使用pg_resetxlog修复PostgreSQL控制文件的方法

德哥 2016-08-14 9884浏览量

简介： PostgreSQL 控制文件在$PGDATA/global目录下名为pg_control. 控制文件中记录了以下三部分信息 :
1. initdb时生成的静态信息 : pg_control version number: 922 C

PostgreSQL 控制文件在$PGDATA/global目录下名为pg_control.

控制文件中记录了以下三部分信息 :

1. initdb时生成的静态信息 :

```
pg_control version number:            922
Catalog version number:               201204301
Database system identifier:           5831753892046499175
Maximum data alignment:               8
Database block size:                  8192
Blocks per segment of large relation: 131072
WAL block size:                       16384
Bytes per WAL segment:                16777216
Maximum length of identifiers:        64
Maximum columns in an index:          32
Maximum size of a TOAST chunk:        1996
Date/time type storage:               64-bit integers
Float4 argument passing:              by value
Float8 argument passing:              by value
2. postgresql.conf中的配置信息 :
Current wal_level setting:            hot_standby
Current max_connections setting:      1000
Current max_prepared_xacts setting:   10
Current max_locks_per_xact setting:   64
3. write-ahead logging以及checkpoint的动态信息 :
Latest checkpoint location:           96E8/5B000020
Prior checkpoint location:            96E8/5A0C8CC0
Latest checkpoint's REDO location:    96E8/5B000020
Latest checkpoint's TimeLineID:       1
Latest checkpoint's full_page_writes: on
Latest checkpoint's NextXID:          0/1183664222
Latest checkpoint's NextOID:          309701
Latest checkpoint's NextMultiXactId:  1
Lat   t   eckpoint's NextMultiOffset:  0
      kpoint's oldestXID:             1006759584
La      ckpoint's oldestXID's DB:     1
La      ckpoint's oldestActiveXID:    0
Time of latest checkpoint:            Fri 11 Jan 2013 07:44:19 AM CST
Minimum recovery ending location:     0/0
```

```
Minimum recovery ending location:     0/0
Backup start location:                0/0
Backup end location:                  0/0
End-of-backup record required:        no
```

以上信息可以使用pg_controldata从pg_control获取 :

```
src/bin/pg_controldata/pg_controldata.c
 * pg_controldata
 *
 * reads the data from $PGDATA/global/pg_control
```

如果控制文件$PGDATA/global/pg_control损坏或丢失, 数据库将运行异常, 无法启动.

如何修复? 关键在于恢复write-ahead logging以及checkpoint的动态信息.

这些信息可以从pg_xlog, pg_clog, pg_multixact这些目录的文件中解析出来。

pg_xlog的文件名解析可参看, 不同的段大小, 命名大不相同, pg_resetxlog的帮助文件适用16MB的段大小, 如果是其他大小, 需要重新计算名字 :
http://blog.163.com/digoal@126/blog/static/163877040201291411294954 6/

接下来介绍一下使用pg_resetxlog重建pg_control的方法.
pg_resetxlog功能如下 :

```
src/bin/pg_resetxlog/pg_resetxlog.c
 * pg_resetxlog.c
 *        A utility to "zero out" the xlog when it's corrupt beyond recovery.
 *        Can also rebuild pg_control if needed.
 *
 * The theory of operation is fairly simple:
 *        1. Read the existing pg_control (which will include the last
 *               checkpoint record).  If it is an old format then update to
 *               current format.
 *        2. If pg_control is corrupt, attempt to intuit reasonable values,
 *               by scanning the old xlog if necessary.
 *        3. Modify pg_control to reflect a "shutdown" state with a checkpoint
 *               record at the start of xlog.
 *        4. Flush the existing xlog files and write a new segment with
 *               just a checkpoint record in it.  The new segment is positioned
 *               just past the end of the old xlog, so that existing LSNs in
 *               data pages will appear to be "in the past".
 * This is all pretty straightforward except for the intuition part of
 * step 2 ...
```

pg_ 用法 :

```
ocz@db-172-16-3-150-> pg_resetxlog --help
pg_resetxlog resets the PostgreSQL transaction log
```

```
pg_resetxlog resets the PostgreSQL transaction log.

Usage:
  pg_resetxlog [OPTION]... DATADIR

Options:
  -e XIDEPOCH      set next transaction ID epoch
  -f               force update to be done
  -l TLI,FILE,SEG  force minimum WAL starting location for new transaction log
  -m XID           set next multitransaction ID
  -n               no update, just show extracted control values (for testing)
  -o OID           set next OID
  -O OFFSET        set next multitransaction offset
  -V, --version    output version information, then exit
  -x XID           set next transaction ID
  -?, --help       show this help, then exit
```

参数具体含义：

```
-l timelineid,fileid,seg
  The WAL starting address (-l) should be larger than any WAL segment file name currently e
directory pg_xlog under the data directory. These names are also in hexadecimal and have th
first part is the “timeline ID” and should usually be kept the same. Do not choose a value
(0xFF) for the third part; instead increment the second part and reset the third part to 0.
000000010000003200000004A is the largest entry in pg_xlog, -l 0x1,0x32,0x4B will work; but i
entry is 000000010000003A000000FF, choose -l 0x1,0x3B,0x0 or more.
    Note
    pg_resetxlog itself looks at the files in pg_xlog and chooses a default -l setting beyo
    existing file name. Therefore, manual adjustment of -l should only be needed if you are
    segment files that are not currently present in pg_xlog, such as entries in an offline
    the contents of pg_xlog have been lost entirely.

-e XIDEPOCH
  The transaction ID epoch is not actually stored anywhere in the database except in the fi
pg_resetxlog, so any value will work so far as the database itself is concerned. You might
this value to ensure that replication systems such as Slony-I work correctly - if so, an ap
should be obtainable from the state of the downstream replicated database.

-x XID
  A safe value for the next transaction ID (-x) can be determined by looking for the numeri
name in the directory pg_clog under the data directory, adding one, and then multiplying by
that the file names are in hexadecimal. It is usually easiest to specify the option value i
too. For example, if 0011 is the largest entry in pg_clog, -x 0x1200000 will work (five tra
provide the proper multiplier).

-m XID
  A safe value for the next multitransaction ID (-m) can be determined by looking for the n
file name in the directory pg_multixact/offsets under the data directory, adding one, and t
by 65536. As above, the file names are in hexadecimal, so the easiest way to do this is to
c      e in hexadecimal and add four zeroes.

-O
   A safe value for the next multitransaction offset (-O) can be determined by looking for
```

```
  largest file name in the directory pg_multixact/members under the data directory, adding on
  multiplying by 65536. As above, the file names are in hexadecimal, so the easiest way to do
  specify the option value in hexadecimal and add four zeroes.

  -o OID
    There is no comparably easy way to determine a next OID that's beyond the largest one in
  fortunately it is not critical to get the next-OID setting right.

  -n
    no update, just show extracted control values (for testing)
  -f
    force
```

测试步骤如下(基于PostgreSQL 9.2.1) :

1. 新建测试数据, 用到with oids的表, 因为OID无法确定, 看看是否会有异常.
2. 关闭数据库
3. 记下pg_controldata信息, 方便修复后进行比对
4. 删除$PGDATA/global/pg_control
5. 开启数据库观察报错输出
6. touch $PGDATA/global/pg_control
7. 使用pg_resetxlog修复pg_control
8. 记下pg_controldata信息, 与前面的pg_controldata输出进行比对
9. 启动数据库
10. 查看测试数据是否正常, 新插入数据
11. 关闭数据库, 并记下pg_controldata的信息, 看看有何变化.

    测试过程 :

1. 测试数据

```
digoal=> create table oid_test(id int primary key) with oids;
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "oid_test_pkey" for table "o
CREATE TABLE
digoal=> insert into oid_test select generate_series(1,100000);
INSERT 0 100000
digoal=> select min(oid),max(oid) from oid_test ;
  min   |  max
--------+--------
 16397 | 116396
(1 row)
```

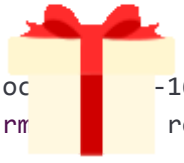
2. 关闭数据

```
ocz@db-172-16-3-150-> pg_ctl stop -m fast
waiting for server to shut down.... done
server stopped
```

3. 记下pg_controldata信息, 方便修复后进行比对

```
ocz@db-172-16-3-150-> pg_controldata
pg_control version number:            922
Catalog version number:               201204301
Database system identifier:           5832000131111550393
Database cluster state:               shut down
pg_control last modified:             Fri 11 Jan 2013 09:48:18 AM CST
Latest checkpoint location:           96E8/5F000020
Prior checkpoint location:            96E8/5EE5C698
Latest checkpoint's REDO location:    96E8/5F000020
Latest checkpoint's TimeLineID:       1
Latest checkpoint's full_page_writes: on
Latest checkpoint's NextXID:          0/1183842312
Latest checkpoint's NextOID:          116414
Latest checkpoint's NextMultiXactId:  65536
Latest checkpoint's NextMultiOffset:  65536
Latest checkpoint's oldestXID:        1006759584
Latest checkpoint's oldestXID's DB:   1
Latest checkpoint's oldestActiveXID:  0
Time of latest checkpoint:            Fri 11 Jan 2013 09:48:18 AM CST
Minimum recovery ending location:     0/0
Backup start location:                0/0
Backup end location:                  0/0
End-of-backup record required:        no
Current wal_level setting:            hot_standby
Current max_connections setting:      1000
Current max_prepared_xacts setting:   10
Current max_locks_per_xact setting:   64
Maximum data alignment:               8
Database block size:                  8192
Blocks per segment of large relation: 131072
WAL block size:                       16384
Bytes per WAL segment:                16777216
Maximum length of identifiers:        64
Maximum columns in an index:          32
Maximum size of a TOAST chunk:        1996
Date/time type storage:               64-bit integers
Float4 argument passing:              by value
Float8 argument passing:              by value
```

4. 删除$PGDATA/global/pg_control



```
oc        -16-3-150-> rm $PGDATA/global/pg_control
rm        regular file `/data05/ocz/pg_root/global/pg_control'? y
```

5. 开启数据库观察报错输出

```
ocz@db-172-16-3-150-> pg_ctl start
server starting
ocz@db-172-16-3-150-> postgres: could not find the database system
Expected to find it in the directory "/data05/ocz/pg_root",
but could not open file "/data05/ocz/pg_root/global/pg_control": No such file or directory
```

接下来进行修复 :

6. touch $PGDATA/global/pg_control

```
ocz@db-172-16-3-150-> touch $PGDATA/global/pg_control
ocz@db-172-16-3-150-> chmod 600 $PGDATA/global/pg_control
```

7. 使用pg_resetxlog修复pg_control

首先确定-l timelineid,fileid,seg的信息 :

```
ocz@db-172-16-3-150-> cd $PGDATA/pg_xlog
ocz@db-172-16-3-150-> ll
total 65M
-rw------- 1 ocz ocz 16M Jan 11 09:39 00000001000096E80000005C
-rw------- 1 ocz ocz 16M Jan 11 09:39 00000001000096E80000005D
-rw------- 1 ocz ocz 16M Jan 11 09:48 00000001000096E80000005E
-rw------- 1 ocz ocz 16M Jan 11 09:48 00000001000096E80000005F
drwx------ 2 ocz ocz 44K Jan 11 09:48 archive_status
```

-l timelineid,fileid,seg 的数据来自pg_xlog文件名的三个部分, 分别占用8个16进制位.

段大小为16MB, 所以末端最大为0xFF.

得出-l 0x1,0x96E8,0x60

接下来确定-x XID的信息

来自pg_clog

```
ocz@db-172-16-3-150-> cd $PGDATA/pg_clog
ocz@db-172-16-3-150-> ll -t|head -n 5
total 43M
-rw------- 1 ocz ocz 8.0K Jan 11 09:48 0469
-rw------- 1 ocz ocz 216K Jan 10 21:00 0468
-rw------- 1 ocz ocz 256K Jan 10 12:56 0467
-rw------- 1 ocz ocz 256K Jan 10 09:35 0466
```

取最大值加1然后乘以1048576.

转换成16进制的话相当于取最大值加1然后末尾添加5个0

得到-x 0x046A00000

接下来确定-m XID的信息

来自pg_multixact/offsets

```
ocz@db-172-16-3-150-> cd $PGDATA/pg_multixact/offsets
ocz@db-172-16-3-150-> ll
total 0
```

取最大值加1然后乘以65536.

转换成16进制的话相当于取最大值加1然后末尾添加4个0

没有文件的话使用0加1, 然后末尾添加4个0

得到-m 0x10000

接下来确定-O OFFSET的信息

来自pg_multixact/members

```
ocz@db-172-16-3-150-> cd $PGDATA/pg_multixact/members
ocz@db-172-16-3-150-> ll
total 0
```

取最大值加1然后乘以65536.

转换成16进制的话相当于取最大值加1然后末尾添加4个0

没有文件的话使用0加1, 然后末尾添加4个0

得到-O 0x10000

最后, 不确定的值有2个 :

```
-e XIDEPOCH     ，如果么有使用 slony或者londiste这种基于触发器的数据同步软件，则-e意义不大，它实际
-o OID     ，系统会自动跳过已经分配的OID，自动容错，例如OID被别的程序使用掉了，PG会自动生成下一个OI
```

可以[illegible]个值.

执行pg_resetxlog 如下：

```
ocz@db-172-16-3-150-> pg_resetxlog -l 0x1,0x96E8,0x60 -x 0x046A00000 -m 0x10000 -O 0x10000
pg_resetxlog: pg_control exists but is broken or unknown version; ignoring it
Transaction log reset
```

8. 记下pg_controldata信息, 与前面的pg_controldata输出进行比对

```
ocz@db-172-16-3-150-> pg_controldata
pg_control version number:            922
Catalog version number:               201204301
Database system identifier:           5832008033851373032
Database cluster state:               shut down
pg_control last modified:             Fri 11 Jan 2013 10:09:44 AM CST
Latest checkpoint location:           96E8/60000020
Prior checkpoint location:            0/0
Latest checkpoint's REDO location:    96E8/60000020
Latest checkpoint's TimeLineID:       1
Latest checkpoint's full_page_writes: off
Latest checkpoint's NextXID:          0/1184890880
Latest checkpoint's NextOID:          10000
Latest checkpoint's NextMultiXactId:  65536
Latest checkpoint's NextMultiOffset:  65536
Latest checkpoint's oldestXID:        3479858176
Latest checkpoint's oldestXID's DB:   0
Latest checkpoint's oldestActiveXID:  0
Time of latest checkpoint:            Fri 11 Jan 2013 10:09:44 AM CST
Minimum recovery ending location:     0/0
Backup start location:                0/0
Backup end location:                  0/0
End-of-backup record required:        no
Current wal_level setting:            minimal
Current max_connections setting:      100
Current max_prepared_xacts setting:   0
Current max_locks_per_xact setting:   64
Maximum data alignment:               8
Database block size:                  8192
Blocks per segment of large relation: 131072
WAL block size:                       16384
Bytes per WAL segment:                16777216
Maximum length of identifiers:        64
Maximum columns in an index:          32
Maximum size of a TOAST chunk:        1996
Date/time type storage:               64-bit integers
Float4 argument passing:              by value
Float8 argument passing:              by value
```

注意[illegible]控制文件读取到的不确定的-e XIDEPOCH和-o OID信息如下：

也就是initdb后的初始值.

```
Latest checkpoint's NextXID:          0/1184890880  : XIDEPOCH=0
Latest checkpoint's NextOID:          10000
```

与修复pg_control前发生了变化的值如下 :

修复前

```
Database system identifier:           5832000131111550393
pg_control last modified:             Fri 11 Jan 2013 09:48:18 AM CST
Prior checkpoint location:            96E8/5EE5C698
Latest checkpoint's full_page_writes: on
Latest checkpoint's NextXID:          0/1183842312
Latest checkpoint's NextOID:          116414
Latest checkpoint's oldestXID:        1006759584
Latest checkpoint's oldestXID's DB:   1
Time of latest checkpoint:            Fri 11 Jan 2013 09:48:18 AM CST
Current wal_level setting:            hot_standby
Current max_connections setting:      1000
Current max_prepared_xacts setting:   10
```

修复后

```
Database system identifier:           5832008033851373032
pg_control last modified:             Fri 11 Jan 2013 10:09:44 AM CST
Prior checkpoint location:            0/0
Latest checkpoint's full_page_writes: off
Latest checkpoint's NextXID:          0/1184890880
Latest checkpoint's NextOID:          10000
Latest checkpoint's oldestXID:        3479858176
Latest checkpoint's oldestXID's DB:   0
Time of latest checkpoint:            Fri 11 Jan 2013 10:09:44 AM CST
Current wal_level setting:            minimal
Current max_connections setting:      100
Current max_prepared_xacts setting:   0
```

9. 启动数据库

```
ocz@db-172-16-3-150-> pg_ctl start
server starting
ocz@db-172-16-3-150-> LOG:  00000: loaded library "pg_stat_statements"
LOCATION:  load_libraries, miscinit.c:1249
```

10. 是否正常, 然后新插入数据

```
ocz@db-172-16-3-150-> psql digoal digoal
psql (9.2.1)
```

```
Type "help" for help.
digoal=> select min(oid),max(oid),count(*) from oid_test ;
  min  |  max   | count
-------+--------+--------
 16397 | 116396 | 100000
(1 row)
```

数据可以正常访问.

新插入数据 :

```
digoal=> insert into oid_test select generate_series(100001,200000);
INSERT 0 100000
digoal=> select min(oid),max(oid),count(*) from oid_test ;
  min  |  max   | count
-------+--------+--------
 16384 | 116396 | 200000
(1 row)
digoal=> select oid,* from oid_test where oid=16397;
  oid  |   id
-------+--------
 16397 |      1
 16397 | 100014
(2 rows)
```

注意oid出现了重复, 印证了PostgreSQL中的说明, OID不确保唯一性.

11. 关闭数据库, 并记下pg_controldata的信息, 看看有何变化.

```
ocz@db-172-16-3-150-> pg_ctl stop -m fast
waiting for server to shut down.... done
server stopped
ocz@db-172-16-3-150-> pg_controldata
pg_control version number:            922
Catalog version number:               201204301
Database system identifier:           5832008033851373032
Database cluster state:               shut down
pg_control last modified:             Fri 11 Jan 2013 10:16:18 AM CST
Latest checkpoint location:           96E8/61000020
Prior checkpoint location:            96E8/60DFF470
Latest checkpoint's REDO location:    96E8/61000020
Latest checkpoint's TimeLineID:       1
Latest checkpoint's full_page_writes: on
Latest checkpoint's NextXID:          0/1184890883
Latest checkpoint's NextOID:          116385
Latest checkpoint's NextMultiXactId:  65536
Latest checkpoint's NextMultiOffset:  65536
Latest checkpoint's oldestXID:        1006759584
Latest checkpoint's oldestXID's DB:   1
Latest checkpoint's oldestActiveXID:  0
Time of latest checkpoint:            Fri 11 Jan 2013 10:16:18 AM CST
```

```
Minimum recovery ending location:     0/0
Backup start location:                0/0
Backup end location:                  0/0
End-of-backup record required:        no
Current wal_level setting:            hot_standby
Current max_connections setting:      1000
Current max_prepared_xacts setting:   10
Current max_locks_per_xact setting:   64
Maximum data alignment:               8
Database block size:                  8192
Blocks per segment of large relation: 131072
WAL block size:                       16384
Bytes per WAL segment:                16777216
Maximum length of identifiers:        64
Maximum columns in an index:          32
Maximum size of a TOAST chunk:        1996
Date/time type storage:               64-bit integers
Float4 argument passing:              by value
Float8 argument passing:              by value
```

关闭数据库后与刚修复好时的控制文件信息变化如下：

开库前：

```
pg_control last modified:             Fri 11 Jan 2013 10:09:44 AM CST
Latest checkpoint location:           96E8/60000020
Prior checkpoint location:            0/0
Latest checkpoint's REDO location:    96E8/60000020
Latest checkpoint's full_page_writes: off
Latest checkpoint's NextXID:          0/1184890880
Latest checkpoint's NextOID:          10000
Latest checkpoint's oldestXID:        3479858176
Latest checkpoint's oldestXID's DB:   0
Time of latest checkpoint:            Fri 11 Jan 2013 10:09:44 AM CST
Current wal_level setting:            minimal
Current max_connections setting:      100
Current max_prepared_xacts setting:   0
```

关库后：

```
pg_control last modified:             Fri 11 Jan 2013 10:16:18 AM CST
Latest checkpoint location:           96E8/61000020
Prior checkpoint location:            96E8/60DFF470
Latest checkpoint's REDO location:    96E8/61000020
Latest checkpoint's full_page_writes: on
Latest checkpoint's NextXID:          0/1184890883
Latest checkpoint's NextOID:          116385
L          kpoint's oldestXID:        1006759584
La         ckpoint's oldestXID's DB:  1
Ti         test checkpoint:           Fri 11 Jan 2013 10:16:18 AM CST
Current wal_level setting:            hot_standby
Current max_connections setting:      1000
```

```
Current max_prepared_xacts setting:   10
```

小结

1. 使用pg_resetxlog后, 先检查数据一致性, 必要时将数据导出, 使用initdb新建数据库, 再导入.

2. 如果控制文件丢失, 并且没有备份的话, pg_resetxlog你不知道该填啥, 但是可以从pg_xlog目录中获得大概的redo location, 或者pg_resetxlog 会猜测一些值, 直接-f生成控制文件, 启动数据库后, 可能由于XID回归到以前的XID而致使数据"消失", 你可以使用txid_current()函数不断的消耗XID来得到一致的值.

http://blog.163.com/digoal@126/blog/static/16387704020118304315362 2/
http://blog.163.com/digoal@126/blog/static/16387704020125191181366 1/

使用pg_xlogdump从xlog中抽取信息, 包括txid.
http://blog.163.com/digoal@126/blog/static/16387704020134993845555/

pg_resetxlog的版本必须要与数据库集群的版本一致。

注意

控制文件和PostgreSQL的大版本相关，建议用户使用pg_resetxlog前，看一看对应版本的使用说明。

参考

1. man pg_controldata
2. man pg_resetxlog
3. http://blog.163.com/digoal@126/blog/static/16387704020117123371058 2/
4. http://blog.163.com/digoal@126/blog/static/16387704020129141129495 46/
5. http://www.postgresql.org/docs/9.2/static/wal-internals.html
6. http://www.postgresql.org/docs/9.2/static/app-pgresetxlog.html
7. http://www.postgresql.org/docs/9.2/static/pgupgrade.html
8. http://www.postgresql.org/docs/9.2/static/continuous-archiving.html
9. src/bin/pg_resetxlog/pg_resetxlog.c
10. src/include/catalog/pg_control.h
11. src/bin/pg_controldata/pg_controldata.c
12. src/backend/access/transam/clog.c
13. src/include/access/clog.h
14. src/backend/access/transam/xlog.c
15. ccess/xlog.h

祝大家玩得开心，欢迎随时来 **阿里云促膝长谈业务需求** ，恭候光临。

阿里云的小伙伴们加油，努力 **做好内核与服务，打造最贴地气的云数据库** 。

关系型数据库　数据库　PostgreSQL

评论

登录后可评论

pgfans
2016-08-16

德哥，我在9.4.5上重复同样的测试遇到一个问题就是 - O 参数的计算。现在的版本说明是： A safe value for the next multitransaction offset (-O) ========== ========== can be determined by looking for the numerically largest file name in the directory pg_multixact/members under the data directory, adding one, and then multiplying by 52352. As above, the file names are in hexadecimal. There is no simple recipe such as the ones above of appending zeroes. 如果我的数据库 pg_multixact/members 的最后文件名是：40BF 那么 -O 这个参数应该是多少啊？

5

pis-j
2017-03-28
德哥，pg_control丢失倒是不太可能，还有什么其他场景需要用pg_resetxlog去做数据库修复的吗？

德哥
20 08-17
s 你是什么版本的PG，就看对应版本的说明。不同的版本可能不一样。

pgfans

2016-08-17

我设置为-O 0x40C00000 重新创建pg_control之后报错： could not access status of transaction 1086324736","Could not open file ""pg_multixact/members/510E"": No such file or directory.

pgfans

2016-08-17

是40C0*52352 还是40C0*65536。文档描述multiplying by 52352

德哥

2016-08-17

用对应版本的说明，9.4 : 40C0 * 65536

阿里云数据库

+ 订阅

帮用户承担一切数据库风险，给您何止是安心！

官方博客

如宝马3系和5系：PolarDB-X 与 DRDS 并驾齐驱

PolarDB-X 存储架构之“基于Paxos的最佳生产实践”

让SQL优化器更准确！AnalyticDB PG版发布统计信息自动收集功能

PolarDB-X 私有协议：提升集群的性能和稳定性

今天10点，阿里云重磅出品《超级探访·春节特辑》首次上线

展开

链接

RDS-MySQL版: https://cn.aliyun.com/product/rds

MongoDB版: https://cn.aliyun.com/product/mongodb

HBase版: https://cn.aliyun.com/product/hbase

推荐