2022数据库大咖讲坛(第1期)

源于PG的数据库技术趋势研讨

中国DBA联盟 All China DBA Union

墨天轮

中国DBA联盟
All China DBA Union

墨天轮

通过细节点对比来看
国产数据库与开源数据库的相互促进

Difference between MogDB and PostgreSQL

云和恩墨　彭冲

个人简介

彭冲

PostgreSQL ACE

云和恩墨

PG技术顾问

中国 天津

多年从事基于PostgreSQL数据库的软件研发，擅长于PL/PGSQL业务迁移及优化，曾在天津通卡智能公交项目中作为核心开发人员将项目成功实施于委内瑞拉交通部现场，并成功完成全国公交系统整体迁移升级(包括应用软件升级及PG数据库9.0升级至9.3)。后在银联体系从事商户交易数据异构处理工作。2019年加入云和恩墨，担任PG技术顾问，专职从事PG相关的技术研究工作，热衷于PostgreSQL实践技术分享，在自己的岗位积极推广使用PostgreSQL。

PostgreSQL工作实践

PostgreSQL工作实践分享

共92篇 | 130k浏览 | 4天前 更新

- PostgreSQL资料集串接功能string_agg示例
- PostgreSQL数据库json类型与Java中JsonString的映射
- PostgreSQL使用序列名获取表及列信息
- PostgreSQL数据库触发器分两步拆解创建
- 掌握PostgreSQL新特性学习笔记五：PostgreSQL 13
- 掌握PostgreSQL新特性学习笔记四：PostgreSQL 12
- 掌握PostgreSQL新特性学习笔记三：PostgreSQL 11
- 掌握PostgreSQL新特性学习笔记二：PostgreSQL 10
- 掌握PostgreSQL新特性学习笔记一：PostgreSQL 9.6
- PostgreSQL并行特性简介
- PostgreSQL(update ≈ delete + insert)
- 翻译:PostgreSQL自动生成主键如何选择?
- PostgreSQL中的restartpoint重启点

https://www.modb.pro/db/21482

MogDB技术分享

云和恩墨MogDB技术分享：包括标准化部署及参数配置、MogHA主从切换、MTK迁移等。

共108篇 | 39k浏览 | 7小时前 更新

置顶推荐

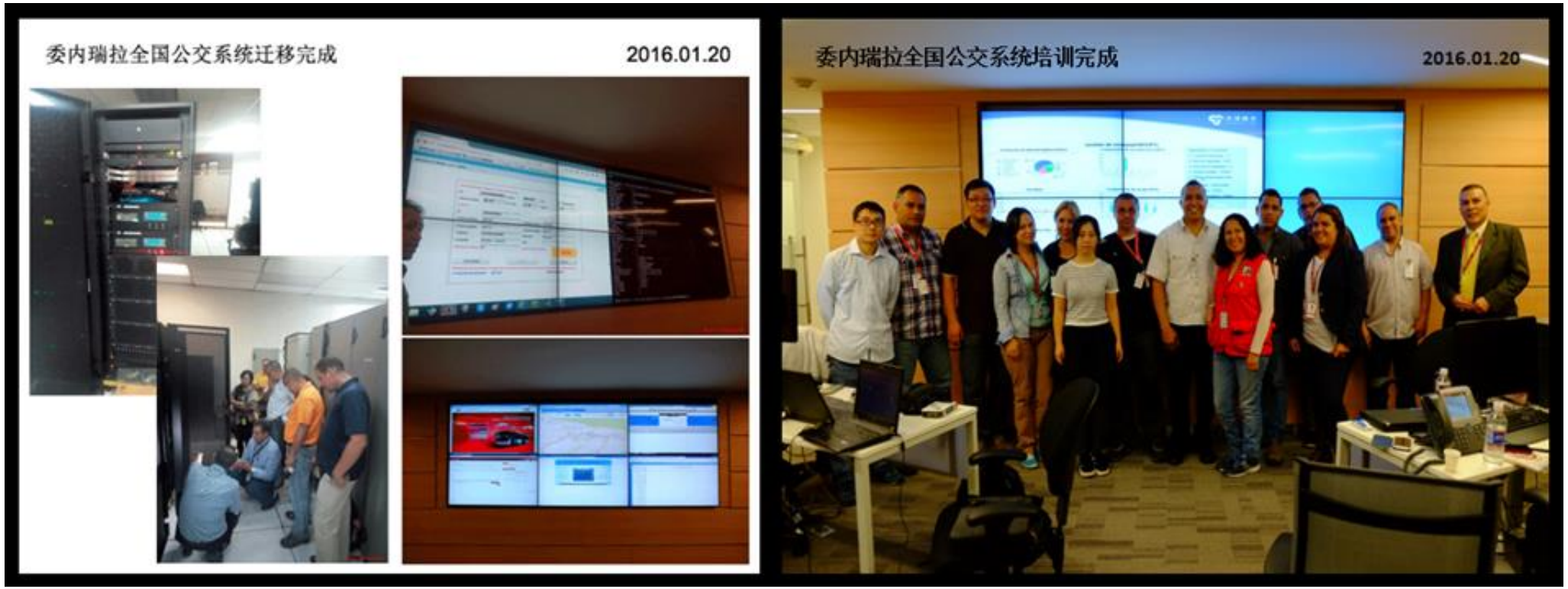
- MogDB数据库问答集萃
- MogDB数据库支持R2DBC响应式协议访问
- MogDB数据库package关键字的两种用法
- openGauss/MogDB大对象LargeObject存取测试
- openGauss/MogDB/PostgreSQL数据库高可用及负载均
- openGauss/MogDB零字节问题处理
- openGauss/MogDB与PostgreSQL表分区语法测试
- openGauss/MogDB数据库完美适配Grafana及Prometh
- openGauss/MogDB/PostgreSQL数据库易犯的十大错误

https://www.modb.pro/db/214825

内容简介

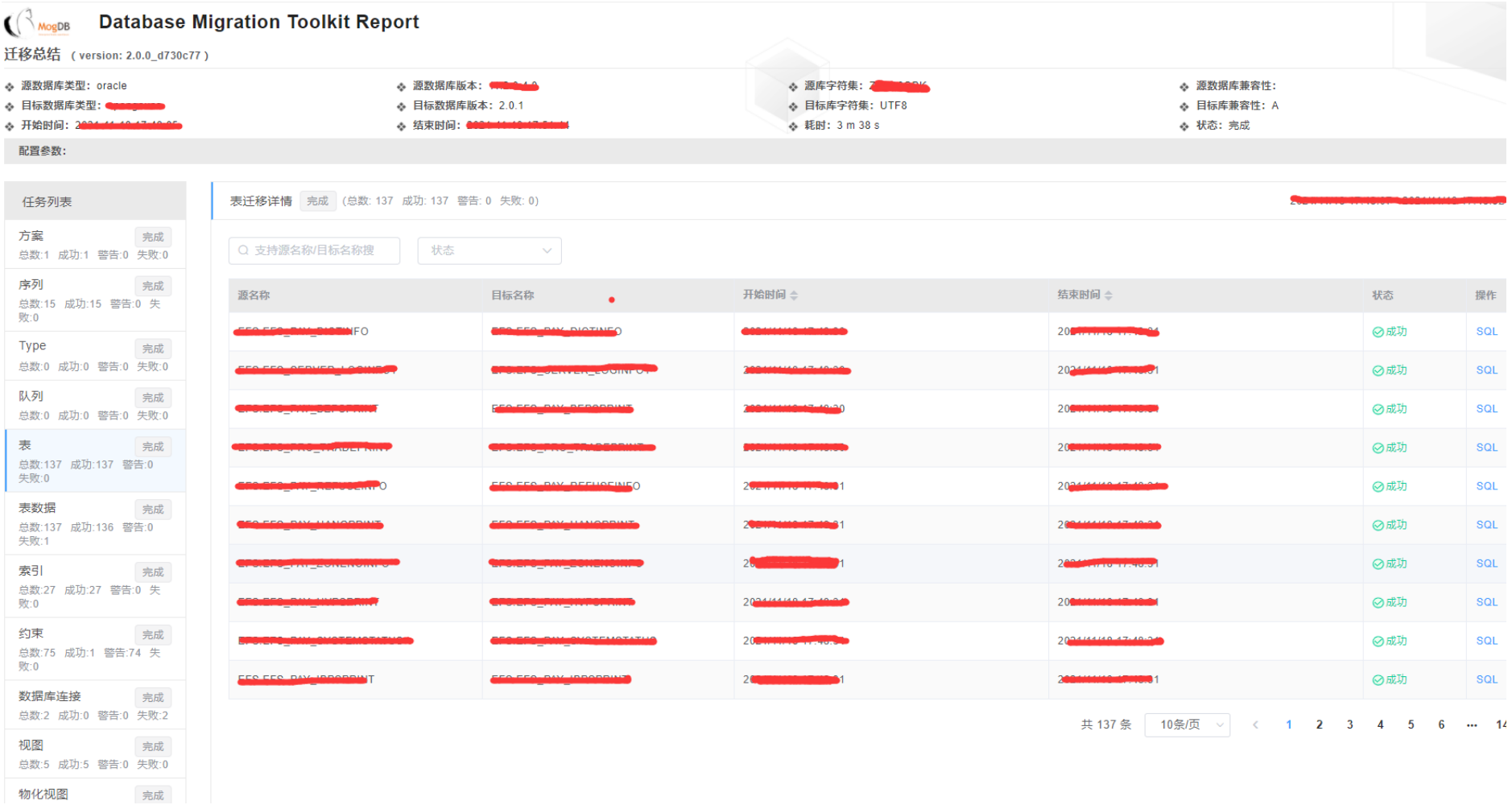
01 我的项目案例

PostgreSQL项目实践



- 公交行业：智能公交系统（IC卡收费、车辆调度）
- 技术架构：Java + Applet + PostgreSQL + Linux
- 应用技术点：使用动态函数生成模板报表、数据库内置分组统计、基于json热部署监控大屏、数据库与应用层加解密互联互通、异地消费数据回传整合。

MogDB项目实践



已成功落地YC、MS、BHCX、HB等银行

02 主从集群对比

主从配置对比

PostgreSQL

- synchronous_commit(PG v9.6)

off, local, remote_write, on, remote_apply。

- quorum commit (PG v10)

synchronous_standby_names='FIRST/ANY 2 (node1, node2, node3)'

- application_name

primary_conninfo= 'application_name=node1 …'

MogDB

- synchronous_commit

off, local, remote_write, remote_receive, on, remote_apply。

- quorum commit

synchronous_standby_names='FIRST/ANY (dn_6002, dn_6003)'

- application_name(独立的配置参数，更方便进行设置)

application_name='dn_6001'

主从启动模式对比

PostgreSQL

```
[postgres@ecs-bethune-demo ~]$ ps f -u postgres
  PID TTY      STAT   TIME COMMAND
19312 pts/26   S      0:00 -bash
19369 pts/26   R+     0:00  \_ ps f -u postgres
31202 ?        Ss     0:18 /opt/pg124/bin/postgres -D /opt/data6000
31203 ?        Ss     0:27  \_ postgres: pg104_6000: logger
31205 ?        Ss     0:06  \_ postgres: pg104_6000: checkpointer
31206 ?        Ss     0:14  \_ postgres: pg104_6000: background writer
31207 ?        Ss     1:11  \_ postgres: pg104_6000: walwriter
31208 ?        Ss     0:15  \_ postgres: pg104_6000: autovacuum launcher
31209 ?        Ss     0:05  \_ postgres: pg104_6000: archiver   last was 000000040000000100000090
31210 ?        Ss     1:18  \_ postgres: pg104_6000: stats collector
31211 ?        Ss     0:00  \_ postgres: pg104_6000: logical replication launcher
18001 ?        Ss     0:15  \_ postgres: pg104_6000: walsender repuser 139.9.242.121(53682) streaming 1/91A14D48
```

```
[postgres@enmotech-bpv3-1 ~]$ ps f -u postgres
  PID TTY      STAT   TIME COMMAND
18760 pts/22   S      0:00 -bash
18829 pts/22   R+     0:00  \_ ps f -u postgres
21335 ?        Ss     0:01 /opt/pg124/bin/postgres -D /opt/data6000
21336 ?        Ss     0:55  \_ postgres: pg104_6000: logger
21337 ?        Ss     0:19  \_ postgres: pg104_6000: startup   recovering 000000040000000100000091
21341 ?        Ss     0:07  \_ postgres: pg104_6000: checkpointer
21342 ?        Ss     0:13  \_ postgres: pg104_6000: background writer
21343 ?        Ss     0:07  \_ postgres: pg104_6000: archiver   last was 000000040000000100000090
21344 ?        Ss     1:32  \_ postgres: pg104_6000: stats collector
21345 ?        Ss    13:38  \_ postgres: pg104_6000: walreceiver   streaming 1/91A14D48
```

MogDB

```
root@op_master:/# ps f -u omm
  PID TTY      STAT   TIME COMMAND
    1 ?        Ssl    0:10 mogdb -M primary
root@op_master:/#
```

```
root@op_slave_one:/# ps f -u omm
  PID TTY      STAT   TIME COMMAND
    1 ?        Ssl    0:02 mogdb -M standby
root@op_slave_one:/#
```

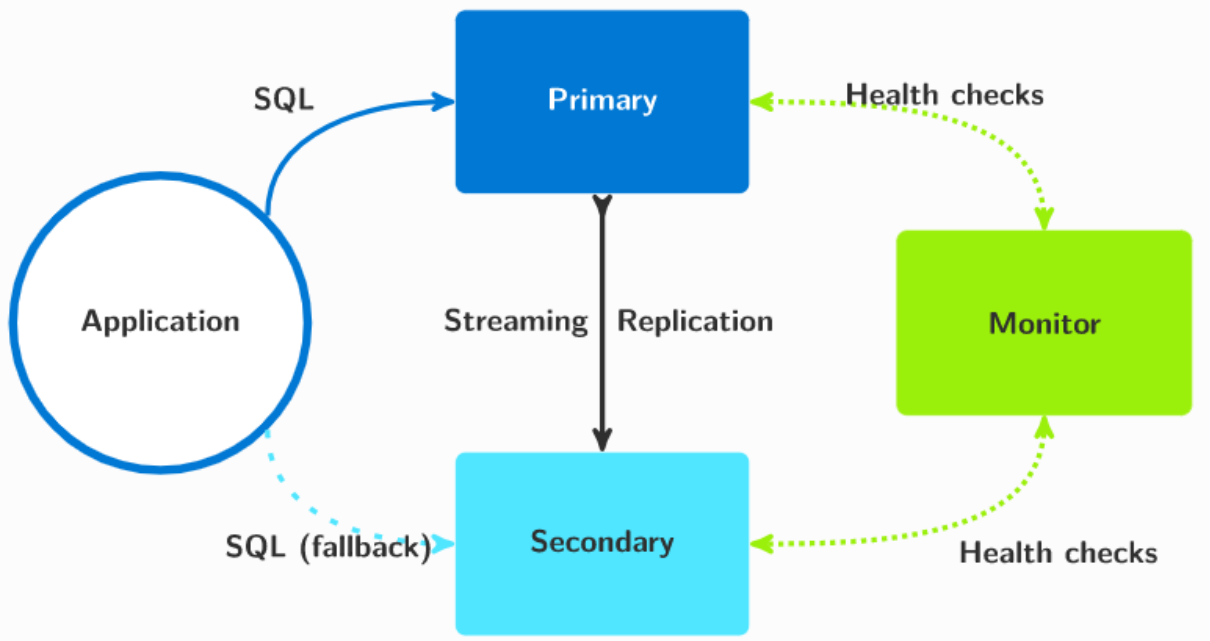
最大可用模式对比

PostgreSQL

pg_auto_failover 为每个 PostgreSQL 服务使用三个节点

- PostgreSQL primary节点
- PostgreSQL standby节点，使用同步热备
- 既充当监控者又充当协调者Monitor 节点

当standby节点被检测为不可用，或者当其lag延迟高于定义的阈值时，Monitor会在主节点上的synchronous_standby_names设置中移除该不可用节点；
在standby节点恢复正常监控之前，不允许进行failover和switchover操作，从而防止数据丢失；
当standby节点已恢复或WAL赶上到定义的阈值内时，同步热备将自动恢复。



pg_auto_failover单standby架构

这种经过特别优化的两节点是可以满足业务连续性的。

最大可用模式对比

MogDB

03 并行特性对比

并行特性对比

PostgreSQL

	14	13	12	11	10	9.6	9.5
Parallel query execution on remote databases	Yes	No	No	No	No	No	No
Foreign data wrapper query parallelism	Yes	No	No	No	No	No	No
Query parallelism for RETURN QUERY	Yes	No	No	No	No	No	No
Parallelized VACUUM for Indexes	Yes	Yes	No	No	No	No	No
Parallelized CREATE INDEX for B-tree indexes	Yes	Yes	Yes	Yes	No	No	No
Parallel hash joins	Yes	Yes	Yes	Yes	No	No	No
Parallel B-tree index scans	Yes	Yes	Yes	Yes	Yes	No	No
Parallel bitmap heap scans	Yes	Yes	Yes	Yes	Yes	No	No
Parallel merge joins	Yes	Yes	Yes	Yes	Yes	No	No
Parallel full table scans (sequential scans)	Yes	Yes	Yes	Yes	Yes	Yes	No
Parallel JOIN, aggregate	Yes	Yes	Yes	Yes	Yes	Yes	No
Parallel query	Yes	Yes	Yes	Yes	Yes	Yes	No
Parallel vacuumdb jobs	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Parallel restore	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Parallel pg_dump	Yes	Yes	Yes	Yes	Yes	Yes	Yes

MogDB

- SMP并行技术；
- COPY并行导入；
- 并行事务日志回放；

支持并行的算子：计划中存在以下算子支持并行。

- Scan：支持行存普通表和行存分区表顺序扫描 、列存普通表和列存分区表顺序扫描。
- Join：HashJoin、NestLoop
- Agg：HashAgg、SortAgg、PlainAgg、WindowAgg（只支持partition by，不支持order by）。
- Stream：Local Redistribute、Local Broadcast
- 其他：Result、Subqueryscan、Unique、Material、Setop、Append、VectoRow、RowToVec

04 连接池特性对比

连接池特性对比

连接池的作用

- cache database connections and reuse

removes the overhead in initializing and closing connections on the database cluster.

- provide a FIFO queue

provide a queue for connections in excess of max_connections so that incoming connections won't be rejected but instead delayed while they wait for the next available connection from the pool.

Java应用连接池

- DBCP
- c3p0
- druid

PostgreSQL连接池

- pgBouncer
- pgPool-II

MogDB线程池

- enable_thread_pool = on
- thread_pool_attr = 'thread_num,group_num,cpubind_info'

05 COPY备份对比

COPY备份

- copy 命令用于表(或查询)与文件之间的相互拷贝；
- copy 命令在数据库服务端操作文件，需要在数据库中配置读写权限;
- \copy 命令在客户端执行导入客户端的数据文件，无需额外申请权限；
- 支持三种格式：文本格式(默认)、CSV格式(可跨平台)和二进制格式；

COPY分割符

- PostgreSQL

```
postgres=> \copy t1 to t1.dat delimiter E'\t'
COPY 2
postgres=> \! cat t1.dat
1       a
2       b
postgres=> \copy t1 to t1.dat
COPY 2
postgres=> \! cat t1.dat
1       a
2       b
```

```
postgres=> \copy t1 to t1.csv (format csv, delimiter ',')
COPY 2
postgres=> \! cat t1.csv
1,a
2,b
postgres=> \copy t1 to t1.csv (format csv)
COPY 2
postgres=> \! cat t1.csv
1,a
2,b
```

```
postgres=> \copy t1 to t1.dat delimiter '@$'
ERROR:  COPY delimiter must be a single one-byte character
```

- MogDB

```
MogDB=> \copy t1 to t1.dat delimiter '@$'
MogDB=> \! cat t1.dat
1@$a
2@$b
```

支持多字符分隔符，

分隔符不能超过10个字节

COPY容错机制对比

PostgreSQL

```
[postgres@node11 ~]$ cat foo.csv
1,one
2
3,three,111
four,4
5,five
[postgres@node11 ~]$ pg_bulkload sample_csv.ctl -p1303
NOTICE: BULK LOAD START
NOTICE: BULK LOAD END
        0 Rows skipped.
        2 Rows successfully loaded.
        3 Rows not loaded due to parse errors.
        0 Rows not loaded due to duplicate errors.
        0 Rows replaced with new rows.
WARNING: some rows were not loaded due to errors.
[postgres@node11 ~]$ cat pg_bulkload_bad.log
2
3,three,111
four,4
[postgres@node11 ~]$ /opt/pg13/bin/psql -p1303
psql (13.3)
Type "help" for help.

postgres=# select * from foo;
 a |  b
---+------
 1 | one
 5 | five
(2 rows)
```

MogDB

```
MogDB=# \! cat tab.dat
 1      one
 2
 3      three   111
 four    4
 5      five
MogDB=# copy foo from '/home/omm/tab.dat'
with(ignore_extra_data,log_errors_data,reject_limit 'unlimited');
COPY 4
MogDB=# select * from foo;
 a |   b
---+-------
 1 | one
 2 |
 3 | three
 5 | five
(4 rows)

MogDB=# \x
Expanded display is on.
MogDB=# select * from pgxc_copy_error_log ;
-[ RECORD 1 ]-----------------------------------------------
relname   | pg_temp_og_6432_13_3_139762165806848.foo
begintime | 2022-01-20 15:09:13.126048+08
filename  | /home/omm/tab.dat
lineno    | 4
rawrecord |  four    4
detail    | invalid input syntax for type bigint: " four    4"
```

COPY进阶用法

PostgreSQL

```
psql --command="\copy schema1.tab to stdout"
|
psql --command="\copy schema2.tab from stdin"
```

```
COPY (SELECT 1) TO PROGRAM '/xxx/xxx/xxx.sh';
```

```
EXECUTE format('COPY (SELECT 1) TO PROGRAM
''/xxx/xxx/xxx.sh --sid=%s --cid=%s'' ', sid, cid);
```

MogDB

```
\copy t1 from '/xxx/text' delimiter ',' parallel 8;
```

```
copy t2(a,b,c) from '/xxx/t2.dat' fixed
formatter(a(0,2),b(2,2),c(4,2));
```

```
copy t3 from '/home/omm/t4.dat'
transform(a text, b float8, c timestamp
without time zone) delimiter ',';
```

06 JDBC驱动对比

SCRAM兼容性

org.postgresql.util.PSQLException:
invalid or unsupported by client scram mechanisms

PostgreSQL

password_encryption

- md5
- scram-sha-256(PG v10)

MogDB

password_encryption_type

- 0表示采用md5方式对密码加密。
- 1表示采用sha256和md5两种方式分别对密码加密。
- 2表示采用sha256方式对密码加密。

级联操作与BatchMode参数

```
org.hibernate.jdbc.BatchedTooManyRowsAffectedException: Batch update returned unexpected row count from update [0];
        at org.hibernate.jdbc.Expectations$BasicExpectation.checkBatched(Expectations.java:71)
        at org.hibernate.jdbc.Expectations$BasicExpectation.verifyOutcome(Expectations.java:46)
        at org.hibernate.jdbc.BatchingBatcher.checkRowCounts(BatchingBatcher.java:68)
        at org.hibernate.jdbc.BatchingBatcher.doExecuteBatch(BatchingBatcher.java:48)
        at org.hibernate.jdbc.AbstractBatcher.executeBatch(AbstractBatcher.java:242)
        at org.hibernate.engine.ActionQueue.executeActions(ActionQueue.java:235)
        at org.hibernate.engine.ActionQueue.executeActions(ActionQueue.java:143)
```

```
//创建对象
Incident inc =new Incident();

Set<Person> handlers = new HashSet<Person>(16);

handlers.add(new Person(100L,100L,"user1"));
handlers.add(new Person(200L,200L,"user2"));

inc.setHandlers(handlers);

//将对象保存到数据库
session.saveOrUpdate(inc);
```

jdbc:postgresql://[ip:port]/dbname?
batchMode=(on/off)

07 表分区特性对比

PostgreSQL表分区

创建分区表（父表）

CREATE TABLE table_name (...)

PARTITION BY { RANGE | LIST | HASH} ({ column_name | (expression) }

创建分区表（子表）

CREATE TABLE table_name

PARTITION OF parent_table FOR VALUES partition_bound_spec

创建子分区

CREATE TABLE table_name

PARTITION OF parent_table FOR VALUES partition_bound_spec

PARTITION BY { RANGE | LIST | HASH} ({ column_name | (expression) }

MogDB表分区

■ 列表分区

```
CREATE TABLE tab_list (
   country varchar(20),
   …
) PARTITION BY LIST (country)
```

■ 范围分区

```
CREATE TABLE pt1(
   id      INTEGER ,
   name    varchar(20) ,
   score   DECIMAL(5,2)
)PARTITION BY RANGE(score)
(    PARTITION P1 VALUES LESS THAN(60) tablespace tbs1,
     PARTITION P2 VALUES LESS THAN(85) tablespace tbs2,
     PARTITION P3 VALUES LESS THAN(MAXVALUE) tablespace tbs2
);
```

■ 哈希分区

```
CREATE TABLE tab_hash (
   part_no varchar(20),
   …
) PARTITION BY HAH (part_no)
```

■ 自动扩展间隔分区

```
CREATE TABLE tab_interval (
   create_date date,
   …
) PARTITION BY RANGET (create_date)
INTERVAL ('1 month')
```

■ 分区表操作

- select * from … partition (p1);
- alter table … add partition p2
- alter table … drop partition p3;
- alter table … merge partitions p4,p5 into partition p4_5;
- alter table … split partition p6 into (partition p61,partition p62);
- alter table … exchange partition (p7) …;
- alter table … truncate partition p8;

表分区对比总结

	PostgreSQL	MogDB
支持方式	继承/声明式分区语法	声明式分区语法
范围分区	支持默认分区	支持三种从句写法
哈希分区	取模的方式，分区数无限制	指定分区名，最多支持64个分区
列表分区	分区数无限制	最多支持64个分区
添加/删除	attach/deatch partition	add/drop/exchange partition
合并/切割/置换	-	merge/split/exchange partition
索引支持	支持在线添加删除	创建全局索引会对整个表添加排他锁
分区键	主键或唯一约束必须包含分区键	主键或唯一约束必须包含分区键， 本地唯一索引必须包含分区键
子分区	支持	-
分区结构查看	\d+	pg_partition视图
分区大小查看	\dP+	pg_total_size()函数

08 程序体特性对比

匿名块Anonymous Block

PostgreSQL

```
do $$
declare
  r record;
begin
  for r in select * from pg_language
  loop
    raise notice 'language : %',r.lanname;
  end loop;
end;
$$ language plpgsql;
```

```
psql (14.1)
Type "help" for help.

postgres=# do $$
postgres$# declare
postgres$#     r record;
postgres$# begin
postgres$#     for r in select * from pg_language
postgres$#     loop
postgres$#         raise notice 'language : %',r.lanname;
postgres$#     end loop;
postgres$# end;
postgres$# $$ language plpgsql;
NOTICE:  language : internal
NOTICE:  language : c
NOTICE:  language : sql
NOTICE:  language : plpgsql
DO
```

MogDB

```
declare
  r record;
begin
  for r in select * from pg_language
  loop
    raise notice 'language : %',r.lanname;
  end loop;
end;
/
```

```
gsql ((MogDB 2.1.0 build 56189e20) compiled at 2022-01
Non-SSL connection (SSL connection is recommended when
Type "help" for help.

MogDB=# declare
MogDB-#     r record;
MogDB-# begin
MogDB$#     for r in select * from pg_language
MogDB$#     loop
MogDB$#         raise notice 'language : %',r.lanname;
MogDB$#     end loop;
MogDB$# end;
MogDB$# /
NOTICE:  language : internal
NOTICE:  language : c
NOTICE:  language : sql
NOTICE:  language : java
NOTICE:  language : plpgsql
ANONYMOUS BLOCK EXECUTE
```

自定义函数User Defined Functions

兼容PostgreSQL风格

```
create or replace function foo() returns number
as $function$
declare
begin
  return 0;
  exception when …
end;
$function$ language plpgsql;
```

兼容Oracle风格

```
create or replace function foo() return number
as
declare
begin
  return 1;
  exception when …
end;
/
```

调用方式

```
postgres=# select pg_control_system();
                      pg_control_system
-------------------------------------------------------------
 (1300,202107181,7021349128397517969,"2022-01-22 10:43:34+08")
(1 row)

postgres=# select * from pg_control_system();
 pg_control_version | catalog_version_no |  system_identifier  | pg_control_last_modified
--------------------+--------------------+---------------------+--------------------------
               1300 |          202107181 | 7021349128397517969 | 2022-01-22 10:43:34+08
(1 row)
```

```
MogDB-# begin
MogDB$#     select pg_switch_xlog();
MogDB$# end;
MogDB$# /
ERROR:  query has no destination for result data
HINT:  If you want to discard the results of a SELECT, use PERFORM instead.
CONTEXT:  PL/pgSQL function inline_code_block line 3 at SQL statement
MogDB=#
MogDB=#
MogDB=# declare
MogDB-#     r record;
MogDB-# begin
MogDB$#     perform pg_switch_xlog();
MogDB$# end;
MogDB$# /
ANONYMOUS BLOCK EXECUTE
```

存储过程Stored Procedure

PostgreSQL

```
create or replace procedure proc1(inout p1 text)
as $$
begin
   raise notice 'procedure parameter: %', p1 ;
end ;
$$
language plpgsql;
```

```
postgres=# call proc1(p1=>'enmotech');
NOTICE:  procedure parameter: enmotech
    p1
---------
 enmotech
(1 row)
```

MogDB

```
create or replace procedure proc1(in p1 text,out
p2 text)
as
begin
       p2 = concat('procedure parameter: ',p1,'.');
end;
/
```

```
MogDB=# call proc1(p1=>'enmtech',p2);
              p2
-------------------------------
 procedure parameter: enmtech.
(1 row)
```

自定义类型Type

```
psql (14.1)
Type "help" for help.

postgres=# select 0.4::real - 0.3::real ;
  ?column?
------------
 0.099999994
(1 row)
```

```
gsql ((MogDB 2.1.0 build 56189e20) compiled at 2022-01-07 18:47:53
Non-SSL connection (SSL connection is recommended when requiring h
Type "help" for help.

MogDB=# select 0.4::real - 0.3::real ;
 ?column?
----------
      0.1
(1 row)
```

PostgreSQL

MogDB

```
postgres=# select ((0.2,0.2)::complex + (0.3,0.5)::complex)
postgres-# as by_operator, f_complex_plus((0.2,0.2),(0.3,0.5))
postgres-# as by_function ;
 by_operator | by_function
-------------+-------------
 (0.5,0.7)   | (0.5,0.7)
(1 row)
```

```
MogDB=# select ((0.2,0.2)::complex + (0.3,0.5)::complex)
MogDB-# as by_operator, f_complex_plus((0.2,0.2),(0.3,0.5))
MogDB-# as by_function ;
 by_operator | by_function
-------------+-------------
 (0.5,0.7)   | (0.5,0.7)
(1 row)
```

```
create type complex as (
rpart decimal/numeric,
ipart decimal/numeric
);
```

```
create operator + (
 leftarg = complex,
 rightarg = complex,
 procedure = f_complex_plus
);
```

```
create or replace function f_complex_plus(c1 complex,c2 complex)
returns complex
as $function$
declare
 l_return complex;
begin
   l_return = ((c1).rpart+(c2).rpart, (c1).ipart+(c2).ipart);
   return l_return ;
end;
$function$ language plpgsql;
```

```
create or replace function f_complex_les(c1 complex,c2 complex)
returns complex
as $function$
declare
 l_return complex;
begin
   l_return = ((c1).rpart-(c2).rpart, (c1).ipart-(c2).ipart);
   return l_return;
end;
$function$ language plpgsql;
```

包接口及实现Package

Package Specification

```
create package employee_management as
  c_empno numeric = 9999;
  function hire_emp (name varchar, job varchar,
            mgr numeric, hiredate timestamp,
            sal numeric, comm numeric,
            deptno numeric) return numeric;
  procedure fire_emp (emp_id numeric);
end employee_management;
/
```

```
MogDB=# \i package_body_employee_management.sql
CREATE PACKAGE BODY
MogDB=# \i package_employee_management.sql
CREATE PACKAGE
MogDB=# \i package_body_employee_management.sql
CREATE PACKAGE BODY
MogDB=# call employee_management.hire_emp('tom',
MogDB(#      'Manager',1,localtimestamp,1,1,1);
 hire_emp
----------
        1
(1 row)
```

Package Body

```
create package body employee_management as
        function hire_emp (name varchar, job varchar,  mgr
numeric, hiredate timestamp, sal numeric, comm numeric,  deptno
numeric) return numeric  as
        declare
                new_empno numeric;
        begin
                select nextval('emp_empno_seq') into new_empno;
                    insert into emp values (new_empno, name, job,
mgr,hiredate, sal, comm, deptno);
                return new_empno;
        end;

        procedure fire_emp(emp_id in number)
        as
          begin
              delete from emp where empno = emp_id;
          end;
end employee_management;
/
```

存储过程commit与exception并存问题

PG里不能在有exception子句的存储过程里使用commit语句

```
begin
        drop table if exists t2;
        alter table t1 rename to t2;
        commit;
        alter table t5 rename to t6;
        exception when others then
              raise notice 'sqlstate=%,sqlerrm=%', sqlstate,sqlerrm;
end;
```

PostgreSQL

```
postgres=# create table t1(id int);
CREATE TABLE
postgres=# call myproc();
NOTICE:  table "t2" does not exist, skipping
NOTICE:  sqlstate=2D000,sqlerrm=cannot commit while a subtransaction is active
CALL
```

详细测试参考 <<存储过程事务控制与异常块>>

MogDB

```
MogDB=# create table t1(id int);
CREATE TABLE
MogDB=# call myproc();
NOTICE:  table "t2" does not exist, skipping
CONTEXT:  SQL statement "drop table if exists t2"
PL/pgSQL function myproc() line 3 at SQL statement
NOTICE:  sqlstate=42P01,sqlerrm=relation "t5" does not exist
```

存储过程重载问题

错误示例

```
create or replace procedure myproc(in p1 varchar,in p2 varchar,
out p3 varchar) as $$
begin
   p3 = p1;  raise notice 'procedure parameter: %', p1 ;
end ;
$$ language plpgsql;
```

```
create or replace procedure myproc(in p1 varchar,in p2 varchar)
as $$
begin
   raise notice 'procedure parameter: %', p1 ;
end ;
$$ language plpgsql;
```

正确示例

```
create or replace procedure myproc3(in p1 varchar, in p2 varchar)
package as
begin
   raise notice 'procedure parameter: %', p1 ;
end ;
/
```

```
create or replace procedure myproc3(in p1 integer, in p2 integer)
package as
begin
   raise notice 'procedure parameter: %', p1 ;
end ;
/
```

详细测试参考 <<MogDB数据库package关键字的两种用法>>

其它问题

- 零字节问题处理

<<零字节问题处理>>

- R2DBC响应式协议访问

<<R2DBC响应式协议访问>>

- 大对象LargeObject存取测试

<<大对象LargeObject存取测试>>

- 高可用及负载均衡JDBC参数测试

<<高可用及负载均衡JDBC参数测试>>

- Json类型与Java中JsonString的映射处理

<<Json类型与Java中JsonString的映射处理>>


中国DBA联盟
All China DBA Union
墨天轮
感谢聆听！