

Two methods for creating a database:

(1) At the command line

```
createdb -U postgres -D pgdata_local <db_name>
```

- user must have previously been granted permission to create databases
- createdb is a “wrapper” for “psql -d “

The “-D” option creates the database in the PGDATA\_LOCAL partition. Note the absence of a \$ in front of the PGDATA\_LOCAL partition name. The PGDATA\_LOCAL partition is sized at 32 GBytes.

If a database is created without the “-D” option, it will be created in the PGDATA partition which is only .5 GBytes in size. If this partition fills up, the postgres engine will crash!

Beginning with AWIPS OB8.3, the partition name must be specified as lower case (pgdata\_ihfs, pgdata\_local, etc)

At RFCs, the IHFS db resides in the pgdata\_ihfs partition.

(2) Using the psql utility

```
psql db_name1
```

```
CREATE DATABASE db_name2;
```

- open psql utility for db\_name1 and create a new db with name = db\_name2

Get a list of previously created databases with “psql -l”.

### **Renaming a Database**

```
ALTER DATABASE <old_name> RENAME TO <new_name> ;
```

### **Database Privileges**

If user A creates a db, then user B automatically has access to it. Users A and B must be known to postgres through the “createuser” command. This was different in Informix. In Informix, if user A created a db, then user B did NOT have access to it unless granted CONNECT, RESOURCE or DBA privilege by user A.

Max length of database name = 64 char

### **Creating Local Databases at RFCs**

In AWIPS OB6, the IHFS and damcrest databases were created in the PGDATA\_IHFS partition. All local databases at the RFCs should be created in the PGDATA\_LOCAL partition.

The following statement will create a database in the PGDATA\_LOCAL partition:

```
createdb -D pgdata_local dbname
```

Note the absence of a \$ in front of the PGDATA\_LOCAL partition name.

The PGDATA\_LOCAL partition is sized at 32 GBytes.

If a database is created without the “-D” option, it will be created in the PGDATA partition which is only .5 GBytes in size. If this partition fills up, the postgres engine will crash!

Note that in AWIPS OB8.3 and earlier, partition names were specified as lower case.

### **Determining the Size of a Database**

On OB9.2 systems (using postgres Version 8.2.6), the following SQL statement can be used:

```
psql db_name -- db_name can be any database
```

```
SELECT pg_database.datname,  
pg_size_pretty(pg_database_size(pg_database.datname)) AS size  
FROM pg_database;
```

### **Error Attempting to Create a Database**

We have seen some cases where the creation of a new database failed because a postgres job was running which had the system database “template1” open. This database must not be in use by another user when attempting to create a new database. Doing a

```
ps -ef | grep post
```

on the database server machine will show if a job has template1 in use.

### **Migrating Databases to the 8.2.x Server**

One difference between the 7.4.8 server and the new 8.2.x server is the default encoding type for databases. The 7.4.8 server had an encoding type of “SQL\_ASCII” as its default. The 8.2.x server uses “UTF-8” as its default type. For an explanation of these encoding types, see Section 21.2.2 of the postgres 8.2.x documentation.

The default encoding type can be changed in the postgresql.conf file or it can be defined

when postgres is initialized using the initdb command. At OHD, the default value of “UTF-8” is used. At AWIPS sites, the default encoding type is “SQL\_ASCII”.

At OHD, a consequence of the above change in default encoding is that dumping some databases on the 7.4.8 server and then attempting to restore them on the 8.2.x server **MAY RESULT IN LOSS OF DATA RECORDS**. Records in tables such as the Descrip and Observer tables having special characters such as “/” and “#” will not be handled properly when an insert is attempted on the 8.2.x server. This problem will result in all records being lost in these tables.

To get around this problem, the user must explicitly set the encoding type to “SQL\_ASCII” using the “-E” option in the CREATEDB statement when creating the database on the 8.2.x server. For example

```
createdb -U postgres -E SQL_ASCII hd_ob83fwr
```

will create a database which can be loaded using the “psql” command as has been done previously without loss of data.

This problem does not occur at AWIPS sites which use “SQL\_ASCII” as the default encoding. At AWIPS sites, databases created using the “createdb” command do not need to include the “-E” option.

### **Database Name With Upper Case Letters or Mixed Case**

Postgres automatically “folds” the database name to lower-case unless it is surrounded with double quotes (“”) like

```
dropdb -U postgres “DB_Name”
```

### **Checking for Processes Open on a Database**

In window 1:

```
psql template1  
SELECT * FROM pg_stat_database;
```

Result: numbackends = 0 for database hd\_ob82empty

In window 2:

```
psql hd_ob82empty
```

In window 1:

```
SELECT * FROM pg_stat_database;
```

Result: numbackends = 1 for database hd\_ob82empty

Note that the parameter “stats\_start\_collector” must be set to true in the postgresql.conf file for the pg\_stats\_database table to be populated. This is the default value.

### **List top 10 tables, indexes in MBytes**

```
select relname,((relpages*8)/1000) as mb from pg_class order by mb desc limit 10;
```

### **Dropping a Database**

You're about to move a database to archival storage, possibly to /dev/null, but pesky users keep talking to it.

So you try

```
DROP DATABASE foo;
```

but you get

```
ERROR: database "foo" is being accessed by other users  
DETAIL: There are 2 other session(s) using the database.
```

Next, you try killing off all the connections to foo, but those pesky users just keep re-connecting! What's to do? Here's what:

```
UPDATE pg_catalog.pg_database SET datallowconn=false WHERE datname='foo';
```

```
SELECT pg_catalog.pg_terminate_backend(procpid) FROM pg_catalog.pg_stat_activity  
WHERE datname='foo';
```

```
DROP DATABASE foo;
```

All gone!

### **Upgrading**

Two things that frequently bite people during an upgrade:

1. Forgetting to ANALYZE the database after reload. Autovacuum would probably fix that for you eventually, but it's better to just issue one manually.

2. Creating the new database with the wrong locale. I see a bunch of LIKE operators in the query plans you show later, so I'm wondering if you went from C locale to a non-C locale and that defeated LIKE optimizations that used to work.