the Security database to blank.

*fixed by A. Peshkov*

~ ~ ~

## Utilities

### nBackup

( [CORE-1151](#) )      Error “database file not available” when running NBackup.

*fixed by N. Samofatov*

~ ~ ~

### isql

( [CORE-703](#) )      Using DEL-Key in isql under Linux would give “~”

*fixed by A. Peshkov*

~ ~ ~

### gbak

( [CORE-1237](#) )      gbak would fail to create a backup in service\_mgr mode if there was no space on disk, but reported no error.

*fixed by A. Peshkov*

~ ~ ~

( [CORE-1205](#) )      v2.1 gbak would crash the server when attempting to perform a backup.

*fixed by D. Yemanov, C. Valderrama*

~ ~ ~

( [CORE-1174](#) )      gbak would restore NULL rdb\$description in rdb\$functions as blob (0, 0).

*fixed by C. Valderrama*

~ ~ ~

( [CORE-949](#) )      Restore would fail with a UDF call in a 'COMPUTED BY' field.

*fixed by D. Sibiryakov*

~ ~ ~

( [CORE-132](#) )      Restore would fail on external table.

*fixed by V. Horsun*

~ ~ ~

## **gfix**

( [CORE-1249](#) ) Full shutdown mode failed on Classic if there were other connections to the database.

*fixed by D. Yemanov*

~ ~ ~

## **Building/Installers**

( [CORE-981](#) ) x86\_64 RPM package missing “provides”.

*fixed by A. Peshkov*

~ ~ ~

( [CORE-107](#) ) An instance of fb\_lock\_mgr would be left running after a build.

*fixed by A. Peshkov*

~ ~ ~

## **Fixed Regressions**

( [CORE-1286](#) ) Bug with COMPUTED BY fields.

*fixed by A. dos Santos Fernandes*

~ ~ ~

( [CORE-1167](#) ) Character set GBK was not getting installed.

*fixed by A. dos Santos Fernandes*

~ ~ ~

## **Not Fixed**

( [CORE-1079](#) ) Every attach of fbclient/fbembed library to the host process leaks 64KB of memory

No information available.

~ ~ ~

# Appendix A:

## New Built-in Functions, Firebird 2.1

Function	Format	Description
<i>ABS</i>	<i>ABS( &lt;number&gt; )</i>	Returns the absolute value of a number.
<code>select abs(amount) from transactions;</code>		
<i>ACOS</i>	<i>ACOS( &lt;number&gt; )</i>	Returns the arc cosine of a number. Argument to ACOS must be in the range -1 to 1. Returns a value in the range 0 to PI.
<code>select acos(x) from y;</code>		
<i>ASCII_CHAR</i>	<i>ASCII_CHAR( &lt;number&gt; )</i>	Returns the ASCII character with the specified code. The argument to ASCII_CHAR must be in the range 0 to 255. The result is returned in character set NONE.
<code>select ascii_char(x) from y;</code>		
<i>ASCII_VAL</i>	<i>ASCII_VAL( &lt;string&gt; )</i>	Returns the ASCII code of the first character of the specified string.  <ol style="list-style-type: none"><li>Returns 0 if the string is empty</li><li>Throws an error if the first character is multi-byte</li></ol>
<code>select ascii_val(x) from y;</code>		
<i>ASIN</i>	<i>ASIN( &lt;number&gt; )</i>	Returns the arc sine of a number. The argument to ASIN must be in the range -1 to 1. It returns a result in the range -PI/2 to PI/2.
<code>select asin(x) from y;</code>		
<i>ATAN</i>	<i>ATAN( &lt;number&gt; )</i>	Returns the arc tangent of a number. Returns a value in the range -PI/2 to PI/2.
<code>select atan(x) from y;</code>		
<i>ATAN2</i>	<i>ATAN2( &lt;number&gt;, &lt;number&gt; )</i>	

Function	Format	Description
		Returns the arc tangent of the first number / the second number. Returns a value in the range -PI to PI.
select atan2(x, y) from z;		
<i>BIN_AND</i>	BIN_AND( <number> [, <number> ...] )	Returns the result of a binary AND operation performed on all arguments.
select bin_and(flags, 1) from x;		
<i>BIN_OR</i>	BIN_OR( <number> [, <number> ...] )	Returns the result of a binary OR operation performed on all arguments.
select bin_or(flags1, flags2) from x;		
<i>BIN_SHL</i>	BIN_SHL( <number>, <number> )	Returns the result of a binary shift left operation performed on the arguments (first << second).
select bin_shl(flags1, 1) from x;		
<i>BIN_SHR</i>	BIN_SHR( <number>, <number> )	Returns the result of a binary shift right operation performed on the arguments (first >> second).
select bin_shr(flags1, 1) from x;		
<i>BIN_XOR</i>	BIN_XOR( <number> [, <number> ...] )	Returns the result of a binary XOR operation performed on all arguments.
select bin_xor(flags1, flags2) from x;		
<i>CEIL / CEILING</i>	{ CEIL   CEILING }( <number> )	Returns a value representing the smallest integer that is greater than or equal to the input argument.
1) select ceil(val) from x; 2) select ceil(2.1), ceil(-2.1) from rdb\$database; -- returns 3, -2		
<i>COS</i>	COS( <number> )	Returns the cosine of a number. The angle is specified in radians and returns a value in the range -1 to 1.
select cos(x) from y;		
<i>COSH</i>	COSH( <number> )	Returns the hyperbolic cosine of a number.

Function	Format	Description
<pre>select cosh(x) from y;</pre>		
<i>COT</i>	COT( <number> )	Returns 1 / tan(argument).
<pre>select cot(x) from y;</pre>		
<i>DATEADD</i>	See below	Returns a date/time/timestamp value increased (or decreased, when negative) by the specified amount of time.
<p>Format:</p> <pre>DATEADD( &lt;number&gt; &lt;timestamp_part&gt; FOR &lt;date_time&gt; ) /* Do not use the above (expanded) syntax for any permanent code.    The keyword FOR will be replaced with TO */ DATEADD( &lt;timestamp_part&gt;, &lt;number&gt;, &lt;date_time&gt; ) timestamp_part ::= { YEAR   MONTH   DAY   WEEKDAY                        HOUR   MINUTE   SECOND   MILLISECOND }</pre> <ol style="list-style-type: none"> <li>1. YEAR, MONTH, DAY and WEEKDAY cannot be used with time values.</li> <li>2. HOUR, MINUTE, SECOND and MILLISECOND cannot be used with date values.</li> <li>3. All timestamp_part values can be used with timestamp values.</li> <li>4. See the note above regarding the planned change of keyword in the expanded form of this function call.</li> </ol> <p>Example</p> <pre>select dateadd(-1, day, current_date) as yesterday from rdb\$database;</pre>		
<i>DATEDIFF</i>	See below	Returns an exact numeric value representing the interval of time from the first date/time/timestamp value to the second one.
<p>Format:</p> <pre>/* Expanded form: do not use, as it will be changed! */ DATEDIFF( &lt;timestamp_part&gt; FROM &lt;date_time&gt; FOR &lt;date_time&gt; ) /* Re-implemented expanded form, not available in Beta 2 */ DATEDIFF( &lt;timestamp_part&gt; FROM &lt;date_time&gt; TO &lt;date_time&gt; ) /* Contracted form: OK to use */ DATEDIFF( &lt;timestamp_part&gt;, &lt;date_time&gt;, &lt;date_time&gt; ) timestamp_part ::= { YEAR   MONTH   DAY   WEEKDAY                        HOUR   MINUTE   SECOND   MILLISECOND }</pre>		



Function	Format	Description
1. Returns a positive value if the second value is greater than the first one, negative when the first one is greater, or zero when they are equal. 2. Comparison of date with time values is invalid. 3. YEAR, MONTH, DAY and WEEKDAY cannot be used with time values. 4. HOUR, MINUTE, SECOND and MILLISECOND cannot be used with date values. 5. All timestamp_part values can be used with timestamp values. **** NOTE **** 6. The keywords implemented in this beta for the expanded form are being changed in the next Beta to improve the semantics.  Example  <pre>select datediff(HOUR, ('TOMORROW' -10), current_date)   as datediffresult from rdb\$database;</pre>		
<i>DECODE</i>	See below	DECODE is a shortcut for a CASE ... WHEN ... ELSE expression.
Format:  DECODE( <expression>, <search>, <result> [ , <search>, <result> ... ] [, <default> ]  Example  <pre>select decode(state, 0, 'deleted', 1, 'active', 'unknown') from things;</pre>		
<i>EXP</i>	EXP( <number> )	Returns the exponential e to the argument.
<pre>select exp(x) from y;</pre>		
<i>FLOOR</i>	FLOOR( <number> )	Returns a value representing the largest integer that is less than or equal to the input argument.
<pre>1) select floor(val) from x; 2) select floor(2.1), floor(-2.1)    from rdb\$database; -- returns 2, -3</pre>		
<i>GEN_UUID</i>	GEN_UUID() -- no arguments	Returns a universal unique number.
<pre>insert into records (id) value (gen_uuid());</pre>		
<i>HASH</i>	HASH( <value> )	Returns a HASH of a value.

Function	Format	Description
<code>select hash(x) from y;</code>		
<i>LEFT</i>	LEFT( <string>, <number> )	Returns the substring of a specified length that appears at the start of a left-to-right string.
<code>select left(name, char_length(name) - 10) from people where name like '% FERNANDES';</code>		
<i>LN</i>	LN( <number> )	Returns the natural logarithm of a number.
<code>select ln(x) from y;</code>		
<i>LOG</i>	LOG( <number>, <number> )	LOG(x, y) returns the logarithm base x of y.
<code>select log(x, 10) from y;</code>		
<i>LOG10</i>	LOG10( <number> )	Returns the logarithm base ten of a number.
<code>select log10(x) from y;</code>		
<i>LPAD</i>	LPAD( <string>, <number> [, <string> ] )	LPAD(string1, length, string2) prepends string2 to the beginning of string1 until the length of the result string becomes equal to length.
<ol style="list-style-type: none"> <li>1. If the second string is omitted the default value is one space.</li> <li>2. If the result string would exceed the length, the second string is truncated.</li> </ol> <p>Example</p> <code>select lpad(x, 10) from y;</code>		
<i>MAXVALUE</i>	MAXVALUE( <value> [, <value> ... ] )	Returns the maximum value of a list of values.
<code>select maxvalue(v1, v2, 10) from x;</code>		
<i>MINVALUE</i>	MINVALUE( <value> [, <value> ... ] )	Returns the minimum value of a list of values.
<code>select minvalue(v1, v2, 10) from x;</code>		

Function	Format	Description
<i>MOD</i>	MOD( <number>, <number> )	Modulo: MOD(X, Y) returns the remainder part of the division of X by Y.
<pre>select mod(x, 10) from y;</pre>		
<i>OVERLAY</i>	See below	Returns string1 replacing the substring FROM start FOR length by string2.
<p>Format:</p> <pre>OVERLAY( &lt;string1&gt; PLACING &lt;string2&gt; FROM &lt;start&gt; [ FOR &lt;length&gt; ] )</pre> <p>The OVERLAY function is equivalent to:</p> <pre>SUBSTRING(&lt;string1&gt;, 1 FOR &lt;start&gt; - 1)    &lt;string2&gt;    SUBSTRING(&lt;string1&gt;, &lt;start&gt; + &lt;length&gt;)</pre> <p>If &lt;length&gt; is not specified, CHAR_LENGTH( &lt;string2&gt; ) is implied.</p>		
<i>PI</i>	PI() -- no arguments	Returns the PI constant (3.1459...).
<pre>val = PI();</pre>		
<i>POSITION</i>	POSITION( <string> IN <string> )	POSITION(X IN Y) returns the position of the substring X in the string Y. Returns 0 if X is not found within Y.
<pre>select rdb\$relation_name   from rdb\$relations  where position('RDB\$' IN rdb\$relation_name) = 1;</pre>		
<i>POWER</i>	POWER( <number>, <number> )	POWER(X, Y) returns X to the power of Y.
<pre>select power(x, 10) from y;</pre>		
<i>RAND</i>	RAND() -- no argument	Returns a random number between 0 and 1.
<pre>select * from x order by rand();</pre>		
<i>REPLACE</i>	REPLACE( <stringtosearch>, <findstring>, <replstring> )	Replaces all occurrences of <findstring> in <stringtosearch> with <replstring>.

Function	Format	Description
<pre>select replace(x, ' ', ',') from y;</pre>		
<i>REVERSE</i>	REVERSE( <value> )	Returns a string in reverse order. Useful function for creating an expression index that indexes strings from right to left.
<pre>create index people_email on people   computed by (reverse(email)); select * from people   where reverse(email) starting with reverse('.br');</pre>		
<i>RIGHT</i>	RIGHT( <string>, <number> )	Returns the substring, of the specified length, from the right-hand end of a string.
<pre>select right(rdb\$relation_name, char_length(rdb\$relation_name) - 4)   from rdb\$relations   where rdb\$relation_name like 'RDB\$%';</pre>		
<i>ROUND</i>	ROUND( <number>, <number> )	Returns a number rounded to the specified scale.
<p>Example</p> <pre>select round(salary * 1.1, 0) from people;</pre> <p>If the scale (second parameter) is negative, the integer part of the value is rounded. E.g., ROUND(123.456, -1) returns 120.000.</p>		
<i>RPAD</i>	RPAD( <string1>, <length> [, <string2> ] )	Appends <string2> to the end of <string1> until the length of the result string becomes equal to <length>.
<p>Example</p> <pre>select rpad(x, 10) from y;</pre> <ol style="list-style-type: none"> <li>1. If the second string is omitted the default value is one space.</li> <li>2. If the result string would exceed the length, the final application of &lt;string2&gt; will be truncated.</li> </ol>		
<i>SIGN</i>	SIGN( <number> )	Returns 1, 0, or -1 depending on whether the input value is positive, zero or negative, respectively.
<pre>select sign(x) from y;</pre>		

Function	Format	Description
<i>SIN</i>	SIN( <number> )	Returns the sine of an input number that is expressed in radians.
select sin(x) from y;		
<i>SINH</i>	SINH( <number> )	Returns the hyperbolic sine of a number.
select sinh(x) from y;		
<i>SQRT</i>	SQRT( <number> )	Returns the square root of a number.
select sqrt(x) from y;		
<i>TAN</i>	TAN( <number> )	Returns the tangent of an input number that is expressed in radians.
select tan(x) from y;		
<i>TANH</i>	TANH( <number> )	Returns the hyperbolic tangent of a number.
select tanh(x) from y;		
<i>TRUNC</i>	TRUNC( <number> [, <number> ] )	Returns the integral part (up to the specified scale) of a number.
1) select trunc(x) from y; 2) select trunc(-2.8), trunc(2.8) from rdb\$database; -- returns -2, 2 3) select trunc(987.65, 1), trunc(987.65, -1) from rdb\$database; -- returns 987.60, 980.00		

---

# Appendix B:

## INTL Character Sets

Listing of Character Sets supported.

### *Narrow Character Sets*

CYRL,  
DOS437, DOS737, DOS775, DOS850, DOS852, DOS857, DOS858, DOS860,  
DOS861, DOS862, DOS863, DOS864, DOS865, DOS866, DOS869,  
ISO8859\_1, ISO8859\_13, ISO8859\_2, ISO8859\_3, ISO8859\_4,  
ISO8859\_5, ISO8859\_6, ISO8859\_7, ISO8859\_8, ISO8859\_9,  
KOI8R, KOI8U,  
NEXT,  
TIS620,  
WIN1250, WIN1251, WIN1252, WIN1253, WIN1254, WIN1255, WIN1256,  
WIN1257 and WIN1258.

### *ICU Character Sets*

UTF-8 ibm-1208 ibm-1209 ibm-5304 ibm-5305 windows-65001 cp1208  
UTF-16 ISO-10646-UCS-2 unicode csUnicode ucs-2  
UTF-16BE x-utf-16be ibm-1200 ibm-1201 ibm-5297 ibm-13488  
ibm-17584 windows-1201 cp1200 cp1201 UTF16\_BigEndian  
UTF-16LE x-utf-16le ibm-1202 ibm-13490 ibm-17586  
UTF16\_LittleEndian windows-1200  
UTF-32 ISO-10646-UCS-4 csUCS4 ucs-4  
UTF-32BE UTF32\_BigEndian ibm-1232 ibm-1233  
UTF-32LE UTF32\_LittleEndian ibm-1234  
UTF16\_PlatformEndian  
UTF16\_OppositeEndian  
UTF32\_PlatformEndian  
UTF32\_OppositeEndian  
UTF-7 windows-65000  
IMAP-mailbox-name  
SCSU  
BOCU-1 csBOCU-1  
CESU-8  
ISO-8859-1 ibm-819 IBM819 cp819 latin1 8859\_1 csISOLatin1  
iso-ir-100 ISO\_8859-1:1987 ll 819  
US-ASCII ASCII ANSI\_X3.4-1968 ANSI\_X3.4-1986 ISO\_646.irv:1991  
iso\_646.irv:1983 ISO646-US us csASCII iso-ir-6 cp367 ascii7  
646 windows-20127  
ISO\_2022,locale=ja,version=0 ISO-2022-JP csISO2022JP  
ISO\_2022,locale=ja,version=1 ISO-2022-JP-1 JIS JIS\_Encoding  
ISO\_2022,locale=ja,version=2 ISO-2022-JP-2 csISO2022JP2  
ISO\_2022,locale=ja,version=3 JIS7 csJISEncoding  
ISO\_2022,locale=ja,version=4 JIS8  
ISO\_2022,locale=ko,version=0 ISO-2022-KR csISO2022KR  
ISO\_2022,locale=ko,version=1 ibm-25546

---

ISO\_2022,locale=zh,version=0 ISO-2022-CN  
ISO\_2022,locale=zh,version=1 ISO-2022-CN-EXT  
HZ HZ-GB-2312  
ISCII,version=0 x-iscii-de windows-57002 iscii-dev  
ISCII,version=1 x-iscii-be windows-57003 iscii-bng windows-57006  
x-iscii-as  
ISCII,version=2 x-iscii-pa windows-57011 iscii-gur  
ISCII,version=3 x-iscii-gu windows-57010 iscii-guj  
ISCII,version=4 x-iscii-or windows-57007 iscii-ori  
ISCII,version=5 x-iscii-ta windows-57004 iscii-tml  
ISCII,version=6 x-iscii-te windows-57005 iscii-tlg  
ISCII,version=7 x-iscii-ka windows-57008 iscii-knd  
ISCII,version=8 x-iscii-ma windows-57009 iscii-mlm  
gb18030 ibm-1392 windows-54936  
LMBCS-1 lmbcs  
LMBCS-2  
LMBCS-3  
LMBCS-4  
LMBCS-5  
LMBCS-6  
LMBCS-8  
LMBCS-11  
LMBCS-16  
LMBCS-17  
LMBCS-18  
LMBCS-19  
ibm-367\_P100-1995 ibm-367 IBM367  
ibm-912\_P100-1995 ibm-912 iso-8859-2 ISO\_8859-2:1987 latin2  
csISOLatin2 iso-ir-101 l2 8859\_2 cp912 912 windows-28592  
ibm-913\_P100-2000 ibm-913 iso-8859-3 ISO\_8859-3:1988 latin3  
csISOLatin3 iso-ir-109 l3 8859\_3 cp913 913 windows-28593  
ibm-914\_P100-1995 ibm-914 iso-8859-4 latin4 csISOLatin4  
iso-ir-110 ISO\_8859-4:1988 l4 8859\_4 cp914 914 windows-28594  
ibm-915\_P100-1995 ibm-915 iso-8859-5 cyrillic csISOLatinCyrillic  
iso-ir-144 ISO\_8859-5:1988 8859\_5 cp915 915 windows-28595  
ibm-1089\_P100-1995 ibm-1089 iso-8859-6 arabic csISOLatinArabic  
iso-ir-127 ISO\_8859-6:1987 ECMA-114 ASMO-708 8859\_6 cp1089  
1089 windows-28596 ISO-8859-6-I ISO-8859-6-E  
ibm-813\_P100-1995 ibm-813 iso-8859-7 greek greek8 EL0T\_928  
ECMA-118 csISOLatinGreek iso-ir-126 ISO\_8859-7:1987 8859\_7  
cp813 813 windows-28597  
ibm-916\_P100-1995 ibm-916 iso-8859-8 hebrew csISOLatinHebrew  
iso-ir-138 ISO\_8859-8:1988 ISO-8859-8-I ISO-8859-8-E 8859\_8  
cp916 916 windows-28598  
ibm-920\_P100-1995 ibm-920 iso-8859-9 latin5 csISOLatin5  
iso-ir-148 ISO\_8859-9:1989 l5 8859\_9 cp920 920 windows-28599  
ECMA-128  
ibm-921\_P100-1995 ibm-921 iso-8859-13 8859\_13 cp921 921  
ibm-923\_P100-1998 ibm-923 iso-8859-15 Latin-9 l9 8859\_15 latin0  
csisolatin0 csisolatin9 iso8859\_15\_fdis cp923 923 windows-28605  
ibm-942\_P12A-1999 ibm-942 ibm-932 cp932 shift\_jis78 sjis78  
ibm-942\_VSUB\_VPUA ibm-932\_VSUB\_VPUA  
ibm-943\_P15A-2003 ibm-943 Shift\_JIS MS\_Kanji csShiftJIS  
windows-31j csWindows31J x-sjis x-ms-cp932 cp932 windows-932  
cp943c IBM-943C ms932 pck sjis ibm-943\_VSUB\_VPUA  
ibm-943\_P130-1999 ibm-943 Shift\_JIS cp943 943 ibm-943\_VASCII\_VSUB\_VPUA  
ibm-33722\_P12A-1999 ibm-33722 ibm-5050 EUC-JP  
Extended\_UNIX\_Code\_Packed\_Format\_for\_Japanese  
csEUCPkdFmtJapanese X-EUC-JP eucjis windows-51932  
ibm-33722\_VPUA IBM-eucJP  
ibm-33722\_P120-1999 ibm-33722 ibm-5050 cp33722 33722

---

ibm-33722\_VASCII\_VPUA  
ibm-954\_P101-2000 ibm-954 EUC-JP  
ibm-1373\_P100-2002 ibm-1373 windows-950  
windows-950-2000 Big5 csBig5 windows-950 x-big5  
ibm-950\_P110-1999 ibm-950 cp950 950  
macos-2566-10.2 Big5-HKSCS big5hk HKSCS-BIG5  
ibm-1375\_P100-2003 ibm-1375 Big5-HKSCS  
ibm-1386\_P100-2002 ibm-1386 cp1386 windows-936 ibm-1386\_VSUB\_VPUA  
windows-936-2000 GBK CP936 MS936 windows-936  
ibm-1383\_P110-1999 ibm-1383 GB2312 csGB2312 EUC-CN ibm-eucCN  
hp15CN cp1383 1383 ibm-1383\_VPUA  
ibm-5478\_P100-1995 ibm-5478 GB\_2312-80 chinese iso-ir-58  
csISO58GB231280 gb2312-1980 GB2312.1980-0  
ibm-964\_P110-1999 ibm-964 EUC-TW ibm-eucTW cns11643 cp964 964  
ibm-964\_VPUA  
ibm-949\_P110-1999 ibm-949 cp949 949 ibm-949\_VASCII\_VSUB\_VPUA  
ibm-949\_P11A-1999 ibm-949 cp949c ibm-949\_VSUB\_VPUA  
ibm-970\_P110-1995 ibm-970 EUC-KR KS\_C\_5601-1987 windows-51949  
csEUCKR ibm-eucKR KSC\_5601 5601 ibm-970\_VPUA  
ibm-971\_P100-1995 ibm-971 ibm-971\_VPUA  
ibm-1363\_P11B-1998 ibm-1363 KS\_C\_5601-1987 KS\_C\_5601-1989 KSC\_5601  
csKSC56011987 korean iso-ir-149 5601 cp1363 ksc windows-949  
ibm-1363\_VSUB\_VPUA  
ibm-1363\_P110-1997 ibm-1363 ibm-1363\_VASCII\_VSUB\_VPUA  
windows-949-2000 windows-949 KS\_C\_5601-1987 KS\_C\_5601-1989  
KSC\_5601 csKSC56011987 korean iso-ir-149 ms949  
ibm-1162\_P100-1999 ibm-1162  
ibm-874\_P100-1995 ibm-874 ibm-9066 cp874 TIS-620 tis620.2533  
eucTH cp9066  
windows-874-2000 TIS-620 windows-874 MS874  
ibm-437\_P100-1995 ibm-437 IBM437 cp437 437 csPC8CodePage437  
windows-437  
ibm-850\_P100-1995 ibm-850 IBM850 cp850 850 csPC850Multilingual  
windows-850  
ibm-851\_P100-1995 ibm-851 IBM851 cp851 851 csPC851  
ibm-852\_P100-1995 ibm-852 IBM852 cp852 852 csPCp852 windows-852  
ibm-855\_P100-1995 ibm-855 IBM855 cp855 855 csIBM855 csPCp855  
ibm-856\_P100-1995 ibm-856 cp856 856  
ibm-857\_P100-1995 ibm-857 IBM857 cp857 857 csIBM857 windows-857  
ibm-858\_P100-1997 ibm-858 IBM00858 CCSID00858 CP00858  
PC-Multilingual-850+euro cp858  
ibm-860\_P100-1995 ibm-860 IBM860 cp860 860 csIBM860  
ibm-861\_P100-1995 ibm-861 IBM861 cp861 861 cp-is csIBM861  
windows-861  
ibm-862\_P100-1995 ibm-862 IBM862 cp862 862 csPC862LatinHebrew  
DOS-862 windows-862  
ibm-863\_P100-1995 ibm-863 IBM863 cp863 863 csIBM863  
ibm-864\_X110-1999 ibm-864 IBM864 cp864 csIBM864  
ibm-865\_P100-1995 ibm-865 IBM865 cp865 865 csIBM865  
ibm-866\_P100-1995 ibm-866 IBM866 cp866 866 csIBM866 windows-866  
ibm-867\_P100-1998 ibm-867 cp867  
ibm-868\_P100-1995 ibm-868 IBM868 CP868 868 csIBM868 cp-ar  
ibm-869\_P100-1995 ibm-869 IBM869 cp869 869 cp-gr csIBM869  
windows-869  
ibm-878\_P100-1996 ibm-878 KOI8-R koi8 csKOI8R cp878  
ibm-901\_P100-1999 ibm-901  
ibm-902\_P100-1999 ibm-902  
ibm-922\_P100-1999 ibm-922 cp922 922  
ibm-4909\_P100-1999 ibm-4909  
ibm-5346\_P100-1998 ibm-5346 windows-1250 cp1250  
ibm-5347\_P100-1998 ibm-5347 windows-1251 cp1251



---

ibm-5348\_P100-1997 ibm-5348 windows-1252 cp1252  
 ibm-5349\_P100-1998 ibm-5349 windows-1253 cp1253  
 ibm-5350\_P100-1998 ibm-5350 windows-1254 cp1254  
 ibm-9447\_P100-2002 ibm-9447 windows-1255 cp1255  
 windows-1256-2000 windows-1256 cp1256  
 ibm-9449\_P100-2002 ibm-9449 windows-1257 cp1257  
 ibm-5354\_P100-1998 ibm-5354 windows-1258 cp1258  
 ibm-1250\_P100-1995 ibm-1250 windows-1250  
 ibm-1251\_P100-1995 ibm-1251 windows-1251  
 ibm-1252\_P100-2000 ibm-1252 windows-1252  
 ibm-1253\_P100-1995 ibm-1253 windows-1253  
 ibm-1254\_P100-1995 ibm-1254 windows-1254  
 ibm-1255\_P100-1995 ibm-1255  
 ibm-5351\_P100-1998 ibm-5351 windows-1255  
 ibm-1256\_P110-1997 ibm-1256  
 ibm-5352\_P100-1998 ibm-5352 windows-1256  
 ibm-1257\_P100-1995 ibm-1257  
 ibm-5353\_P100-1998 ibm-5353 windows-1257  
 ibm-1258\_P100-1997 ibm-1258 windows-1258  
 macos-0\_2-10.2 macintosh mac csMacintosh windows-10000  
 macos-6-10.2 x-mac-greek windows-10006 macgr  
 macos-7\_3-10.2 x-mac-cyrillic windows-10007 maccy  
 macos-29-10.2 x-mac-centraleurroman windows-10029 x-mac-ce macce  
 macos-35-10.2 x-mac-turkish windows-10081 mactr  
 ibm-1051\_P100-1995 ibm-1051 hp-roman8 roman8 r8 csHPRoman8  
 ibm-1276\_P100-1995 ibm-1276 Adobe-Standard-Encoding  
     csAdobeStandardEncoding  
 ibm-1277\_P100-1995 ibm-1277 Adobe-Latin1-Encoding  
 ibm-1006\_P100-1995 ibm-1006 cp1006 1006  
 ibm-1098\_P100-1995 ibm-1098 cp1098 1098  
 ibm-1124\_P100-1996 ibm-1124 cp1124 1124  
 ibm-1125\_P100-1997 ibm-1125 cp1125  
 ibm-1129\_P100-1997 ibm-1129  
 ibm-1131\_P100-1997 ibm-1131 cp1131  
 ibm-1133\_P100-1997 ibm-1133  
 ibm-1381\_P110-1999 ibm-1381 cp1381 1381  
 ibm-37\_P100-1995 ibm-37 IBM037 ibm-037 ebcdic-cp-us ebcdic-cp-ca  
     ebcdic-cp-wt ebcdic-cp-nl csIBM037 cp037 037 cpibm37 cp37  
 ibm-273\_P100-1995 ibm-273 IBM273 CP273 csIBM273 ebcdic-de cpibm273  
     273  
 ibm-277\_P100-1995 ibm-277 IBM277 cp277 EBCDIC-CP-DK EBCDIC-CP-NO  
     csIBM277 ebcdic-dk cpibm277 277  
 ibm-278\_P100-1995 ibm-278 IBM278 cp278 ebcdic-cp-fi ebcdic-cp-se  
     csIBM278 ebcdic-sv cpibm278 278  
 ibm-280\_P100-1995 ibm-280 IBM280 CP280 ebcdic-cp-it csIBM280  
     cpibm280 280  
 ibm-284\_P100-1995 ibm-284 IBM284 CP284 ebcdic-cp-es csIBM284  
     cpibm284 284  
 ibm-285\_P100-1995 ibm-285 IBM285 CP285 ebcdic-cp-gb csIBM285  
     ebcdic-gb cpibm285 285  
 ibm-290\_P100-1995 ibm-290 IBM290 cp290 EBCDIC-JP-kana csIBM290  
 ibm-297\_P100-1995 ibm-297 IBM297 cp297 ebcdic-cp-fr csIBM297  
     cpibm297 297  
 ibm-420\_X120-1999 ibm-420 IBM420 cp420 ebcdic-cp-ar1 csIBM420 420  
 ibm-424\_P100-1995 ibm-424 IBM424 cp424 ebcdic-cp-he csIBM424 424  
 ibm-500\_P100-1995 ibm-500 IBM500 CP500 ebcdic-cp-be csIBM500  
     ebcdic-cp-ch cpibm500 500  
 ibm-803\_P100-1999 ibm-803 cp803  
 ibm-838\_P100-1995 ibm-838 IBM-Thai csIBMThai cp838 838 ibm-9030  
 ibm-870\_P100-1995 ibm-870 IBM870 CP870 ebcdic-cp-roece  
     ebcdic-cp-yu csIBM870

---

---

ibm-871\_P100-1995 ibm-871 IBM871 ebcdic-cp-is csIBM871 CP871  
ebcdic-is cpibm871 871  
ibm-875\_P100-1995 ibm-875 IBM875 cp875 875  
ibm-918\_P100-1995 ibm-918 IBM918 CP918 ebcdic-cp-ar2 csIBM918  
ibm-930\_P120-1999 ibm-930 ibm-5026 cp930 cpibm930 930  
ibm-933\_P110-1995 ibm-933 cp933 cpibm933 933  
ibm-935\_P110-1999 ibm-935 cp935 cpibm935 935  
ibm-937\_P110-1999 ibm-937 cp937 cpibm937 937  
ibm-939\_P120-1999 ibm-939 ibm-931 ibm-5035 cp939 939  
ibm-1025\_P100-1995 ibm-1025 cp1025 1025  
ibm-1026\_P100-1995 ibm-1026 IBM1026 CP1026 csIBM1026 1026  
ibm-1047\_P100-1995 ibm-1047 IBM1047 cpibm1047  
ibm-1097\_P100-1995 ibm-1097 cp1097 1097  
ibm-1112\_P100-1995 ibm-1112 cp1112 1112  
ibm-1122\_P100-1999 ibm-1122 cp1122 1122  
ibm-1123\_P100-1995 ibm-1123 cp1123 1123 cpibm1123  
ibm-1130\_P100-1997 ibm-1130  
ibm-1132\_P100-1998 ibm-1132  
ibm-1140\_P100-1997 ibm-1140 IBM01140 CCSID01140 CP01140 cp1140  
cpibm1140 ebcdic-us-37+euro  
ibm-1141\_P100-1997 ibm-1141 IBM01141 CCSID01141 CP01141 cp1141  
cpibm1141 ebcdic-de-273+euro  
ibm-1142\_P100-1997 ibm-1142 IBM01142 CCSID01142 CP01142 cp1142  
cpibm1142 ebcdic-dk-277+euro ebcdic-no-277+euro  
ibm-1143\_P100-1997 ibm-1143 IBM01143 CCSID01143 CP01143 cp1143  
cpibm1143 ebcdic-fi-278+euro ebcdic-se-278+euro  
ibm-1144\_P100-1997 ibm-1144 IBM01144 CCSID01144 CP01144 cp1144  
cpibm1144 ebcdic-it-280+euro  
ibm-1145\_P100-1997 ibm-1145 IBM01145 CCSID01145 CP01145 cp1145  
cpibm1145 ebcdic-es-284+euro  
ibm-1146\_P100-1997 ibm-1146 IBM01146 CCSID01146 CP01146 cp1146  
cpibm1146 ebcdic-gb-285+euro  
ibm-1147\_P100-1997 ibm-1147 IBM01147 CCSID01147 CP01147 cp1147  
cpibm1147 ebcdic-fr-297+euro  
ibm-1148\_P100-1997 ibm-1148 IBM01148 CCSID01148 CP01148 cp1148  
cpibm1148 ebcdic-international-500+euro  
ibm-1149\_P100-1997 ibm-1149 IBM01149 CCSID01149 CP01149 cp1149  
cpibm1149 ebcdic-is-871+euro  
ibm-1153\_P100-1999 ibm-1153 cpibm1153  
ibm-1154\_P100-1999 ibm-1154 cpibm1154  
ibm-1155\_P100-1999 ibm-1155 cpibm1155  
ibm-1156\_P100-1999 ibm-1156 cpibm1156  
ibm-1157\_P100-1999 ibm-1157 cpibm1157  
ibm-1158\_P100-1999 ibm-1158 cpibm1158  
ibm-1160\_P100-1999 ibm-1160 cpibm1160  
ibm-1164\_P100-1999 ibm-1164 cpibm1164  
ibm-1364\_P110-1997 ibm-1364 cp1364  
ibm-1371\_P100-1999 ibm-1371 cpibm1371  
ibm-1388\_P103-2001 ibm-1388 ibm-9580  
ibm-1390\_P110-2003 ibm-1390 cpibm1390  
ibm-1399\_P110-2003 ibm-1399  
ibm-16684\_P110-2003 ibm-16684  
ibm-4899\_P100-1998 ibm-4899 cpibm4899  
ibm-4971\_P100-1999 ibm-4971 cpibm4971  
ibm-12712\_P100-1998 ibm-12712 cpibm12712 ebcdic-he  
ibm-16804\_X110-1999 ibm-16804 cpibm16804 ebcdic-ar  
ibm-1137\_P100-1999 ibm-1137  
ibm-5123\_P100-1999 ibm-5123  
ibm-8482\_P100-1999 ibm-8482  
ibm-37\_P100-1995,swaplfnl ibm-37-s390 ibm037-s390  
ibm-1047\_P100-1995,swaplfnl ibm-1047-s390

---

ibm-1140\_P100-1997,swaplfnl ibm-1140-s390  
ibm-1142\_P100-1997,swaplfnl ibm-1142-s390  
ibm-1143\_P100-1997,swaplfnl ibm-1143-s390  
ibm-1144\_P100-1997,swaplfnl ibm-1144-s390  
ibm-1145\_P100-1997,swaplfnl ibm-1145-s390  
ibm-1146\_P100-1997,swaplfnl ibm-1146-s390  
ibm-1147\_P100-1997,swaplfnl ibm-1147-s390  
ibm-1148\_P100-1997,swaplfnl ibm-1148-s390  
ibm-1149\_P100-1997,swaplfnl ibm-1149-s390  
ibm-1153\_P100-1999,swaplfnl ibm-1153-s390  
ibm-12712\_P100-1998,swaplfnl ibm-12712-s390  
ibm-16804\_X110-1999,swaplfnl ibm-16804-s390  
ebcdic-xml-us