Overview

PostgreSQL进阶之路第二章数据文件

https://bbs.cherry-creek.net/thread-34.htm

本章继续通过对照数据库参数和源代码中的数据结构，深入理解各个参数的含义。补充C语言中的基本位运算，进而理解源代码中的向上取整的宏算法。理解数据库从文件到数据块的底层数据存储结构以及其对应的数据结构和指针。理解PG数据库插件的编译安装和使用。最后对PG数据库进行远程性能测试。所有实验基于以下文档配置的虚拟机（https://docs.google.com/document/d/1t1rH2vOpXD7mU_Hbuy-yqBhufN8nd0K9hiS-JdBXKVU/edit?tab=t.0）。

第二章 数据文件

2.1 PostgreSQL 源代码的基础知识

通过find和wc命令行工具获取PG的代码量。PG17.5 包含 1020个头文件，1345个源代码文件，180万行（1804179行）源代码。

```
postgres@pg-50:~/postgresql-17.5/src$ pwd
/home/postgres/postgresql-17.5/src
postgres@pg-50:~/postgresql-17.5/src$ find . -name *.h | wc -l
1020
postgres@pg-50:~/postgresql-17.5/src$ find . -name *.c | wc -l
1345
postgres@pg-50:~/postgresql-17.5/src$ find . -name *.[hc] | xargs wc -l | grep total
 1804179 total
postgres@pg-50:~/postgresql-17.5/src$
```

使用find，xargs，和grep，快速搜索指定的字符串。

```
postgres@pg-50:~/postgresql-17.5/src$ find . -name *.h | xargs grep '#define' | grep BLCKSZ
./include/access/gist_private.h:#define PAGE_IS_EMPTY(nbp) (nbp->freespace == BLCKSZ - BUFFER_PAGE_DATA_OFFSET)
./include/access/slru.h:#define SLRU_MAX_ALLOWED_BUFFERS ((1024 * 1024 * 1024) / BLCKSZ)
./include/access/htup_details.h:#define MaxHeapTupleSize  (BLCKSZ - MAXALIGN(SizeOfPageHeaderData + sizeof(ItemIdData)))
./include/access/hash.h:#define HASH_MAX_BITMAPS                        Min(BLCKSZ / 8, 1024)
./include/pg_config.h:#define BLCKSZ 8192
./include/pg_config.h:#define XLOG_BLCKSZ 8192
./include/storage/large_object.h:#define LOBLKSIZE               (BLCKSZ / 4)
./include/storage/off.h:#define MaxOffsetNumber                 ((OffsetNumber) (BLCKSZ / sizeof(ItemIdData)))
./include/storage/fsm_internals.h:#define NodesPerPage (BLCKSZ - MAXALIGN(SizeOfPageHeaderData) - \
./include/storage/fsm_internals.h:#define NonLeafNodesPerPage (BLCKSZ / 2 - 1)
./include/storage/bufmgr.h:#define DEFAULT_IO_COMBINE_LIMIT Min(MAX_IO_COMBINE_LIMIT, (128 * 1024) / BLCKSZ)
./include/postmaster/walwriter.h:#define DEFAULT_WAL_WRITER_FLUSH_AFTER ((1024 * 1024) / XLOG_BLCKSZ)
postgres@pg-50:~/postgresql-17.5/src$
```

理解基本的数据类型定义。C语言中使用typedef进行数据类型定义。
src/include/c.h
src/include/postgres_ext.h

```c
/*
 * intN
 *      Signed integer, EXACTLY N BITS IN SIZE,
 *      used for numerical computations and the
 *      frontend/backend protocol.
 */
#ifndef HAVE_INT8
typedef signed char int8;       /* == 8 bits */
typedef signed short int16;     /* == 16 bits */
typedef signed int int32;       /* == 32 bits */
#endif                          /* not HAVE_INT8 */

/*
 * uintN
 *      Unsigned integer, EXACTLY N BITS IN SIZE,
 *      used for numerical computations and the
 *      frontend/backend protocol.
 */
#ifndef HAVE_UINT8
typedef unsigned char uint8;    /* == 8 bits */
typedef unsigned short uint16;  /* == 16 bits */
typedef unsigned int uint32;    /* == 32 bits */
#endif                          /* not HAVE_UINT8 */

/*
 * bitsN
 *      Unit of bitwise operation, AT LEAST N BITS IN SIZE.
 */
typedef uint8 bits8;            /* >= 8 bits */
typedef uint16 bits16;          /* >= 16 bits */
typedef uint32 bits32;          /* >= 32 bits */
```

想要理解对齐的宏定语，需要理解C语言中的基本的位运算。C语言中的基本位运算通常都有可替代的数学基本代数运算。在C语言中之所以大量的使用位运算，而不使用基本的数学运算的原因是因为位运算的效率远远高于基本数学代数运算。举一个简单的例子，>> 是右移运算符，x >> 1 表示向右移动一位，位运算的向右移动一位等价于 x / 2，但是前者的计算机实现性能远高于后者。

常见的位运算（C语言）

~：按位取反（NOT）
&：按位与（AND）
|：按位或（OR）
^：按位异或（XOR）
~(a ^ b)：按位同或（XNOR）
<<：左移（Shift Left）
>>：右移（Shift Right）

按位取反的例子和应用场景。
例子：
x = 0b00001111;
result = ~x; // = 0b11110000
应用：

- 构造掩码：~(align - 1) 用于对齐
- 补码运算：-x = ~x + 1

按位与的例子和应用场景。
例子：
x = 0b1101;
y = 0b1011;
result = x & y; // = 0b1001
应用：

- 提取特定位：x & 0x0F 提取低 4 位
- 内存对齐掩码：x & ~(align - 1)
- 权限位检测：判断某个标志是否被置位

按位或的例子和应用场景。
例子：
x = 0b1101;
y = 0b1011;
result = x | y; // = 0b1111
应用：
- 设置标志位：打开某个权限或状态
- 位合并：合并两个二进制值

按位异或的例子和应用场景。
例子：
x = 0b1101;
y = 0b1011;
result = x ^ y; // = 0b0110
应用：
- 反转特定位：配合掩码 x ^ 0x01
- 加密解密：XOR 密码原理
- 无临时变量交换两个数：

按位同或的例子和应用场景。
例子：
x = 0b1101;
y = 0b1011;
result = ~(x ^ y); // = 0b1001 （按位比较相同为1，不同为0，再取反）
应用：
- 判断两个位是否一致（相同）：~(x ^ y) 中每一位为 1，代表 x 和 y 的对应位相同。可以用来批量比对标志位是否一致。
- 数字相似度比较：在图像/音频/数据压缩领域，用于比较两个模式（bit pattern）有多少位相同。
- 模拟数字电路中的“等值门”（XNOR gate）：如果你在做FPGA/CPLD/电路仿真，按位同或就是等值判断的最底层构件。
- 冗余校验/纠错算法（ECC）：某些硬件或协议中，使用 XNOR 来生成一致性检查位。

或，异或，同或的对比对照理解。
逻辑名称	英文	逻辑意义	C 语言中对应的 **位运算符**	示例		

或	OR	有一个为真即可	\`	\`（按位或）	\`a	b\`
异或	XOR	仅一个为真	`^`（按位异或）	`a ^ b`		
同或	XNOR	两个相同为真	`~(a ^ b)`（通过异或+取反实现 ）	`~(a ^ b)`		

左移的例子和应用场景。
例子：
x = 0b00000011; // 3
result = x << 2; // = 0b00001100 = 12
应用：

- 快速乘 2^n
- 设置第 n 位：1 << n
- 构造掩码：0x01 << 3 得到 0b00001000

右移的例子和应用场景。
例子：
x = 0b00001100; // 12
result = x >> 2; // = 0b00000011 = 3
应用：

- 快速除 2^n
- 提取某一位：(x >> n) & 1
- 压缩数据

理解基本宏定义。C语言中使用#define进行宏定义。
src/include/pg_config.h
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/sys/_types/_uintptr_t.h
src/include/c.h
现在你应该能够理解 & ~ 被用来在C语言中进行向上对齐的位操作。

```
/* Define as the maximum alignment requirement of any C data type. */
#define MAXIMUM_ALIGNOF 8
```

```
#ifndef __has_attribute
typedef unsigned long          uintptr_t;
#else
typedef unsigned long          uintptr_t;
#endif /* __has_attribute */
```

```
#define TYPEALIGN(ALIGNVAL,LEN)  \
    (((uintptr_t) (LEN) + ((ALIGNVAL) - 1)) & ~((uintptr_t) ((ALIGNVAL) - 1)))

#define SHORTALIGN(LEN)         TYPEALIGN(ALIGNOF_SHORT, (LEN))
#define INTALIGN(LEN)           TYPEALIGN(ALIGNOF_INT, (LEN))
#define LONGALIGN(LEN)          TYPEALIGN(ALIGNOF_LONG, (LEN))
#define DOUBLEALIGN(LEN)        TYPEALIGN(ALIGNOF_DOUBLE, (LEN))
#define MAXALIGN(LEN)           TYPEALIGN(MAXIMUM_ALIGNOF, (LEN))
```

2.2 数据文件

理解逻辑文件/表/索引，物理文件/段，块/数据页的相对关系。
逻辑文件/表/索引 → 物理文件/段 → 块/数据页
16389（逻辑文件的 oid）→ 16389（逻辑文件的第一个段/物理文件）/ 16389.1（逻辑文件的第二个段/物理文件）→ 块（1）

默认的块大小是8KB。默认段的大小是131072个块，也就是1GB。
src/include/pg_config.h

```
/* Size of a disk block --- this also limits the size of a tuple. You can set
   it bigger if you need bigger tuples (although TOAST should reduce the need
   to have large tuples, since fields can be spread across multiple tuples).
   BLCKSZ must be a power of 2. The maximum possible value of BLCKSZ is
   currently 2^15 (32768). This is determined by the 15-bit widths of the
   lp_off and lp_len fields in ItemIdData (see include/storage/itemid.h).
   Changing BLCKSZ requires an initdb. */
#define BLCKSZ 8192
```

```c
/* RELSEG_SIZE is the maximum number of blocks allowed in one disk file. Thus,
   the maximum size of a single file is RELSEG_SIZE * BLCKSZ; relations bigger
   than that are divided into multiple files. RELSEG_SIZE * BLCKSZ must be
   less than your OS' limit on file size. This is often 2 GB or 4GB in a
   32-bit operating system, unless you have large file support enabled. By
   default, we make the limit 1 GB to avoid any possible integer-overflow
   problems within the OS. A limit smaller than necessary only means we divide
   a large relation into more chunks than necessary, so it seems best to err
   in the direction of a small limit. A power-of-2 value is recommended to
   save a few cycles in md.c, but is not absolutely required. Changing
   RELSEG_SIZE requires an initdb. */
#define RELSEG_SIZE 131072
```

```
willi@macbook-pro-m3 postgresql-17.5 % echo '8192*131072/1024/1024/1024' | bc
1
```

理解块编号。块编号在一个逻辑文件中是统一的和一致的。所以如果使用默认的8KB的块，最大的逻辑文件/表/索引是大小是32TB（2^32 * 8KB）。

```c
/*
 * BlockNumber:
 *
 * each data file (heap or index) is divided into postgres disk blocks
 * (which may be thought of as the unit of i/o -- a postgres buffer
 * contains exactly one disk block).  the blocks are numbered
 * sequentially, 0 to 0xFFFFFFFE.
 *
 * InvalidBlockNumber is the same thing as P_NEW in bufmgr.h.
 *
 * the access methods, the buffer manager and the storage manager are
 * more or less the only pieces of code that should be accessing disk
 * blocks directly.
 */
typedef uint32 BlockNumber;

#define InvalidBlockNumber      ((BlockNumber) 0xFFFFFFFF)

#define MaxBlockNumber          ((BlockNumber) 0xFFFFFFFE)`
```

```
postgres@pg-50:~/postgresql-17.5$ echo "2^32 * 8 / 1024 / 1024 / 1024" | bc
32
```

通过实验理解PG按块（8KG）分配物理文件的可用空间。

```
\c oracle
CREATE TABLE state(id INT, name CHAR(2));
SELECT pg_relation_filepath('state');
\! ls -l $PGDATA/base/16387/16418
INSERT INTO state VALUES(0, 'TX');
checkpoint;
\! ls -l $PGDATA/base/16387/16418
```

```
oracle=# \c oracle
You are now connected to database "oracle" as user "postgres".
oracle=# CREATE TABLE state(id INT, name CHAR(2));
CREATE TABLE
oracle=# SELECT pg_relation_filepath('state');
 pg_relation_filepath
----------------------
 base/16387/16418
(1 row)

oracle=# \! ls -l $PGDATA/base/16387/16418
-rw------- 1 postgres postgres 0 Jun  5 19:55 /home/postgres/data1/base/16387/16418
oracle=# INSERT INTO state VALUES(0, 'TX');
INSERT 0 1
oracle=# checkpoint;
CHECKPOINT
oracle=# \! ls -l $PGDATA/base/16387/16418
-rw------- 1 postgres postgres 8192 Jun  5 19:55 /home/postgres/data1/base/16387/16418
oracle=#
```

数据页的基本结构。这里的结构从左到右，从上到下，从低位地址到高位地址。

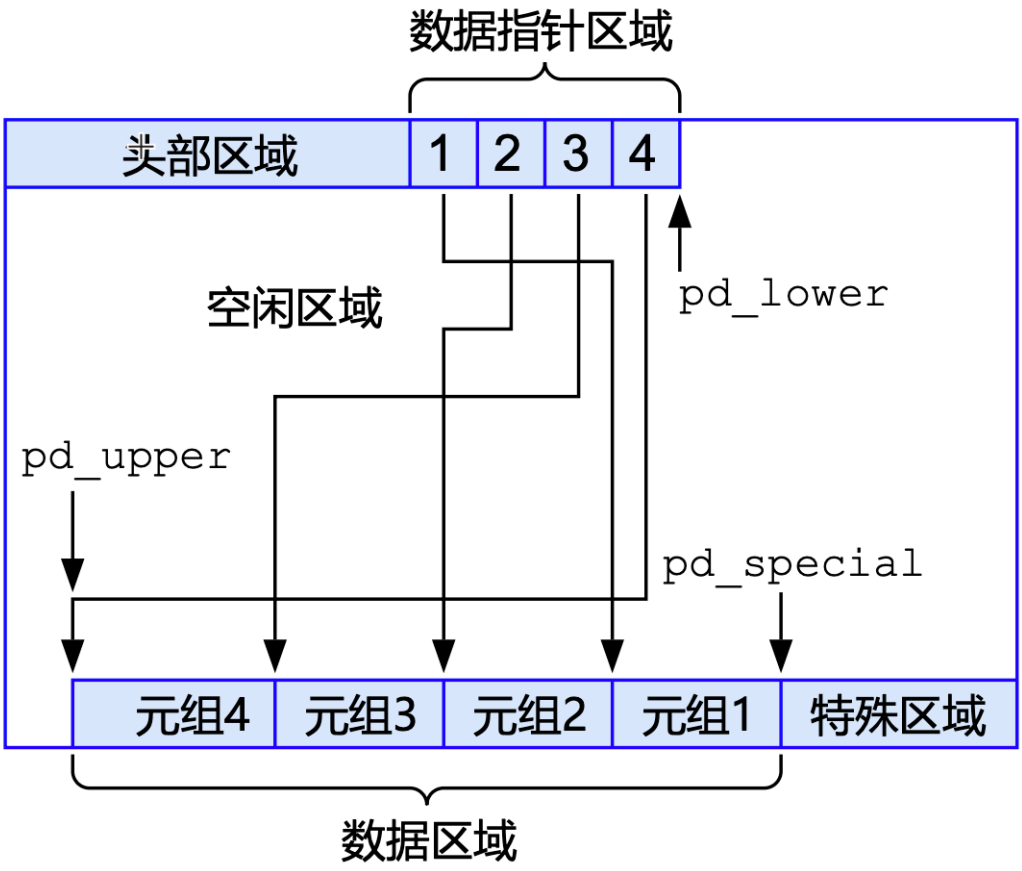


图 2.2: 数据页的基本结构

查看源代码，数据页头的数据结构。使用 hexdump，对比数据结构和实际的十六进制数据块/数据页信息。该数据结构对应上图的头部区域和数据指针区域。
src/include/storage/bufpage.h

```c
typedef struct PageHeaderData
{
    /* XXX LSN is member of *any* block, not only page-organized ones */
    PageXLogRecPtr pd_lsn;      /* LSN: next byte after last byte of xlog
                                 * record for last change to this page */
    uint16      pd_checksum;    /* checksum */
    uint16      pd_flags;       /* flag bits, see below */
    LocationIndex pd_lower;     /* offset to start of free space */
    LocationIndex pd_upper;     /* offset to end of free space */
    LocationIndex pd_special;   /* offset to start of special space */
    uint16      pd_pagesize_version;
    TransactionId pd_prune_xid; /* oldest prunable XID, or zero if none */
    ItemIdData  pd_linp[FLEXIBLE_ARRAY_MEMBER]; /* line pointer array */
} PageHeaderData;

typedef PageHeaderData *PageHeader;
```

```
oracle=# \! ls -l $PGDATA/base/16387/16418
-rw------- 1 postgres postgres 8192 Jun  5 19:55 /home/postgres/data1/base/16387/16418
oracle=# \! hexdump -C $PGDATA/base/16387/16418
00000000  00 00 00 00 28 2c a1 01  00 00 00 00 1c 00 e0 1f  |....(,..........|
00000010  00 20 04 20 00 00 00 00  e0 9f 3e 00 00 00 00 00  |. . ......>.....|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001fe0  f8 02 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001ff0  01 00 02 00 02 08 18 00  00 00 00 00 07 54 58 00  |.............TX.|
00002000
oracle=#
```

查看源代码，记录/行/数据指针区域的元素，所代表的数据结构。
src/include/storage/itemid.h

```c
typedef struct ItemIdData
{
    unsigned    lp_off:15,      /* offset to tuple (from start of page) */
                lp_flags:2,     /* state of line pointer, see below */
                lp_len:15;      /* byte length of tuple */
} ItemIdData;

typedef ItemIdData *ItemId;
```

查询PG已安装的插件。
SELECT oid, extname, extversion FROM pg_extension;

```
postgres=# SELECT oid, extname, extversion FROM pg_extension;
  oid  | extname | extversion
-------+---------+------------
 13615 | plpgsql | 1.0
(1 row)
```

编译新的插件 pageinspect 和 pg_buffercache。
cd /home/postgres/postgresql-17.5/contrib/pageinspect
make
make install
cd /home/postgres/postgresql-17.5/contrib/pg_buffercache
make
make install

```
postgres@pg-50:~/postgresql-17.5/contrib/pageinspect$ pwd
/home/postgres/postgresql-17.5/contrib/pageinspect
postgres@pg-50:~/postgresql-17.5/contrib/pageinspect$ make
make -C ../../src/backend generated-headers
make[1]: Entering directory '/home/postgres/postgresql-17.5/src/backend'
make -C ../include/catalog generated-headers
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/include/catalog'
make[2]: Nothing to be done for 'generated-headers'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/include/catalog'
make -C nodes generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make[2]: Nothing to be done for 'generated-header-symlinks'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make -C utils generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils'
make -C adt jsonpath_gram.h
make[3]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[3]: 'jsonpath_gram.h' is up to date.
make[3]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils'
make[1]: Leaving directory '/home/postgres/postgresql-17.5/src/backend'
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -I. -I. -I../../src/include  -D_GNU_SOUR
CE   -c -o brinfuncs.o brinfuncs.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -I. -I. -I../../src/include  -D_GNU_SOUR
CE   -c -o btreefuncs.o btreefuncs.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -I. -I. -I../../src/include  -D_GNU_SOUR
CE   -c -o fsmfuncs.o fsmfuncs.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -I. -I. -I../../src/include  -D_GNU_SOUR
CE   -c -o ginfuncs.o ginfuncs.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
```

```
postgres@pg-50:~/postgresql-17.5/contrib/pageinspect$ make install
make -C ../../src/backend generated-headers
make[1]: Entering directory '/home/postgres/postgresql-17.5/src/backend'
make -C ../include/catalog generated-headers
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/include/catalog'
make[2]: Nothing to be done for 'generated-headers'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/include/catalog'
make -C nodes generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make[2]: Nothing to be done for 'generated-header-symlinks'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make -C utils generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils'
make -C adt jsonpath_gram.h
make[3]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[3]: 'jsonpath_gram.h' is up to date.
make[3]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils'
make[1]: Leaving directory '/home/postgres/postgresql-17.5/src/backend'
/usr/bin/mkdir -p '/home/postgres/pg17.5/lib'
/usr/bin/mkdir -p '/home/postgres/pg17.5/share/extension'
/usr/bin/mkdir -p '/home/postgres/pg17.5/share/extension'
/usr/bin/install -c -m 755  pageinspect.so '/home/postgres/pg17.5/lib/pageinspect.so'
/usr/bin/install -c -m 644 ./pageinspect.control '/home/postgres/pg17.5/share/extension/'
/usr/bin/install -c -m 644 ./pageinspect--1.11--1.12.sql ./pageinspect--1.10--1.11.sql ./pageinspect--1.9--1.10.sql ./pageinspect--1
.8--1.9.sql ./pageinspect--1.7--1.8.sql ./pageinspect--1.6--1.7.sql ./pageinspect--1.5.sql ./pageinspect--1.5--1.6.sql ./pageinspect
--1.4--1.5.sql ./pageinspect--1.3--1.4.sql ./pageinspect--1.2--1.3.sql ./pageinspect--1.1--1.2.sql ./pageinspect--1.0--1.1.sql  '/ho
me/postgres/pg17.5/share/extension/'
postgres@pg-50:~/postgresql-17.5/contrib/pageinspect$
```

```
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ pwd
/home/postgres/postgresql-17.5/contrib/pg_buffercache
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ make
make -C ../../src/backend generated-headers
make[1]: Entering directory '/home/postgres/postgresql-17.5/src/backend'
make -C ../include/catalog generated-headers
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/include/catalog'
make[2]: Nothing to be done for 'generated-headers'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/include/catalog'
make -C nodes generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make[2]: Nothing to be done for 'generated-header-symlinks'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make -C utils generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils'
make -C adt jsonpath_gram.h
make[3]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[3]: 'jsonpath_gram.h' is up to date.
make[3]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils'
make[1]: Leaving directory '/home/postgres/postgresql-17.5/src/backend'
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -I. -I. -I../../src/include  -D_GNU_SOUR
CE   -c -o pg_buffercache_pages.o pg_buffercache_pages.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-format-attribute -
Wimplicit-fallthrough=3 -Wcast-function-type -Wshadow=compatible-local -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-preci
sion=standard -Wno-format-truncation -Wno-stringop-truncation -O2 -fPIC -fvisibility=hidden -shared -o pg_buffercache.so  pg_bufferc
ache_pages.o -L../../src/port -L../../src/common    -Wl,--as-needed -Wl,-rpath,'/home/postgres/pg17.5/lib',--enable-new-dtags -fvisi
bility=hidden
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$
```

```
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ make install
make -C ../../src/backend generated-headers
make[1]: Entering directory '/home/postgres/postgresql-17.5/src/backend'
make -C ../include/catalog generated-headers
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/include/catalog'
make[2]: Nothing to be done for 'generated-headers'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/include/catalog'
make -C nodes generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make[2]: Nothing to be done for 'generated-header-symlinks'.
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/nodes'
make -C utils generated-header-symlinks
make[2]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils'
make -C adt jsonpath_gram.h
make[3]: Entering directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[3]: 'jsonpath_gram.h' is up to date.
make[3]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils/adt'
make[2]: Leaving directory '/home/postgres/postgresql-17.5/src/backend/utils'
make[1]: Leaving directory '/home/postgres/postgresql-17.5/src/backend'
/usr/bin/mkdir -p '/home/postgres/pg17.5/lib'
/usr/bin/mkdir -p '/home/postgres/pg17.5/share/extension'
/usr/bin/mkdir -p '/home/postgres/pg17.5/share/extension'
/usr/bin/install -c -m 755  pg_buffercache.so '/home/postgres/pg17.5/lib/pg_buffercache.so'
/usr/bin/install -c -m 644 ./pg_buffercache.control '/home/postgres/pg17.5/share/extension/'
/usr/bin/install -c -m 644 ./pg_buffercache--1.2.sql ./pg_buffercache--1.2--1.3.sql ./pg_buffercache--1.1--1.2.sql ./pg_buffercache-
-1.0--1.1.sql ./pg_buffercache--1.3--1.4.sql ./pg_buffercache--1.4--1.5.sql  '/home/postgres/pg17.5/share/extension/'
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$
```

安装新的插件。

psql

CREATE EXTENSION pageinspect;

CREATE EXTENSION pg_buffercache;

SELECT oid, extname, extversion FROM pg_extension;

```
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ psql
psql (17.5)
Type "help" for help.

postgres=# CREATE EXTENSION pageinspect;
CREATE EXTENSION
postgres=# CREATE EXTENSION pg_buffercache;
CREATE EXTENSION
postgres=# SELECT oid, extname, extversion FROM pg_extension;
  oid  |    extname     | extversion
-------+----------------+------------
 13615 | plpgsql        | 1.0
 16421 | pageinspect    | 1.12
 16466 | pg_buffercache | 1.5
(3 rows)

postgres=#
```

查看 pg_buffercache 系统表，包含了共享缓冲区的信息。共享缓冲中包含一个个缓冲的数据页。之所以需要缓冲数据，是因为应用程序从内存中读取数据的效率远远高于从磁盘中读取的效率。

```
postgres=# \d pg_buffercache
                 View "public.pg_buffercache"
      Column      |   Type   | Collation | Nullable | Default
------------------+----------+-----------+----------+---------
 bufferid         | integer  |           |          |
 relfilenode      | oid      |           |          |
 reltablespace    | oid      |           |          |
 reldatabase      | oid      |           |          |
 relforknumber    | smallint |           |          |
 relblocknumber   | bigint   |           |          |
 isdirty          | boolean  |           |          |
 usagecount       | smallint |           |          |
 pinning_backends | integer  |           |          |

postgres=#
```

表结构的数据结构和源代码中的数据结构一一对应。
typedef unsigned int Oid;
typedef uint32 BlockNumber;
typedef Oid RelFileNumber;

查询共享缓存区的大小。共享缓存区缓存的数据页的数量。缓存的数据页的数量的总和的大小等于共享缓存区的大小。
SHOW shared_buffers;
SELECT count(*) FROM pg_buffercache;

```
postgres=# show shared_buffers;
 shared_buffers
----------------
 128MB
(1 row)

postgres=# SELECT count(*) FROM pg_buffercache;
 count
-------
 16384
(1 row)

postgres=# select 16384 * 8 / 1024;
 ?column?
----------
      128
(1 row)

postgres=#
```

定位数据快/数据页的五元组的数据结构。从上到下分别是表空间号，数据库号，关系文件号，文件分支号（衍生类型号，main = 0，fsm = 1，
vm = 2），块号。
src/include/storage/buf_internals.h

```c
typedef struct buftag
{
    Oid         spcOid;         /* tablespace oid */
    Oid         dbOid;          /* database oid */
    RelFileNumber relNumber;    /* relation file number */
    ForkNumber  forkNum;        /* fork number */
    BlockNumber blockNum;       /* blknum relative to begin of reln */
} BufferTag;
```

共享缓存区/共享池中数据页描述数组 (buffer descriptor) 的数据类型。

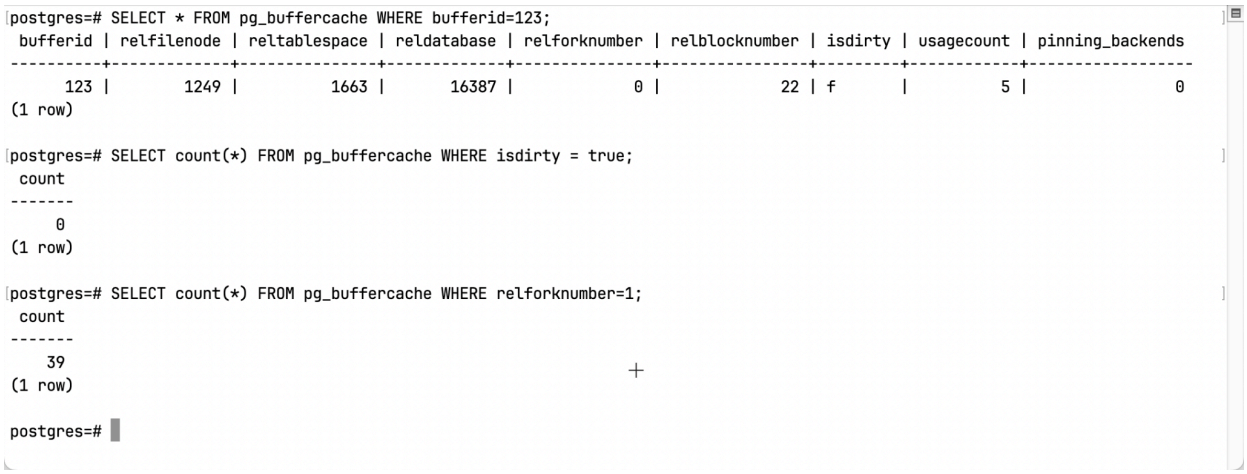
```c
typedef struct BufferDesc
{
    BufferTag   tag;            /* ID of page contained in buffer */
    int         buf_id;         /* buffer's index number (from 0) */

    /* state of the tag, containing flags, refcount and usagecount */
    pg_atomic_uint32 state;

    int         wait_backend_pgprocno;  /* backend of pin-count waiter */
    int         freeNext;       /* link in freelist chain */
    LWLock      content_lock;   /* to lock access to buffer contents */
} BufferDesc;
```

32位的 state的定义，其中第23位表示该数据页是否为脏页。

```c
* Combining these values allows to perform some operations without locking
* the buffer header, by modifying them together with a CAS loop.
*
* The definition of buffer state components is below.
*/
#define BUF_REFCOUNT_ONE 1
#define BUF_REFCOUNT_MASK ((1U << 18) - 1)
#define BUF_USAGECOUNT_MASK 0x003C0000U
#define BUF_USAGECOUNT_ONE (1U << 18)
#define BUF_USAGECOUNT_SHIFT 18
#define BUF_FLAG_MASK 0xFFC00000U

/* Get refcount and usagecount from buffer state */
#define BUF_STATE_GET_REFCOUNT(state) ((state) & BUF_REFCOUNT_MASK)
#define BUF_STATE_GET_USAGECOUNT(state) (((state) & BUF_USAGECOUNT_MASK) >> BUF_USAGECOUNT_SHIFT)

/*
* Flags for buffer descriptors
*
* Note: BM_TAG_VALID essentially means that there is a buffer hashtable
* entry associated with the buffer's tag.
*/
#define BM_LOCKED               (1U << 22)  /* buffer header is locked */
#define BM_DIRTY                (1U << 23)  /* data needs writing */
#define BM_VALID                (1U << 24)  /* data is valid */
#define BM_TAG_VALID            (1U << 25)  /* tag is assigned */
#define BM_IO_IN_PROGRESS       (1U << 26)  /* read or write in progress */
#define BM_IO_ERROR             (1U << 27)  /* previous I/O failed */
#define BM_JUST_DIRTIED         (1U << 28)  /* dirtied since write started */
#define BM_PIN_COUNT_WAITER     (1U << 29)  /* have waiter for sole pin */
#define BM_CHECKPOINT_NEEDED    (1U << 30)  /* must write for checkpoint */
#define BM_PERMANENT            (1U << 31)  /* permanent buffer (not unlogged,
                                             * or init fork) */
/*
```

查询 Shared buffers 中数据页的信息。查询 Shared Buffer池中有多少个脏页。

```
/* 查询编号为123的Page中的有关信息 */
SELECT * FROM pg_buffercache WHERE bufferid=123;
/* 查询Shared Buffer池中有多少个脏页 */
SELECT count(*) FROM pg_buffercache WHERE isdirty = true;
/* 查询Shared Buffer池中有多少个fsm页 */
SELECT count(*) FROM pg_buffercache WHERE relforknumber=1;
```



```
postgres=# SELECT * FROM pg_buffercache WHERE bufferid=123;
 bufferid | relfilenode | reltablespace | reldatabase | relforknumber | relblocknumber | isdirty | usagecount | pinning_backends
----------+-------------+---------------+-------------+---------------+----------------+---------+------------+------------------
      123 |        1249 |          1663 |       16387 |             0 |             22 | f       |          5 |                0
(1 row)

postgres=# SELECT count(*) FROM pg_buffercache WHERE isdirty = true;
 count
-------
     0
(1 row)

postgres=# SELECT count(*) FROM pg_buffercache WHERE relforknumber=1;
 count
-------
    39
(1 row)

postgres=#
```

使用 pageinspect扩展包括的函数 get_raw_page，page_header，heap_page_items 查看数据页信息。

```
/* state里面只有两条记录，且每条记录非常短，故它只有一个数据块，编号为0 */
SELECT * FROM state;
/* 查看0号块的页头信息，请参考PageHeaderData的定义理解之 */
SELECT * FROM page_header(get_raw_page('state',0));
\x
/* 这里展示了每条记录的记录头的信息，请参考HeapTupleHeaderData结构理解之 */
SELECT * FROM heap_page_items(get_raw_page('state',0)) LIMIT 1;
```

```
oracle=# SELECT * FROM state;
 id | name
----+------
  0 | TX
(1 row)

oracle=# SELECT * FROM page_header(get_raw_page('state',0));
    lsn    | checksum | flags | lower | upper | special | pagesize | version | prune_xid
-----------+----------+-------+-------+-------+---------+----------+---------+-----------
 0/1A12C28 |        0 |     0 |    28 |  8160 |    8192 |     8192 |       4 |         0
(1 row)

oracle=# \x
Expanded display is on.
oracle=# SELECT * FROM heap_page_items(get_raw_page('state',0)) LIMIT 1;
-[ RECORD 1 ]----------------
lp          | 1
lp_off      | 8160
lp_flags    | 1
lp_len      | 31
t_xmin      | 760
t_xmax      | 0
t_field3    | 0
t_ctid      | (0,1)
t_infomask2 | 2
t_infomask  | 2306
t_hoff      | 24
t_bits      |
t_oid       |
t_data      | \x00000000075458

oracle#
```

```c
typedef struct PageHeaderData
{
    /* XXX LSN is member of *any* block, not only page-organized ones */
    PageXLogRecPtr pd_lsn;      /* LSN: next byte after last byte of xlog
                                 * record for last change to this page */
    uint16      pd_checksum;    /* checksum */
    uint16      pd_flags;       /* flag bits, see below */
    LocationIndex pd_lower;     /* offset to start of free space */
    LocationIndex pd_upper;     /* offset to end of free space */
    LocationIndex pd_special;   /* offset to start of special space */
    uint16      pd_pagesize_version;
    TransactionId pd_prune_xid; /* oldest prunable XID, or zero if none */
    ItemIdData  pd_linp[FLEXIBLE_ARRAY_MEMBER]; /* line pointer array */
} PageHeaderData;

typedef PageHeaderData *PageHeader;
```

```c
struct HeapTupleHeaderData
{
    union
    {
        HeapTupleFields t_heap;
        DatumTupleFields t_datum;
    }           t_choice;

    ItemPointerData t_ctid;     /* current TID of this or newer tuple (or a
                                 * speculative insertion token) */

    /* Fields below here must match MinimalTupleData! */

#define FIELDNO_HEAPTUPLEHEADERDATA_INFOMASK2 2
    uint16      t_infomask2;    /* number of attributes + various flags */

#define FIELDNO_HEAPTUPLEHEADERDATA_INFOMASK 3
    uint16      t_infomask;     /* various flag bits, see below */

#define FIELDNO_HEAPTUPLEHEADERDATA_HOFF 4
    uint8       t_hoff;         /* sizeof header incl. bitmap, padding */

    /* ^ - 23 bytes - ^ */

#define FIELDNO_HEAPTUPLEHEADERDATA_BITS 5
    bits8       t_bits[FLEXIBLE_ARRAY_MEMBER];  /* bitmap of NULLs */

    /* MORE DATA FOLLOWS AT END OF STRUCT */
};
```

2.3 性能测试工具 pgbench 的使用

修改PG配置文件。允许任意IP远程访问数据库。重启数据库。

```
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ cat $PGDATA/postgresql.conf | grep listen
#listen_addresses = 'localhost'         # what IP address(es) to listen on;
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ vi $PGDATA/postgresql.conf
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ cat $PGDATA/postgresql.conf | grep listen
listen_addresses = '*'          # what IP address(es) to listen on;
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ pg_ctl stop
waiting for server to shut down.... done
server stopped
postgres@pg-50:~/postgresql-17.5/contrib/pg_buffercache$ cd
postgres@pg-50:~$ pg_ctl start -l logfile1
waiting for server to start.... done
server started
postgres@pg-50:~$
```

修改PC客户端认证配置文件。允许指定子网IP通过密码的方式访问数据库。刷新服务器配置。
$PGDATA/pg_hba.conf
UTM subnet connections
host all all 10.0.2.0/24 md5

```
# UTM subnet connections
host    all             all             10.0.2.0/24             md5

"~/data1/pg_hba.conf" 129L, 5805B
```

```
postgres@pg-50:~$ vi $PGDATA/pg_hba.conf
postgres@pg-50:~$ psql
psql (17.5)
Type "help" for help.

postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)

postgres=#
```

修改PG超级用户的密码。
psql -U postgres
\password

```
postgres@pg-50:~$ psql -U postgres
psql (17.5)
Type "help" for help.

postgres=# \password
Enter new password for user "postgres":
Enter it again:
postgres=#
```

是使用子网里的另外一台虚拟机远程登录该PG数据库。远程登录成功。

```
postgres@pg-51:~$ cd
postgres@pg-51:~$ . set.env
postgres@pg-51:~$ hostname
pg-51
postgres@pg-51:~$ ip addr | grep 'inet '
    inet 127.0.0.1/8 scope host lo
    inet 10.0.2.51/24 brd 10.0.2.255 scope global enp0s1
postgres@pg-51:~$ psql -h 10.0.2.50 -U postgres -d oracle
Password for user postgres:
psql (17.5)
Type "help" for help.

oracle=#
```

创建性能测试数据库。

pgbench -d mydb -i -s 1

\c mydb

\dt+

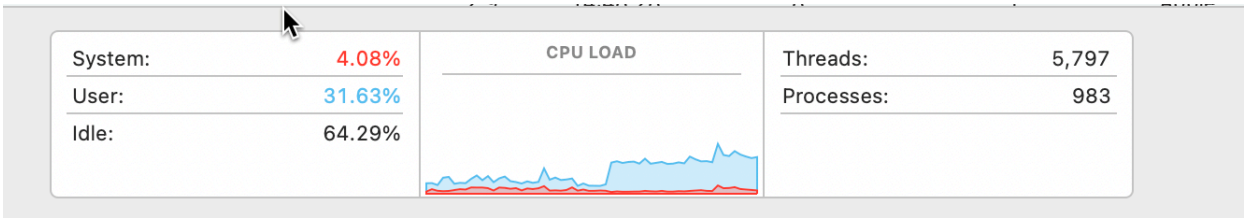
```
postgres@pg-50:~$ pgbench -d mydb -i -s 1
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
vacuuming...
creating primary keys...
done in 1.30 s (drop tables 0.00 s, create tables 0.02 s, client-side generate 0.79 s, vacuum 0.23 s, primary keys 0.25 s).
postgres@pg-50:~$ psql -d mydb
psql (17.5)
Type "help" for help.

mydb=# \dt+
                                    List of relations
 Schema |       Name       | Type  |  Owner   | Persistence | Access method |  Size   | Description
--------+------------------+-------+----------+-------------+---------------+---------+-------------
 public | pgbench_accounts | table | postgres | permanent   | heap          | 13 MB   |
 public | pgbench_branches | table | postgres | permanent   | heap          | 40 kB   |
 public | pgbench_history  | table | postgres | permanent   | heap          | 0 bytes |
 public | pgbench_tellers  | table | postgres | permanent   | heap          | 40 kB   |
(4 rows)

mydb=#
```

增加PG数据库允许的并发连接数，以便接下来的性能测试。
show max_connections;

```
mydb=# show max_connections;
 max_connections
-----------------
 100
(1 row)

mydb=# alter system set max_connections = 500;
ALTER SYSTEM
mydb=# \c
You are now connected to database "mydb" as user "postgres".
mydb=# \q
postgres@pg-50:~$ pg_ctl stop
waiting for server to shut down.... done
server stopped
postgres@pg-50:~$ cd
postgres@pg-50:~$ pg_ctl start -l logfile1
waiting for server to start.... done
server started
postgres@pg-50:~$ psql
psql (17.5)
Type "help" for help.

postgres=# show max_connections;
 max_connections
-----------------
 500
(1 row)

postgres=#
```

使用pgbench性能测试工具，使用另外一台虚拟机对 PG数据库进行性能测试。
pgbench -h 10.0.2.50 -U postgres -d mydb -j 4 -c 400 -T 300

```
postgres@pg-51:~$ pgbench -h 10.0.2.50 -U postgres -d mydb -j 4 -c 400 -T 300
Password:
pgbench (17.5)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 400
number of threads: 4
maximum number of tries: 1
duration: 300 s
number of transactions actually processed: 35300
number of failed transactions: 0 (0.000%)
latency average = 3366.917 ms
initial connection time = 5789.947 ms
tps = 118.803042 (without initial connection time)
postgres@pg-51:~$
postgres@pg-51:~$
postgres@pg-51:~$
postgres@pg-51:~$
```



End