POSTGRES Version 4.2
Release Notes

1994/03/17

1. Introduction

These are the release notes for version 4.2 of the POSTGRES database system from the University of California at Berkeley. The database system and its installation procedure are covered in detail in the setup document for this release, which can be found in the file `doc/postgres-setup.{cat,ps}`. Here, we cover only the most important differences from release 4.1 and earlier versions of the system.

2. Aim

This is the last release of POSTGRES from the POSTGRES group at Berkeley. This release supports a few more platforms than the previous release, has a few more features, has many bug fixes, and has a bit better documentation. The aim was to increase the reliability as much as possible before closing up shop, while providing a few more features that some of our users needed.

3. Compatibility With Existing 4.1 Databases

There have been some changes to the the system catalogs since POSTGRES Version 4.1. Because of this, if you want to migrate your data from Version 4.1 you must copy all your databases out to flat ASCII files and then reload them into the 4.2 database directory. See the `copy` command in the reference manual for details about how to copy data out of a class into a flat ASCII file. Note that this procedure does not preserve rules or large objects; rules must simply be redefined and large objects copied out separately.

4. New Platforms

This release supports the following platforms in addition to the Sun SparcStation and Ultrix DECstation 3100/5000 ports already supported in the past:

H-P 9000 Series 700/800 (PA-RISC) on HP-UX 9.00 or 9.01
DECstation 3000 (Alpha AXP) on DEC OSF/1 1.3
IBM RS/6000 (POWER) on AIX 3.2.5

5. New Features

This section highlights some of the new features. For complete documentation of these features refer to the POSTGRES User Manual and Reference Manual in the appropriate sections.

5.1. Function Overloading

Function overloading is now allowed. That is, the same function name may be used for functions that take different argument lists. The syntax for `define function` is still the same; however, the `remove function` command has changed slightly. The argument types must now be specified, as in

```
remove function distance(point, point)
```

Note that the restriction that dynamically-loaded C functions must have the same C name and POSTGRES name has not been removed, which limits the usefulness of this feature to the extent that duplicate symbol names cause dynamic loading problems on some systems. The main restriction we know about is that you can't define a dynamically-loaded function with the same name as a function that's already built into POSTGRES under OSF/1. This and most other examples involving duplicate function names should work more-or-less as expected on other operating systems.

5.2. Automatic Constant Coercion

In most cases, constants are cast to the correct type as long as there is no ambiguity and the explicit typecast "::type" should not have to be used quite as often. However, additional typecasts may become necessary in a few cases due to the removal of some undocumented, buggy and inconsistent default typecasting (mostly having to do with `char16` and `text`). In addition, there are a few commands (notably `define function`) where parsing is handled in special (broken) ways and type coercion doesn't work correctly.

5.3. Time Data Type Changes

Times prior to January 1, 1970 are now allowed. In addition, the special abstime values `infinity`, `-infinity`, and `current` have been added. The new built-in function `mktinterval` takes two abstime arguments and returns the `tinterval` with those times as its endpoints.

5.4. Hash Access Method

This release of POSTGRES includes a hashed access method for creating secondary indices. The POSTGRES query optimizer will consider using a hashed index whenever an indexed attribute is involved in an equality comparison.

The operator classes defined on hash indices are:

```
int2_ops        float8_ops      char2_ops       char16_ops
int4_ops        oid_ops         char4_ops       text_ops.
float4_ops      char_ops        char8_ops
```

The syntax is similar to that for creating a B-tree or R-tree index. For example, the following POSTQUEL command defines an index on the class `EMP` using the salary attribute:

```
define index EMP_SALARIES on emp using hash (salary int4_ops)
```

Hashing of large objects is not supported.

5.5. External Large Objects

A user may now access files external to the database by the same mechanisms employed for private large objects. Application writers using the LIBPQ interface can now issue a call

```
fd = p_create(pathname, mode, External)
```

to "import" an external file into the database frame of reference. The other calls, such as `p_open`, `p_read` and `p_write`, continue to work exactly as before on "Unix" or "Inversion" files but will now work also on other files imported in this fashion. External files may be accessed according to common permissions of both the user for whom the backend is running and the backend's user id itself. Plainly, if you do not grant the postgres uid access to your files, you cannot `p_open` or `p_creat` them.

Importing a file will automatically create shadow directory entries in the Inversion file system, and the existence of an Inversion file (as regular file) which has the same pathname as an intervening host system directory will abort the inclusion. However, doing a `p_unlink` operation (or the `prm` command) will not actually remove the host system file; it will only make the database forget about its existence.

We have made this extension configurable by the compile time option `-DEXTERNAL_LO` because it represents some security risk. If an application connects to a backend on a remote system masquerading as a user on that system, the application gains access to files according to the constraints mentioned above. Malicious applications could clobber database system catalogs, or obtain copies of the password file for cracking purposes, for example.

An even greater level of risk is exposed if your computer is set up in such a way that allows outside users to place a file where the POSTGRES backend could load it as a user-defined function, as the user-defined function could issue direct calls to `read` and `write` and circumvent even the additional checks imposed by the external large object system. Users who have no account on your machine may be able to place files in your file system using any number of mechanisms, including anonymous FTP and possibly `automount` and `alex`.

POSTGRES should always be run with the Kerberos network authentication system turned on wherever external large objects are to be accessed or user-defined functions may be loaded.

5.6. Array Changes

This version provides a flexible support for multi-dimensional arrays of any base type. It supports two kinds of arrays: “large object arrays” that are stored as postgres large objects and “small arrays” that are stored in the same page as the tuple. The inclusion of large object arrays means that arrays can span multiple pages. The implementation allows update and retrieval of single array elements or sub parts of the array. See the reference manual under `create`, `append`, `retrieve` and `replace` for details.

For optimizing the retrieval of large arrays we provide optional support for chunking the array as described in the paper [SARA94]. A copy of the paper is available with the source in `src/doc/papers/arrays/paper.ps`.

[SARA94]
S. Sarawagi and M. Stonebraker, “Efficient Organization of Large Multidimensional arrays,” *Proc. Tenth Int. Conference on Data Engineering*, Feb. 1994.

5.7. Asynchronous Portals

This release has support for asynchronous portals. See the section in the reference manual under LIBPQ entitled “ASYNCHRONOUS PORTALS AND NOTIFICATION”.

5.8. Dynamic Loader Changes

The user defined function facility has been changed to use the common `dlopen` interface available on most systems to support shared libraries. For the Ultrix platform (which does not have a `dlopen` interface), a version has been provided in `src/tools/libdl`. See the installation notes for details.

5.9. Error Message Cleanups

An attempt was made to change some of the more arcane error messages into something more useful.

6. Documentation

An attempt was made to document how major subsystems in POSTGRES work. This will be of some help to people attempting to modify (or debug) parts of POSTGRES internals. The document is available in `src/doc/implementation`. We cannot guarantee that this document is either complete or entirely accurate (as the code may have changed even since the writing of the doc) but it should nevertheless be very useful to someone attempting to understand the code. The document on the access method interface (`am.me`) is particularly useful.

In addition to the internals documentation, a pass was made over the User Manual and Reference Manual to clean things up. The User Manual is now more than twice as long and contains important information about creation of dynamically-loaded object code. We have actually tried all of the POSTQUEL queries.

7. Bug Fixes and Overdue Improvements

7.1. Portability Improvements

Substantial effort has been invested in fixing bad prototypes and making the code more portable. It should now be easier to build POSTGRES using pedantic compilers such as `gcc` (on CISC platforms, anyway, for alignment reasons) and it is somewhat less dependent on non-POSIX library routines.

7.2. User Programs

icopy

- Defaulted to Unix large objects instead of the documented Inversion large objects.

initdb

- Now runs `vacuum` on the template database by default. This makes queries on the system catalogs much faster.

monitor

- Silently truncated `copy in` and `copy out` lines at 100 characters.

newbki

- Changed the POSTGRES superuser's UID, but did not change the name of the POSTGRES superuser if no user named "postgres" existed.

pcat

- Treated command-line switches as filenames as well as switches.

pmkdir

- Would often fail while executing `p_stat` on files and directories.

postmaster

- Would occasionally leak shared memory segments or exit while attempting to reacquire shared memory after reinitialization. (This appears to be fixed, feedback is desired.)
- Never detached from its original controlling terminal, resulting in error messages being printed to that tty. Now has a command-line option (-S) that causes it to detach and operate silently.

reindexdb

- Now provided to assist in disaster recovery. See the Reference Manual.

vacuum

- Now works. (The POSTQUEL command worked, but the script didn't.)

7.3. Contributed Programs

Most of the stuff that couldn't be compiled anymore (because it required out-of-date external software) or is currently being maintained outside Berkeley has been removed. The most current version of `tkmonitor`, a Tcl/Tk-based replacement for the terminal monitor, is incorporated in the Lassen information retrieval system and can be retrieved as:

```
pub/sequoia/src/lassen.tar.Z
```

from s2k-ftp.CS.Berkeley.EDU (128.32.149.157). The latest version of `pgbrowse`, a Tcl/Tk-based tool for querying a POSTGRES database that contains hooks for user extensions, is located in:

```
pub/pgbrowse
```

on crseo.ucsb.edu (128.111.100.50).

7.4. General Bug Fixes

- A pass was made through the code to find and fix places where `char16` was improperly used. Identifiers that are actually 16 characters long are more likely to work.
- Several serious memory leaks were fixed, increasing the number of queries that can be run in a single transaction.

7.5. Access Methods

B-tree

- Insertion of a new minimal (lowest) value would fail under certain circumstances, causing queries that used the index to return incorrect results. Tuples would be visible using a non-index scan (e.g., one with no qualification) but would not be returned by a query with a qualification that caused an index scan.

R-tree

- Variable-length index keys (e.g., polygons) didn't work at all.
- The calculation for estimating the growth of a bounding box corresponding to the union of two bounding boxes produced garbage values, leading to poor tree layout.
- Use of uninitialized variables caused several problems (e.g., nested-loop joins with an R-tree on the inner relation crashed).

7.6. Query Parser

- Negative `int2` and `float4` constants should now work normally, without being entered like:

  ```
  "-3.14159"::float4
  ```

- Queries that contained multiple commands, such as those specified when defining new rules or POSTQUEL functions, would sometimes have their "from" clauses parsed incorrectly.

7.7. Query Executor

- Now attempts to do a better job of detecting non-functional updates (repeated updates to the same tuple in the same transaction).
- Failed to do adequate pinning of buffers, meaning that buffers still in use could be flushed to disk and reused. This caused corruption of relations. This occurred not only in multiuser operation but also when certain operations were performed on large relations (e.g., mergejoin).
- `replace` queries now acquire write locks before beginning their read operations, reducing concurrency somewhat but preventing many bogus deadlocks.
- Projecting more than one attribute from the result of a POSTQUEL function would crash the system if the attributes were of different types.

7.8. Query Optimizer

- Due to a botch in the system catalogs, the optimizer would never use an index on an `oid` attribute of the inner join relation.
- The optimizer generated invalid query plans if existential qualifications (clauses that are totally disconnected from relations that are actually retrieved in the target list) were joined by an "or" clause. This caused the executor to crash.
- Various optimizer cost functions causes floating point underflow and divide-by-zero, which works better on some machines than others.
- Fixed various memory management bugs in the optimization of expensive functions.
- Archival (time travel) queries on a relation crashed the optimizer if you modified the schema of a relation (e.g., using `addattr`).
- Applying "not" to the result of a function crashed the optimizer.

7.9. System Catalogs

- `pg_class.relnatts` was wrong for some system catalogs.
- `pg_group` is now correctly created at initialization time.
- Many function/operator entries contained invalid argument types. For example, the > operator between `int2` and `int4` was not usable.
- The system thought it could hashjoin using the `!=` operator for the `bool` type, sometimes leading to incorrect results when joining `bool` attributes.

7.10. POSTQUEL Utilities

addattr

- Now accepts a `*` option. Changing only a superclass in a hierarchy without changing the children caused traversals of the inheritance hierarchy to crash.

define aggregate

- Now does substantially more error-checking.
- Current operation is closer to the definition used in 4.0.1. This is not what was originally designed or documented, but (for example) allows the aggregates used in the Sequoia 2000 benchmark to be defined again.

define operator

- Made incorrect system catalog entries for unary operators.

extend index

- Consecutive `extends` didn't work.

load

- No longer requires absolute pathnames.
- Object files that contained symbol names longer than 16 characters would often not be correctly loaded.

purge

- Now does something besides crash. See the Reference Manual.

rename

- Would allow you to rename your classes with the prefix `pg_`, which then prevented other operations (such as destroying them).
- Now accepts a `*` option. Changing only a superclass in a hierarchy without changing the children caused traversals of the inheritance hierarchy to crash.

7.11. Transaction System

- Multiple versions of the same tuple would become visible during an update transaction.
- The following error was caused by a race condition that has been fixed:

```
NOTICE:Jan 20 10:26:35:LockReplace: xid table corrupted
```

- Backends that aborted due to deadlock timeouts would often corrupt shared memory, causing other running backends to crash.
- If one backend sat idle while another backend performed many commands that modified the system catalogs, the mostly-idle backend would (in some circumstances) crash when it tried to begin a new transaction due to a bug in the shared cache invalidation code.

7.12. Storage System

- Race conditions within the buffer manager would cause lost updates.
- On SunOS 4, the backend would crash after about 60 files had been opened due to a miscalculation of the number of available file descriptors.

7.13. Inversion File System/Large Objects

- The current Inversion working directory was not always initialized in the frontend library routines.
- Appending 0-4 bytes using the `p_write` interface failed, sometimes causing crashes.
- Executing `p_stat` on an empty Inversion file caused the system to crash.

7.14. LIBPQ

- No longer calls `exit` for most errors, returning "R" and "E" (or `NULL`, depending on the documented interface) more often.
- Now sets `PQErrorMsg` more consistently.
- In the absence of authentication, now uses `getuid` to figure out who the user is instead of the USER environment variable.
- `PQexec` sometimes returned a pointer to an automatic variable as its return value. This would sometimes cause the frontend application to crash or get garbage.
- The number of open portals now grows dynamically (instead of being hardcoded to a maximum of 10).
- Retrieval of variable-length attributes through binary portals would return objects that were four bytes too large. This was often relatively innocuous unless the user code

estimated values from the size of the variable-length attribute or the out-of-bounds read in the server happened to cause a segmentation violation.

7.15. Abstract Data Types

- Error-checking in input/output routines has been improved in general.
- Insufficient space was allocated for many output routines, causing mysterious errors when out-of-bounds writes occurred.
- The time types were overhauled. For example, time subtraction didn't work.
- Polygons with 0 points were not correctly handled in many routines.

8. Known Bugs and Problems

There are several known bugs that we did not fix in this release. Note that unimplemented features and limitations in existing implementations are, for the most part, documented in the Reference Manual rather than here.

A list of known bugs and suggested work-arounds (as we find them) will be made available for anonymous FTP. This list will be kept in the file

```
pub/postgres/postgres-v4r2/bugs
```

on s2k-ftp.CS.Berkeley.EDU (128.32.149.157).

8.1. User Programs

shmemdoc

- This program has become out of date and should not be used. It is provided in case someone needs the functionality so badly that they are willing to fix it.

8.2. Query Executor

- The current implementation of hash joins in POSTGRES attempts to put the entire hash table in virtual memory. If the hash table is too big to fit into memory the transaction will be aborted. The planner tries to take relation size into account when deciding whether or not to plan a hash join, but it is dependent on the most recent database statistics. If these are out of date POSTGRES might still exhibit this unfriendly behavior. To avoid this problem you should `vacuum` your database after any and all large `append` or `copy` commands.
- If a backend fails while in the course of executing a `retrieve into` query, a spurious file, with the same name as the target class of the `retrieve into`, will be left in the database directory. This file can be safely deleted by the database DBA.

8.3. Crash Recovery

- This isn't really so much a bug as a warning. POSTGRES does not implement standard crash recovery techniques such as write-ahead logging. The POSTGRES storage architecture permits fast crash recovery in the sense that the system can restart instantly without having to go through a lengthy analysis of disk blocks and a log file. However, POSTGRES requires that the data still be readable. This implies that if (1) a bug in POSTGRES scrambles the contents of a relation or (2) a disk block goes bad, there is no way of fixing the relation. You may be able to read some of the contents out or you may not (for example, if one of the shared relations goes bad, the entire database may be unusable). Do frequent offline backups. You are warned.

8.4. Transaction System

- There is a subtle bug relating to cache invalidation that can cause POSTGRES to violate transaction semantics in transactions/queries involving multiple commands. If the backend has a relation open that needs to be invalidated when one command is finished the invalidation message is ignored. Thus the relation descriptor can become out of date, and won't be updated until the next time it is invalidated. At the time of writing we believe that the odds you will ever notice this bug are small.

8.5. Rule System

- As ever, the instance rule system essentially ignores indices, so if you are defining a rule on an indexed attribute, you should use the query rewrite rule system.
- The instance rule system does not handle array references correctly. Don't use arrays in rules.

8.6. Inversion File System/Large Objects

- If a backend fails while it is manipulating large objects, spurious large object files will be left in the database directory. Also, there is no mechanism for getting rid of large objects which are returned by functions but not stored in instances.
- Attempting to overwrite Inversion large objects doesn't work reliably. This is due to a major design flaw and hasn't been fixed. If you need rewritable large objects, you must use Unix large objects.

8.7. LIBPQ

- Again, this is more of a warning than a bug. Several mysterious-looking errors, such as:

  ```
  WARN:Nov  4 14:49:09:init_fcache: Cache lookup
    failed for procedure 0
  ```

  are occasionally caused by a POSTQUEL command returning more return values than the frontend application expects. (In other words, the frontend and backend get out-of-sync in the protocol.) See the LIBPQ section of the Reference Manual under `PQFlushI` for details.

8.8. Abstract Data Types

- While input/output routines have improved error-checking, many numeric operators and functions do not check for underflow, overflow and other exceptions.

9. Machine-Dependent Caveats

9.1. DEC OSF/1

- As previously discussed, you should not define a dynamically-loaded C function with the same name as a function that is already built into the POSTGRES server. Any calls to such a function will actually call the POSTGRES built-in function.

9.2. HP-UX

- We could not test the H-P port on HP-UX 8.07 because all of the machines to which we had access were upgraded to HP-UX 9.0. However, the system did work prior to that point. If you find that some minor changes are required to build the system under 8.07, please let us know and we will archive your patch file on our FTP server.

- The standard HP-UX C compiler does not have the -M (make dependencies) option. If you want dependencies, you will need to install the GNU C compiler, `gcc`. (That is, if you attempt to build dependencies, the `mkdep` script will try to invoke `gcc`).
- HP-UX does not provide a version of the BSD `install` program. We have provided a copy of the `bsdinst` script from the MIT X11R5 distribution, which works acceptably.
- The HP-UX library doesn't handle timezones on dates before Jan. 1, 1970 correctly.
- Alpha-test users have reported that it's not hard to overflow the HP-UX kernel file table on a busy machine. Each POSTGRES server can use up to `NOFILE` (60) file descriptors. If you find a workaround or a kernel parameter that works acceptably for you, again, let us know and we'll pass on the information.

9.3. SunOS 4

- The SunOS library doesn't handle timezones on dates before Jan. 1, 1970 correctly.