

# PostgreSQL WAL日志解析与应用

王硕

山东瀚高基础软件股份有限公司

# CONTENTS

**Part 01**      **WAL 日志简介**

---

**Part 02**      **WAL 日志工作原理**

---

**Part 03**      **利用 WAL 日志我们可以做什么？**

---



# Part 01

## WAL 日志简介

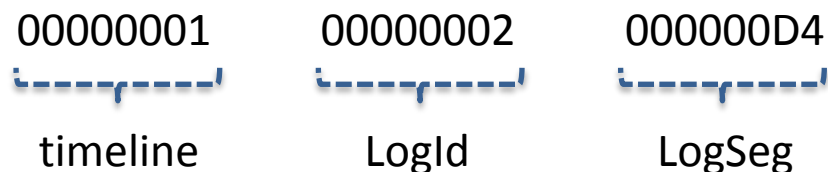
---

[www.highgo.com](http://www.highgo.com)



# Write Ahead Log Files

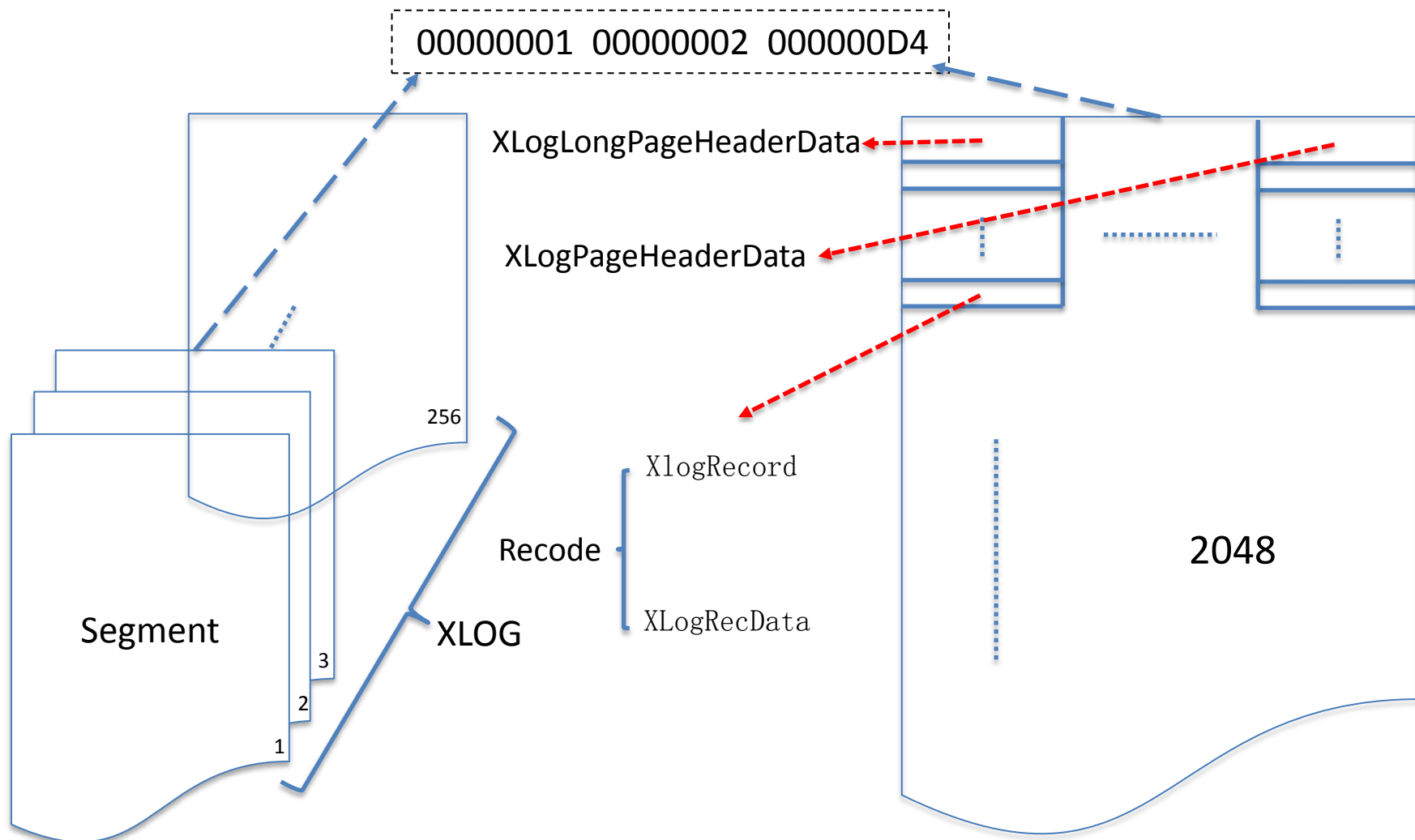
- WAL 日志一般存储在\$PGDATA/pg\_xlog内，他们一般以类似于0000000100000002000000D4的文件存储。



- WAL 日志文件XLOG 文件是一个逻辑概念，每一个XLOG文件，大小为4G(16\*256)，由256个segment组成；
- Segment由2048个Block组成，其大小为16M；
- Block为WAL日志的最小单位，其大小8k，由PageHeaderData、XlogRecord、XLogRecData组成。



# Write Ahead Log Files



# Write Ahead Log Files

```
00a0 0000 068d 0000 c388 0156 0000 0000  
0a00 0000 4cca 61d7 3000 000a 0060 0020  
7f01 0006 d500 0033 0000 0040 0000 0000  
ff00 0003 0000 6000 56bb 0001 0000 2000  
c000 001f 0420 0020 0000 e000 389f c000  
389f 8d00 0006 0000 0000 0000 0000 0000  
0000 0200 0100 0000 1808 cc00 000c 0000  
0000 8c00 0006 0000 0000 0000 0000 0000  
0000 0100 0100 0000 1809 bb00 000b 0000  
0000 0100 0000 1808 cc00 000c 0200 0800  
002e 0000 068d 0000 c3c0 0156 0000 0000  
0180 0000 e1b3 09b4 14ff e9ee ea4a e270  
0001 0001 0000 33d5 0000 067f 0000 0000  
0000 0000 0000 0000 0000 0000 0000 0000
```



XlogRecord



XLogRecData



# Write Ahead Log Files

## Record Data

create table test(id int, name text);

INSERT

```
Insert into test
(id, name)
Values (1, 'jim');
```

```
Header: len, xid, prev,
XLOG_HEAP_INSERT,
rmid, crc
Data:tupleid, infos
Id: 1
Name: jim
```

standby

UPDATE

```
Update test set
name = 'tom'
where id = 1;
```

```
Header: len, xid, prev,
XLOG_HEAP_UPDATE,
rmid, crc
Data:tuples, infos
Id: 1
Name: tom
```

standby

oldtuples

DELETE

```
Delete from
test where id =
1;
```

```
Header: len, xid, prev,
XLOG_HEAP_INSERT,
rmid, crc
Data:tupleid, infos
```

standby

oldtuples



# Part 02

## WAL 日志工作原理

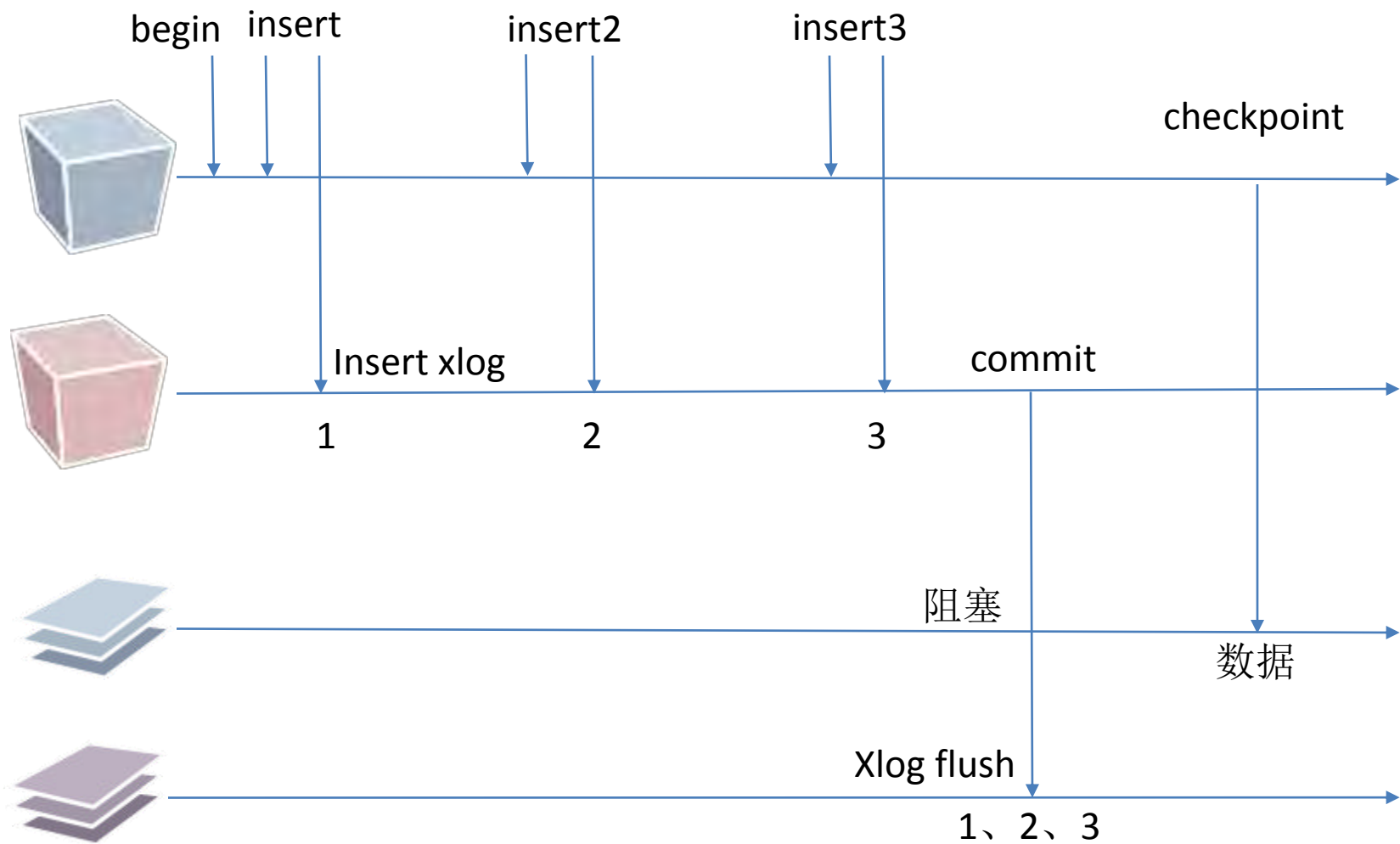
---

[www.highgo.com](http://www.highgo.com)





# Write Ahead Log Files



# ■ Functions of WAL

- **XLogInsert**

插入一条XLOG记录，记录有特殊的RMID标识和附带信息字节，记录的主体包括数据块和数据链。返回XLOG指针（指向记录尾的），可以用做已经存入日志的数据页的LSN，根据LSN，XLOG在修改实际数据之前先写日志（也就是WAL）。

- **XLogFlush**

保证给出位置上的XLOG日志数据被写（flush）到了磁盘上。

- **XLogInsert**

把xlog日志中的内容写出内存



# ■ Locks of WAL

- WALInsertLock

此锁用于将事务日志记录内容插入事务日志内存缓冲区内。首先，加锁，然后将包括整个缓冲区（`full_page_writes`打开状态）的内容复制到日志缓冲区内。

在日志缓冲区刷新期间检查日志缓冲区是否有新内容添加，因为这关系到设置日志缓冲区的刷新点。

确定检查点重做位置在联机备份期间强制执行完整页写入，直到备份完成。

从内置函数获取当前WAL插入位置。



# ■ Locks of WAL

- WALWriteLock

此锁用于将事务日志缓冲区数据写入WAL文件。

在采取这个锁定之后，所有的事务日志缓冲区数据到达预定点将被刷新。

事务日志的刷新可能是由于提交，刷新数据缓冲区，截断提交日志等。

在Xlog的切换期间。

在获取新的Xlog缓冲区期间，如果所有缓冲区已经被占用并且未被刷新。

获取最后一个xlog段切换的时间。



# Part 03

## 利用 WAL 日志我们可以做什么？

---

[www.highgo.com](http://www.highgo.com)



# ■ Write Ahead Log Files

- 基于时间点恢复
- 流复制
- 逻辑复制
- 误操作恢复



## ■ Logminter

- 利用多版本特性，尽可能恢复数据。
- 打开pg\_log，查找误操作信息，进行还原。
- 利用pg\_xlogdump查看wal日志信息，确定时间点，然后恢复。



# Thanks!

## Q & A