Forensic Analysis of Corrupted PostgreSQL Databases

When stuff really hits the fan

Presenter
Gregory Stark

Agenda

- Causes of Database Corruption
  How to stay out of trouble
- Symptoms to Watch For
  How to recognize when you're in trouble
- PostgreSQL Data Storage
  Where to find your data
- Examples
  What to do when you're in trouble

Agenda

- Causes of Database Corruption
  - *How to stay out of trouble*
- Symptoms to Watch For
  - *How to recognize when you're in trouble*
- PostgreSQL Data Storage
  - *Where to find your data*
- Examples
  - *What to do when you're in trouble*

Causes of Database Corruption

- Faulty Hardware
- Kernel or System Malfeasance
- Pilot Error
- PostgreSQL Bugs

Faulty Hardware - Bad Memory

Far more common than you might think

A recent paper from Google analyzed statistics for tens of thousands of machines from multiple manufacturers over a 2.5 year period.

- 8% of DIMMS suffered a correctable error
- 25,000-75,000 FIT per MBit
  (5-15 failures per day per Gbyte)
- Annual incidence of **uncorrectable** errors was
  1.3% per machine and 0.22% per DIMM.

Bianca Schroeder et. al., SIGMETRICS/Performance '09 June 15-19, 2009

Kernel or System Malfeasance

- fsync that doesn't sync

```
# hdparm -W 0 /dev/sda

/dev/sda:
 setting drive write-caching to 0 (off)
 write-caching =  0 (off)
```

- fsync which doesn't sync even after write caching is disabled
  NFS, LVM, Raid controllers can defeat fsync.
- Snapshots that aren't consistent across volumes
- Filesystem Bugs

Pilot Error

- Setting `fsync=off` followed by a system crash or power failure
- Setting `full_page_writes=off` (except in special cases e.g. ZFS)
- Taking hot backups without invoking `pg_start_backup()`
- Not waiting for `pg_start_backup()` to finish before beginning backup
- Failing to archive WAL files during the backup
- Recovering onto a machine with a different architecture
- Marking functions with inconsistent results `IMMUTABLE`
- Recovering onto machine with different collation ordering

PostgreSQL Bugs

Always use the most recent bug-fix release for the release you're using!

Just a brief sample of critical bugs fixed in these releases:

- 8.4.1: Fix problem that could make expired rows visible after a crash
- 8.3.8: Force WAL segment switch during pg_start_backup()
  This avoids corner cases that could render a base backup unusable.
- 8.2.10: Recovery failed if the WAL ended partway through a btree split operation
- 8.1.10: Prevent index corruption when a transaction inserts rows and then aborts close to the end of a concurrent VACUUM on the same table

Minor releases do **not** require a dump/reload and do not introduce new features or behaviour. They only fix bugs. They can be installed in minutes by installing new binaries and restarting the database.

Agenda

- Causes of Database Corruption
  How to stay out of trouble
- Symptoms to Watch For
  How to recognize when you're in trouble
- PostgreSQL Data Storage
  Where to find your data
- Examples
  What to do when you're in trouble

Symptoms - *Anything* Can Happen!

- Random Crashes
- Data in Database Silently Modified
- Inconsistent Query Results
- "Can't Happen" Errors

Symptoms - *Anything* Can Happen!

- Random Crashes
- Data in Database Silently Modified
- Inconsistent Query Results
- "Can't Happen" Errors

Symptoms - *Anything* Can Happen!

- Random Crashes
- Data in Database Silently Modified
- Inconsistent Query Results
- "Can't Happen" Errors

Symptoms - *Anything* Can Happen!

- Random Crashes
- Data in Database Silently Modified
- Inconsistent Query Results
- "Can't Happen" Errors

Symptoms - *Anything* Can Happen!

- Random Crashes
- Data in Database Silently Modified
- Inconsistent Query Results
- "Can't Happen" Errors

Symptoms - *"Can't Happen" Errors*

```
ERROR: invalid page header in block 3527 of relation "foo"

ERROR:  could not access status of transaction 3221180546
DETAIL:  could not open file "pg_clog/0BFF": No such file or directory

ERROR:  missing chunk number 0 for toast value 25692661 in
pg_toast_25497233

ERROR:  attempted to delete invisible tuple

ERROR:  could not read block 6 of relation 1663/35078/1761966: read
only 0 of 8192 bytes
```

Agenda

- Causes of Database Corruption
  How to stay out of trouble
- Symptoms to Watch For
  How to recognize when you're in trouble
- PostgreSQL Data Storage
  Where to find your data
- Examples
  What to do when you're in trouble

Postgres Data File Storage

- Data is stored in <PGDATA>/base/<databaseoid>/<relfilenode>

- Postgres page size is 8192 bytes by default

- Tables over 1GB are stored in 1GB files

<relfilenode>	(contains blocks	0 - 131,071)
<relfilenode>.1	(contains blocks	131,072 - 262,143)
<relfilenode>.2	(contains blocks	262,144 - 393,215)

  - etc.

- Pages (both heap and index) start with a page header which is checked when the page is loaded. It does not contain a checksum.

- Pages containing all-zeroes are considered “empty” by Postgres

- Postgres refers to tuple physical location by “ctid” which consists of a page number and a “line pointer” within the page.
  e.g. Tuple with ctid (3,10) is in page #3 and is tuple #10 on the page

Postgres File System Layout

```
testdb=> select oid from pg_database where datname = 'testdb';
  oid
-------
 16384
(1 row)


testdb=> select relfilenode from pg_class where relname = 'test1';
  oid
-------
 16385
(1 row)


$ cd $PGDATA/base/16384
$ ls -l 16385
-rw------- 1 postgres postgres 40960 Oct 16 12:13 16385
```

Postgres Heap Data Page Layout

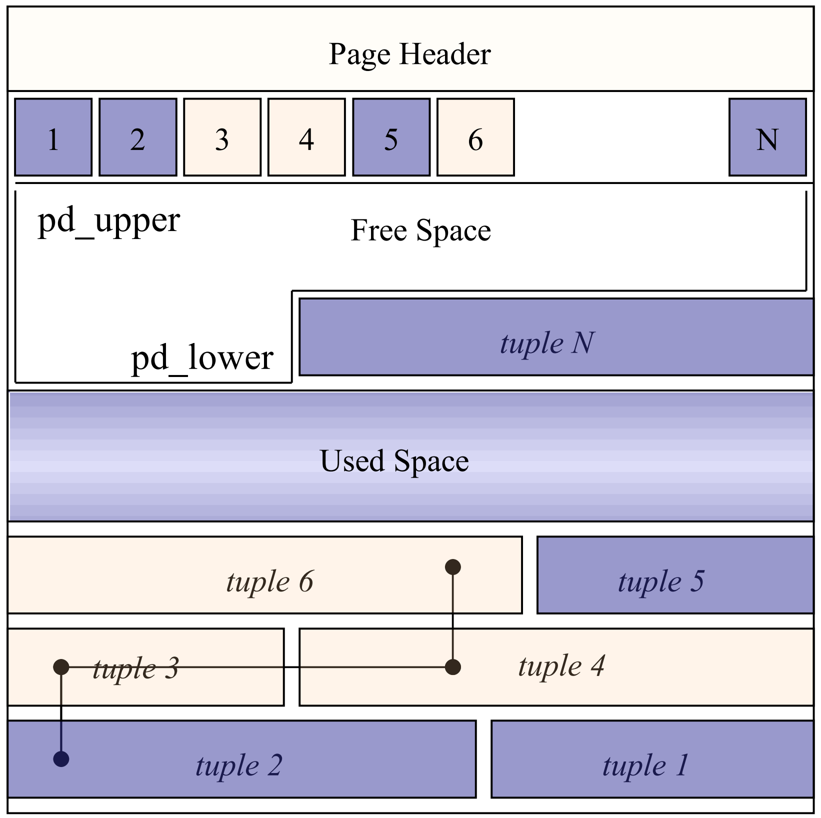

Diagram thanks to Pavan Deolasee ©EnterpriseDB

Page Consists of:

- Page Header
- Line Pointers
- Free Space
- Tuples

Tuples are stored starting from the end of the page moving toward the start.

Separate tuples for each version of row (e.g. Tuples 2,3,4,6 represent a series of updates to the same row)

Agenda

- Causes of Database Corruption
  How to stay out of trouble
- Symptoms to Watch For
  How to recognize when you're in trouble
- PostgreSQL Data Storage
  Where to find your data
- Examples
  What to do when you're in trouble

Example #1 – *Completely corrupt page*

```
testdb=> select count(*) from test1;
 count
-------
   194
(1 row)

$ cd $PGDATA/base/16384

$ dd if=/dev/urandom bs=8192 obs=8192 of=16385 seek=3 count=1
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.0043717 s, 1.9 MB/s

testdb=> select count(*) from test1;
ERROR:  invalid page header in block 3 of relation base/16384/16385

testdb=> set zero_damaged_pages = true;
SET

testdb=> select count(*) from test1;
WARNING:  invalid page header in block 3 of relation base/16384/16385; zeroing out page
 count
-------
   110
(1 row)
```

Example #2 – *Partly corrupt block*

```
$ dd if=/dev/urandom bs=512 obs=512 of=16385 seek=63 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000327559 s, 1.6 MB/s

testdb=> select count(*) from test3;
ERROR:  could not access status of transaction 2341685826
DETAIL:  Could not open file "pg_clog/08B9": No such file or directory.
```

Transaction 2,341,685,826 is not a reasonable transaction id. This is a brand new database. The commit log info for transaction id 2,341,685,826 (8B934A42) would be in pg_clog/08B9 but look at the actual files present in pg_clog for actual recent transactions:

```
$ ls -l $PGDATA/pg_clog
total 8
-rw------- 1 stark eng 8192 Oct 16 19:42 0000
```

Example #2 – *Partly corrupt block*

```
testdb=> \set FETCH_COUNT 1
testdb=> select ctid  from test3;
 ctid
-------
 (0,1)
 (0,2)
 (0,3)
 (0,4)
 (0,5)
 (0,6)
 (0,7)
 (0,8)
 (0,9)
  ...
 (0,38)
 (0,39)
 (1,1)
 (1,2)
 (1,3)
  ...
 (1,36)
 (2,1)
  ...
 (2,38)
ERROR:  could not access status of transaction 2341685826
DETAIL:  Could not open file "pg_clog/08B9": No such file or directory.
```

Advanced Tools: pageinspect

```
testdb=# create table saved_data as select get_raw_page('test3',3) as raw_page;
SELECT

testdb=# \d saved_data
  Table "public.saved_data"
  Column   | Type   | Modifiers
----------+-------+----------
 raw_page | bytea |



testdb=# select * from  heap_page_items(get_raw_page('test3',3));
 lp | lp_off | lp_flags | lp_len |   t_xmin    |   t_xmax    |  t_field3   |        t_ctid        | t_infomask2 | t_infomask | t_hoff | t_bits ...
----+--------+----------+--------+------------+------------+-------------+---------------------+-------------+------------+--------+--------...
  1 |   7724 |        1 |    468 | 3632287242 | 2301944639 | -1953281470 | (3182014523,17515) |        3444 |     -25513 |    183 | 100111 ...
  2 |   7292 |        1 |    432 |        666 |          0 |           0 | (3,2)              |          16 |       2307 |     28 | 110111 ...
  3 |   7132 |        1 |    160 |        666 |          0 |           0 | (3,3)              |          16 |       2307 |     28 | 110110 ...
  4 |   6968 |        1 |    162 |        666 |          0 |           0 | (3,4)              |          16 |       2307 |     28 | 110110 ...
  5 |   6776 |        1 |    191 |        666 |          0 |           0 | (3,5)              |          16 |       2307 |     28 | 111110 ...
  6 |   6580 |        1 |    195 |        666 |          0 |           0 | (3,6)              |          16 |       2307 |     28 | 111110 ...
  7 |   6372 |        1 |    205 |        666 |          0 |           0 | (3,7)              |          16 |       2307 |     28 | 110110 ...
  8 |   6204 |        1 |    167 |        666 |          0 |           0 | (3,8)              |          16 |       2307 |     28 | 110110 ...
  9 |   5936 |        1 |    267 |        666 |          0 |           0 | (3,9)              |          16 |       2307 |     28 | 111111 ...

 ...

 30 |   1548 |        1 |    196 |        666 |          0 |           0 | (3,30)             |          16 |       2307 |     28 | 110110 ...
 31 |   1344 |        1 |    201 |        666 |          0 |           0 | (3,31)             |          16 |       2307 |     28 | 110110 ...
 32 |   1216 |        1 |    126 |        666 |          0 |           0 | (3,32)             |          16 |       2307 |     28 | 110110 ...
 33 |   1056 |        1 |    158 |        666 |          0 |           0 | (3,33)             |          16 |       2307 |     28 | 111110 ...
 34 |    792 |        1 |    262 |        666 |          0 |           0 | (3,34)             |          16 |       2307 |     28 | 110110 ...
 35 |    552 |        1 |    240 |        666 |          0 |           0 | (3,35)             |          16 |       2307 |     28 | 111110 ...
 36 |    388 |        1 |    163 |        666 |          0 |           0 | (3,36)             |          16 |       2307 |     28 | 110110 ...
(36 rows)
```

Extracting Specific Rows Using ctid

```
testdb=> select * from test3 where ctid = '(3,1)';
ERROR:  could not access status of transaction 2341685826
DETAIL:  Could not open file "pg_clog/08B9": No such file or directory.


testdb=> select * from test3 where ctid = '(3,2)';
server closed the connection unexpectedly
         This probably means the server terminated abnormally
         before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.


testdb=> select * from test3 where ctid = '(3,3)';
       name        | setting | unit |        category         |               short_desc          ...
------------------+---------+------+-------------------------+---------------------------------- ...
 log_parser_stats | off     |      | Statistics / Monitoring | Writes parser performance statis ...
(1 row)


testdb=> select * from test3 where ctid = '(3,4)';
       name         | setting | unit |        category         |                short_desc       ...
-------------------+---------+------+-------------------------+--------------------------------- ...
 log_planner_stats | off     |      | Statistics / Monitoring | Writes planner performance stat ...
(1 row)

...
...
```

Manually Zeroing Bad Block

```
testdb=> select oid from pg_database where datname = 'testdb';
  oid
-------
 16384
(1 row)

testdb=> select relfilenode from pg_class where relname = 'test1';
  oid
-------
 16385
(1 row)


LOG:  shutting down
LOG:  database system is shut down

$ dd if=/dev/zero of=/var/tmp/corrupt1/base/16384/16385 bs=8192 seek=3 count=1
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000105741 s, 77.5 MB/s

LOG:  database system was shut down at 2009-10-20 02:07:30 GMT
LOG:  database system is ready to accept connections


testdb=# select count(*) from test3;
 count
-------
   110
```

Thank You

Questions?