POSTGRES Version 4.0.1
Release Notes

1993/10/01

1. Introduction

These are the release notes for version 4.0.1 of the POSTGRES database system from UC 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 ˜postgres/doc/postgres-setup.me. Here, we cover only the most important differences from release 4.0 and earlier versions of the system.

2. Aim

The main focus of this release was an attempt to fix the reported bugs in version 4.0. Because of this, no new functionality has been added to the system. The sole development effort was in fixing bugs reported by users of version 4, and bugs uncovered by a first cut at system validation test suites.

3. Existing 4.0 Databases

There have been no changes to the tuple header structures or the system catalogs since POSTGRES Version 4.0. Because of this you do not have to do the POSTGRES data conversion dance after installing 4.0.1. The entire data directory can be copied from the 4.0 installation to the 4.0.1 installation. Here is a suggested means of conversion (if you have the disk space).

```
Once 4.0.1 is installed in /usr/postgres:

    rm -rf /usr/postgres/data
    cd /usr/postgres4.0
    tar cf - data | (cd /usr/postgres; tar xvf -)
```

If there are any problems, remove /usr/postgres/data, run initdb and fall back to the tried method of copying. **Note that this will only work with Version 4 databases.** Databases from 3.1 (or earlier releases) must still be copied out to unix files, then copied back in once 4.0.1 has been successfully installed.

4. New Debugging Tools

Both of the following utilities are built and installed in $POSTGRESHOME/bin during the standard installation procedure. If you think you are experiencing database corruption, we encourage you to try them out.

4.1. pagedoc

Pagedoc is a postgres relation page doctor. It requires the path name of a heap or an index relation as an argument. It understands postgres page formats and can be used to dump the pages. Pagedoc doesn't know about user data in tuples; it only knows about tuple headers which are uniform across the installation.

```
Usage: pagedoc [-h|b|r] [-d level] filename
```

-h, -b, and -r are for heap, btree, and rtree files, respectively. -d level sets the detail level:

0 - page summaries.
1 - page summaries and line pointer summaries.
2 - all of 1 plus information about each tuple.

-h and -d0 are the defaults.

4.2. shmemdoc

Shmemdoc is a shared memory doctor program. It allows you to view the state of shared memory and semaphores after an abnormal termination (i.e. a backend crash). It assumes that state is static -- that is, no other process should be changing shared memory while shmemdoc is running. In order to use this, you should start the postmaster with the -n option (no reinitialization) in order to avoid reinitializing shared structures after a backend terminates abnormally. For information on the available commands type 'help' at the shmemdoc prompt.

5. Known Bugs

There are a few known bugs that we did not fix in this release.

5.1. Asynchronous Portals

A flaw in the implementation of asynchronous portals was discovered a day or so before shipping 4.0.1. Notifies were handled via signals, and processing them asynchronously can lead to a variety of problems. A cleaner implementation has been conceived, but there was no way to do it for this release. For this reason async portals have been backed out of the system. The cleaner version will be ready by the next release.

5.2. Hash Joins

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/copy commands.

5.3. ISNULL/NOTNULL and Index Scans

Query qualifications based on null detection will cause a crash if there is an index defined on that attribute. One specific example is as follows:

```
create foo (a = int4) \g
define index fooind on foo using btree (a int4_ops) \g
retrieve (foo.all) where foo.a ISNULL \g
```

5.4. Cache Invalidation

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.

5.5. Define Operator

At the very last moment we discovered that you cannot successfully declare the commutator and/or negator of an operator if they are not already defined. This makes it tricky to define an operator class to allow index scans over a user defined data type. You have to do define all the operators in the operator class, and then update the negator and commutator fields by hand (i.e. using the terminal monitor).

5.6. Set Functions and Instance Rules

The instance level rule system requires the ability to convert an arbitrary plan into its string equivalent, as well as the ability to convert the string representation back to a valid plan tree. There were two new plan nodes added to the system for the implementation of postquel functions returning sets of results. The string to plan conversion functions do not know about these nodes and therefore cannot convert them. For this reason you cannot have such functions in the body of an instance level rule. You have to use rewrite rules.

5.7. Indices and the Instance Level Rule System

As ever, the Instance Level Rule System essentially ignores indices, so if you are defining a rule on an indexed attribute, you should use the Query Rewrite rule system.

5.8. Retrieve Into and failed backends

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.

5.9. Large Objects and failed backends

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.

6. Known bugs list

A list of known bugs and suggested work-arounds can be anonymously ftp'ed. This list is kept in the file ˜ftp/pub/postgres-v4r0r1.bugs. We will make every attempt to keep this list up to date.

7. Machine-dependent Problems

7.1. Sparcstations running SunOS 4.0.3

Postgres has been known to crash SunOS 4.0.3 on Sparcstations, due to a SunOS bug in shared memory. It appears to work on SunOS 4.1 and higher, so any reports of crashes on SunOS 4.1 and higher are appreciated.