Welcome to Write-Ahead Log

Heikki Linnakangas

What is Write-Ahead Log?

EnterpriseDB®
The Enterprise Postgres Company

- Also known as the transaction log or redo log
- Practically every database management system has one
- Also used by journaling file systems, transaction managers etc.

PostgreSQL Write-Ahead-Log

EnterpriseDB®
The Enterprise Postgres Company

- Introduced in version 7.1 by Vadim B. Mikheev
- REDO log only
  - No UNDO log

Write-Ahead Log concepts

EnterpriseDB®
The Enterprise Postgres Company

- A transaction log consists of log records
- One log record for every change
- Each WAL record is assign an LSN (Log Sequence Number)

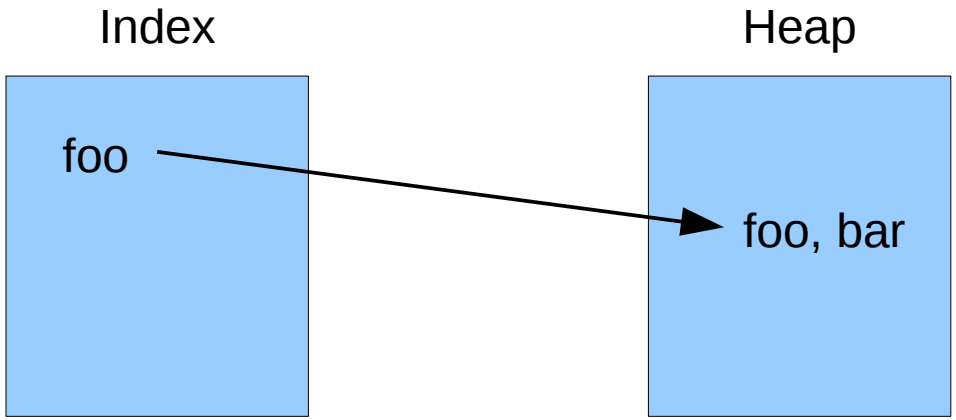
- The log record for an operation is always written to disk before the affected data pages
  - WAL first rule!
- In case of a crash, the WAL is replayed to reconstruct the unsaved changes

Serializing writes

EnterpriseDB®
The Enterprise Postgres Company

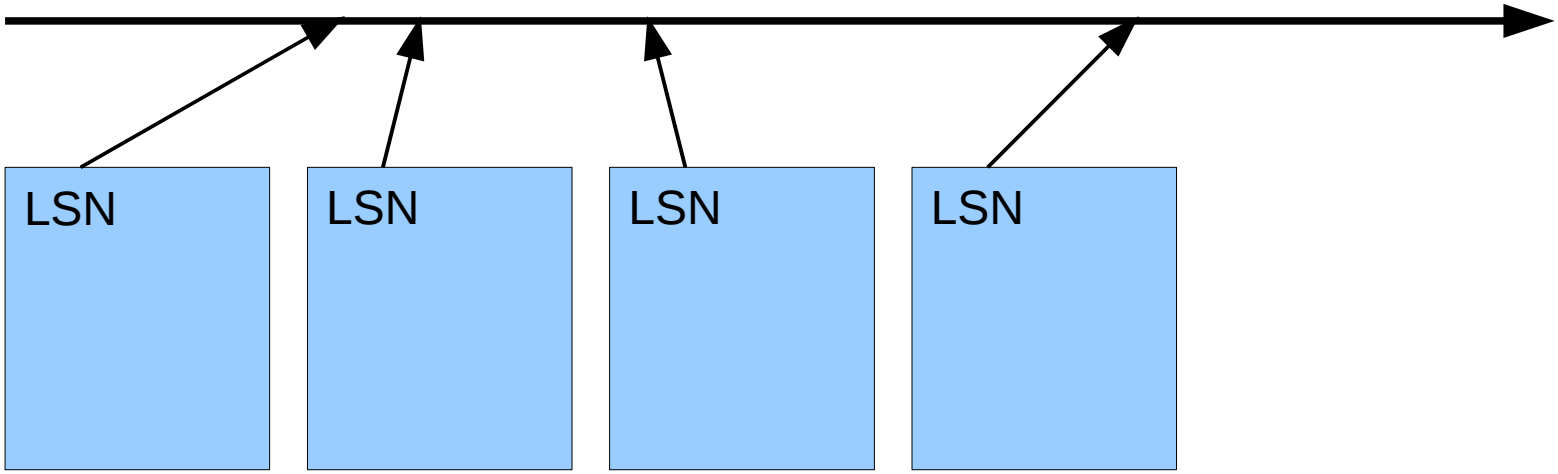
- INSERT INTO table VALUES ('foo', 'bar');

1. Insert to heap
2. Insert to index
3. Commit



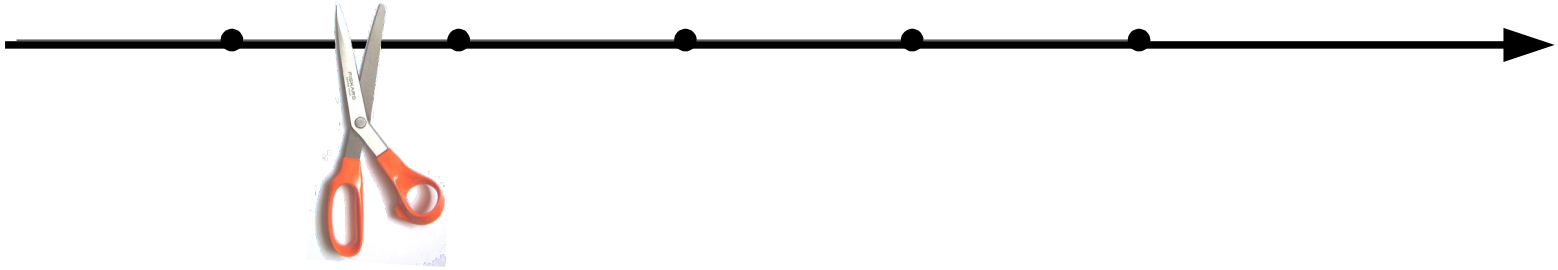
WAL first rule

- In the buffer manager, we refrain from flushing a dirty buffer disk before the WAL record



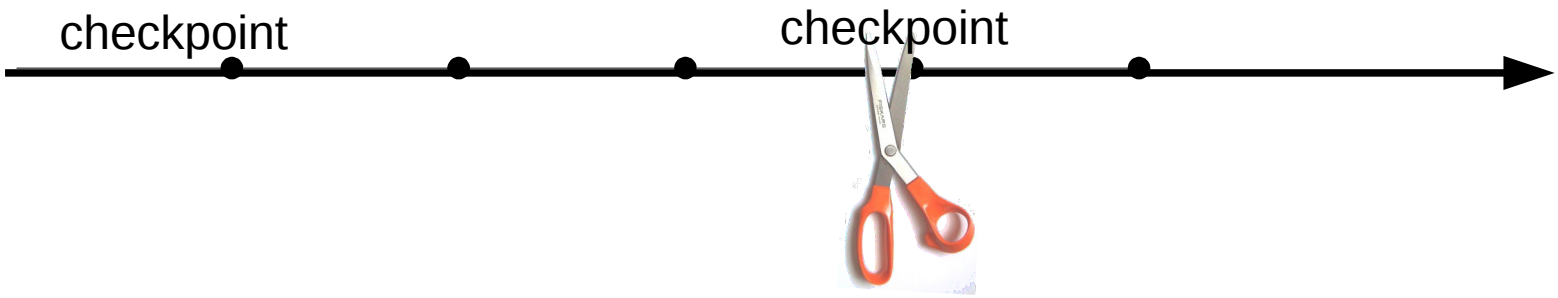
Checkpoints

- WAL grows indefinitely
- Checkpoints allow truncating it



Checkpoints

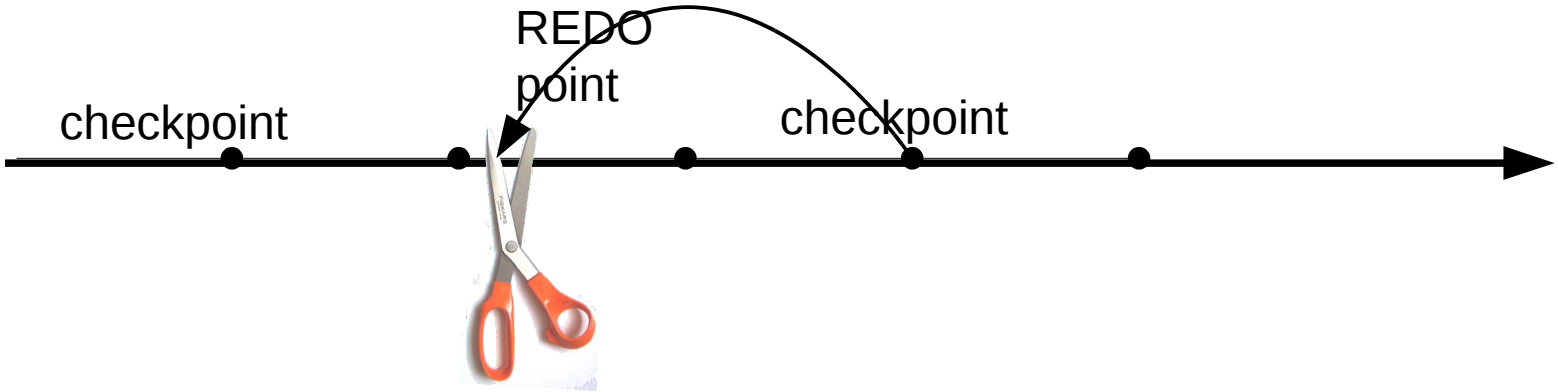
1. Flush all data pages to disk
2. Write a checkpoint record
3. Truncate away old WAL



Online Checkpoints

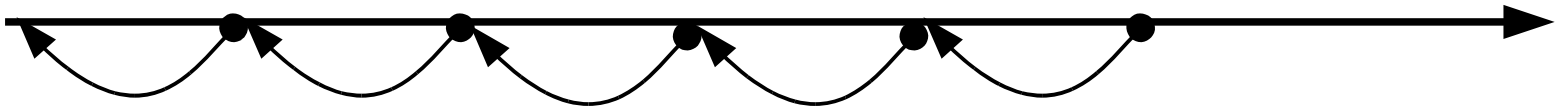
EnterpriseDB®
The Enterprise Postgres Company

1. Establish a REDO point
2. Flush all data pages to disk
3. Write a checkpoint record
4. Truncate away old WAL



Detecting the end of WAL

- Each WAL record has a pointer to the previous record in the chain
- That's how we know where it ends
  - and CRC



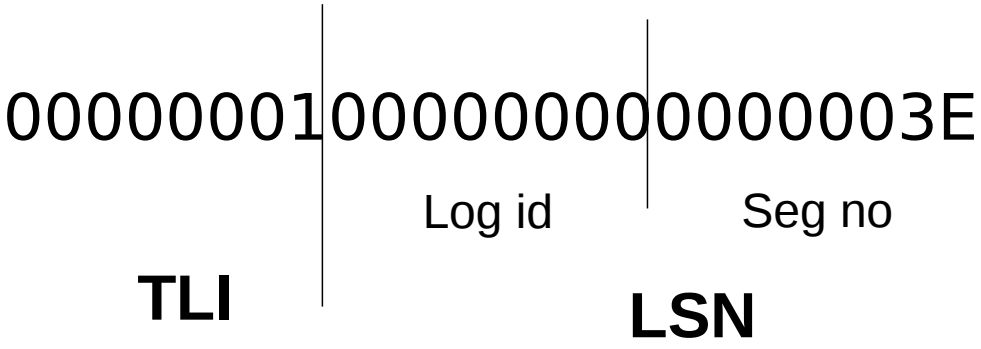
WAL structure

EnterpriseDB®
The Enterprise Postgres Company

- WAL is divided into WAL **segments**
  - Each segment is a file in pg_xlog directory

```
$ ls -l pgsql/data/pg_xlog/
-rw------- 1 heikki heikki      239 2009-11-03 10:39
000000010000000000000034.00000020.backup
-rw------- 1 heikki heikki 16777216 2009-11-03 15:26 00000001000000000000003A
-rw------- 1 heikki heikki 16777216 2009-11-03 15:26 00000001000000000000003B
-rw------- 1 heikki heikki 16777216 2009-11-03 10:39 00000001000000000000003C
-rw------- 1 heikki heikki 16777216 2009-11-03 13:49 00000001000000000000003D
-rw------- 1 heikki heikki 16777216 2009-11-03 15:14 00000001000000000000003E
drwx------ 2 heikki heikki      160 2009-11-03 15:26 archive_status
```

WAL filenames



- Timeline ID is used to distinguish WAL generated before and after a PITR recovery

WAL record format

EnterpriseDB®
The Enterprise Postgres Company

```
typedef struct XLogRecord
{
  pg_crc32      xl_crc;     /* CRC for this record */
  XLogRecPtr    xl_prev;    /* ptr to previous record in log */
  TransactionId xl_xid;     /* xact id */
  uint32        xl_tot_len; /* total len of entire record */
  uint32        xl_len;     /* total len of rmgr data */
  uint8         xl_info;    /* flag bits, see below */
  RmgrId        xl_rmid;    /* resource manager for this record */

/* Depending on MAXALIGN, there are either 2 or 6 wasted bytes here */

/* ACTUAL LOG DATA FOLLOWS AT END OF STRUCT */

} XLogRecord;
```

Tools

EnterpriseDB®
The Enterprise Postgres Company

- Developer tools
  - WAL_DEBUG compile option
  - Xlogdump http://xlogviewer.projects.postgresql.org/
- PITR tools
  - Pglesslog http://pgfoundry.org/projects/pglesslog/
  - Clearxlogtail http://pgfoundry.org/projects/clearxlogtail

wal_debug=on

INSERT INTO foo VALUES (1234, 'foobar');

insert: ts 1663 db 11564 rel 16389 block 0 off 2

header: t_infomask2 2 t_infomask 2050 t_hoff 24

0/004F8E14: prv 0/004F8DD0; xid 655; BTREE info 00 len 30 tot_len 58

insert_leaf: index 1663/11564/16395 tid 1/1

0/004F8E50: prv 0/004F8E14; xid 655; XACT info 00 len 24 tot_len 52

commit: 655 at 1979-10-31 18:02:49 EET

Thank you

EnterpriseDB®
The Enterprise Postgres Company

Questions?

Feedback:
http://2009.pgday.eu/feedback