云和恩墨技术通讯

2024 11月刊

数据驱动 成就未来



智能的数据技术提供商

读者微信：enmoxzs

读者信箱：service001@enmotech.com

简历速递：hr@enmotech.com

业务需求：marketing@enmotech.com

业务咨询：010-65017590

紧急救援：400-660-8755

云和恩墨 ENMOTECH

亲爱的客户：

为了及时共享行业案例，通知共性问题，达成共享和提前预防，以及共同学习国产数据库，我们整理和编辑了《云和恩墨技术通讯》，通过对过去一段时间的知识回顾，故障归纳，以期提供有价值的信息供大家参考。同时，我们也希望能够将热点事件、新的产品特性及其他有价值的信息聚集起来，为您提供具有前瞻性的支持信息，保持对于当前最新的数据库新闻和事件的了解，其中包括重要数据库产品发布、警报、更新、新版本、补丁等。

目录

※ 观点：2024 年 11 月中国数据库排行榜解读

※ 知识：Oracle 通过透明网关以 DBLINK 形式连接 MogDB 的方法

※ 故障：MySQL 8.0 WITH 导致 singal 11 案例分析

※ 经验：提高运维效率，MogDB/openGauss 中如何调用带变量的 SQL 脚本

※ 知识：PostgreSQL 17 增量备份

※ 经验：openGauss/MogDB 内存占用相关 SQL

※ 经验：模拟 Oracle 中 DML、DDL 与 undo 的关系，10046 跟踪 DDL 语句

※ 故障：PostgreSQL 分区表 DDL 操作时导致 pg_dump 等待

※ 公告：第 3 期 openGauss 中级认证 OGCP 直播班招生中！

2024 年 11 月中国数据库排行榜：OB 连续四月居榜首，腾讯云 TDSQL 升第九

文 / 墨天轮编辑部

摘要：墨天轮中国数据库排行新一期已发布，在本期榜单中，前五名的数据库产品排名相对稳定，这些头部产品在技术创新、性能优化和行业应用上持续发力，展现出稳健的增长趋势。

本月中国数据库流行度排行榜更新，本期共有 226 个数据库产品参与，相较于年初的 292 个数据库，产品阵容已进行了更新和优化。近期，多个数据库产品取得了重要成果：

- OceanBase 年度发布会发布了多项创新技术，广受关注；
- PolarDB 的“云原生三层解耦 Serverless 数据库关键技术与应用”项目荣获 2024 年度中国计算机学会（CCF）科技进步一等奖；
- 在《中国云原生事务型数据库厂商技术能力评估，2024》报告中，TDSQL 凭借卓越的技术实力入选代表厂商，并在高性能、高可用、兼容性、安全性等六项能力评估中均获第一；
- KWDB 荣获中国信通院“OSCAR 尖峰开源项目及社区奖”，进一步证明了其在开源数据库领域的影响力。

这些成果展示了中国数据库技术的迅速发展和行业中的卓越表现。接下来，我们将盘点本期榜单中的亮点数据库，分析它们在市场中的表现与未来前景。

一、榜单前五稳中求进，TDSQL 排名首度攀升

在本期榜单中，前五名的数据库产品排名相对稳定，这些头部产品在技术创新、性能优化和行业应用上持续发力，展现出稳健的增长趋势。

排行	上月	半年前	名称	模型	数据处理	部署方式	商业模式	专利	论文	案例	资质	书籍	得分
1	1	1	OceanBase	关系型	HP			151	19	43	15	2	748.91
2	2	2	PolarDB	关系型	HP			592	78	85	14	2	629.19
3	3	↑ 4	TiDB	关系型	HP			40	15	18	9	1	550.75
4	4	↑↑↑ 9	GaussDB	关系型	TP HP			915	62	31	17	5	548.79
5	5	↑↑↑ 8	GoldenDB	关系型	HP			772	49	95	12	2	533.69

图 1: 排行榜前五名得分情况

▸ OceanBase 以 748.91 的得分稳固其在数据库排行榜上的领先地位，并与榜眼拉开近一百分的差距。本月的年度发布会无疑是行业的一大亮点，展现了 OceanBase 在技术创新和产品发展上的显著进步。特别是推出

的 OceanBase 4.2.5 LTS 版本和 OceanBase 4.3.3 GA 版本，在性能、兼容性、稳定性和安全性等方面进行了全面优化和增强。OceanBase 4.2.5 LTS 版本针对核心系统所要求的高读写性能、高兼容性、高稳定性进行了进一步优化，TP 性能提升了 26%，并且加强了多模一体化能力，新增支持 Redis 模型。

▸ PolarDB 在本期榜单中仍以 629.19 的高分稳居前列，特别值得一提的是，阿里云的“云原生三层解耦 Serverless 数据库关键技术与应用”项目荣获 2024 年度中国计算机学会（CCF）科技进步一等奖。该架构有效解决了 Serverless 环境下的跨机事务迁移问题，将跨机迁移时间压缩至 0.5 秒以内，整体性能提升数十倍。

▸ TiDB 凭借其“自主开源”和“技术引领”的核心优势，持续展现出强劲的竞争力，并在本月以 550.75 分的优异成绩跻身前列。作为全球开源数据库的领军者，TiDB 不仅拥有超过 2,200 名贡献者和 3,000 多家活跃用户，还荣获了Gartner ® Peer Insights ™云数据库管理系统“客户之选”殊荣，成为唯一获此荣誉的中国数据库。在技术创新方面，据悉 TiDB 将在即将发布的 8.4 版本中集成向量搜索功能。该功能的引入预计将在 AI 应用和银行业务等领域带来显著进步，进一步提升其在多样化数据处理和智能应用场景中的竞争力。让我们共同期待 TiDB 在未来的表现！

▸ GaussDB 以 548.79 分稳居第四，作为具有 20 余年技术沉淀的数据库，GaussDB 已积累了丰富的专利，这些技术积累不仅提升了产品的成熟度，也增强了其在数据库领域的竞争力。GaussDB 不仅在华为内部成功保障了业务连续性，其技术能力也在银行、保险、证券、能源等行业的核心业务系统中得到广泛应用，展现出在多行业中的稳定性和高性能表现。

▸ GoldenDB 在本月数据库排行榜中以 533.69 分再次位居第五。自 2019 年起，GoldenDB 系统相继在中信银行的信用卡核心业务和总行账务核心业务中投入生产，至今已稳定运行超过四年，充分体现了其在高可用性和可靠性方面的技术优势。目前，GoldenDB 在银行核心系统市场的投产数量占比达 50%，在次核心及非银核心系统市场的投产占比达 32%，这两项数据均居行业首位。

排名变化集中在第六至第十位，竞争加剧。凭借不断的技术突破，这些产品为整个数据库行业注入了新的活力和多样性。

6	↑ 7	↓ 5	金仓数据库	关系型	TP AP			534	1	62	12	3	511.99
7	↓ 6	↓ 6	GBASE	关系型	AP TP			271	0	80	13	0	504.26
8	8	↓ 7	达梦数据库	关系型	TP			518	0	21	8	12	482.35
9	↑ 10	↑ 10	TDSQL	关系型	HP			136	23	18	13	0	461.64
10	↓ 9	↓↓↓ 3	openGauss	关系型	TP			570	20	34	7	6	435.35

图 2: 排行榜第六至十名得分情况

▸ 金仓数据库本月得分为 511.89 分，排名上升一位。近期，金仓数据库在“中国滑雪信创数据库建设项目”中以第一名中标，此外凭借在软件行业的创新能力和卓越表现，电科金仓荣获“2024 年软件和信息技术服务优秀企业”称号。未来，期待金仓数据库在技术创新和市场拓展方面持续取得新突破。

▸ GBase 在本月的数据库排行榜中以 504.26 分的高分表现突出。其两款核心产品 GBase 8a 和 GBase 8s 在最近的奖项中取得了显著成绩：
GBase 8a：以其创新的架构和卓越特性荣获“第二十六届中国国际软件博览会大奖（银奖）”，这一奖项是对其技术能力与商用价值的高度认可。
GBase 8s：入选“金融信创生态实验室 - 金融信创解决方案（第一批）”，并荣获“2022 金融行业核心业务系统优秀解决方案”奖。

▸ 本月达梦数据库保持第八位，排名不变。根据2024 年前三季度财报，达梦数据库实现营收约6.29 亿元人民币，同比大幅增长 40.96%。这一显著增长不仅反映了公司在市场中的强劲竞争力与盈利能力，也进一步巩固了其在国产数据库领域的领先地位。

▸ 腾讯云 TDSQL 在本月的排名中实现了首次突破，成功晋级为第九位。在《中国云原生事务型数据库厂商技术能力评估，2024》报告中，TDSQL 以其卓越的技术实力成功入选代表厂商，并在高性能、高可用、兼容性、安全性、金融行业应用、互联网行业应用等六项能力评估中均获得第一。期待 TDSQL 继续保持其技术优势，推动数据库技术的进一步发展。

▸ 在本月数据库排行榜中，openGauss 虽排名下降，但其技术发展和社区活力依旧引人注目。最新发布的6.0.0 LTS 版本是社区的一个重要里程碑，它汇聚了 940 名开发者的努力，合入 9970 个 PR，尤其在内核能力、资源池化架构、数据全生命周期管理工具和生态兼容性方面取得了显著进步。
此外，在开源四周年之际社区推出的“结队计划”，openGauss 进一步促进了伙伴、客户与社区成员之间的有效联接，期待未来 openGauss 能够继续推动数据库技术的发展和应用，实现生态的繁荣和行业的共赢。

二、盘点数据库市场新动力

在细致盘点了排行榜前十的数据库产品之后，现在让我们将目光转向一些在技术创新和市场表现上同样不容忽视的数据库产品。它们通过不断的技术创新和市场拓展，逐渐在激烈的竞争中脱颖而出。

15	↑↑ 17	↑↑↑ 22	KaiwuDB	时序	HP			236	2	7	6	0	98.07
16	16	↑↑ 18	Apache Doris	关系型	AP			0	0	0	0	1	96.68
17	↑↑↑ 21	↑↑↑ 21	虚谷数据库	关系型	TP AP HP			15	0	1	5	0	94.78
24	↑↑↑ 27	↑↑↑ 36	SinoDB	关系型	TP			74	0	0	3	0	52.86
87	↑↑ 89	↓↓↓ 82	YMatrix	时序	HP			1	0	0	1	0	5.78

▸ KaiwuDB 数据库在最新排行榜中排名上升两位，位列第 15 名，反映了其开源后行业热度有明显提升。2024 年 8 月，KaiwuDB 2.0 宣布开源，社区版名为 KWDB 并于 9 月正式捐赠给开放原子开源基金会。9-10 月期间，KWDB 社区在山东大学、江南大学、天津大学等高校开展技术布道，一系列行动显著促进了技术共享和社区扩展，增强了其在数据库开源领域的影响力。10 月，KWDB 荣获中国信通院“OSCAR 尖峰开源项目及社区”奖。同时，凭借近一年来在国家重点产业应用及客户业务系统赋能方面的突出表现，KaiwuDB 获评“2024

最具商业合作价值数据库企业”称号，并被两项大数据产业图谱收录，其在技术创新、场景服务和社区建设方面的活跃态势受到多方认可。

▸ 虚谷数据库 11 月的排行榜中也取得了显著进步，排名上升了 4 位，晋升至第 17 位。虚谷伟业一直致力于提供高效、安全、稳定的数据库解决方案，其独特的单机分布式一体化架构使其能够灵活应对不同场景需求，支持从单机集中式到数百节点分布式的无缝升级。虚谷数据库在安全性和可靠性方面取得了显著成果，特别是其 V12.0 产品在 2024 年首批通过了中国信息安全测评中心的分布式数据库安全可靠测评。这不仅彰显了虚谷的技术实力，也反映了其在数据安全领域的深厚底蕴。

▸ SinoDB 在数据库排行榜中同样表现亮眼，排名上升 3 位，达到第 24 名。这款由星瑞格开发的高端国产数据库管理系统在市场上取得了进展，更在教育领域实现了突破。SinoDB 的实践案例被纳入清华大学出版社的《数据库原理及应用》教材，这不仅是对其技术实力的学术认可，也体现了其在教育领域的实际应用价值。SinoDB 作为三明学院等高校计算机及相关专业的教学资源，将为学生提供实际操作经验，培养未来的数据库技术专家。这一成就加强了星瑞格与教育机构的合作，推动了国产数据库技术的传播和生态发展，为国内数据库行业的自主可控和人才培养做出了积极贡献。

数据库领域的大版本发布一直是技术领域的亮点，上月 Apache Doris 和 YMatrix 也迎来了产品更新，让我们盘点它们在性能、架构和功能上的进步。

▸ Apache Doris 的 3.0 版本带来了存算分离架构，这一变革不仅支持用户根据需求选择存算一体或存算分离模式，降低存储成本，还实现了查询负载的物理隔离。3.0 版本作为湖仓一体演化的重要里程碑，增强了 ETL批处理场景的事务支持和查询执行的可观测性，并通过查询优化器的增强和自适应 Runtime Filter 计算方式，提升了大数据量场景下的性能表现，进一步巩固了 Apache Doris 在分布式分析型数据库领域的地位。

▸ YMatrix 最新发布的 6.0 版本引入了多个关键创新，令人瞩目。该版本的 Domino 流计算引擎使得数据库能够通过 SQL 实现业务数据变动的连锁触发，支持实时数据分析。新版本还增强了异地灾备功能，进一步保障关键场景下的数据安全。此外，存储引擎和向量化执行引擎的持续优化显著提升了数据库的性能和稳定性。

这些数据库的进展和成就，不仅展示了国产数据库技术的快速发展，也体现了它们在技术创新、开源贡献、教育合作和数据安全等方面的卓越表现。随着国产数据库技术的不断进步，我们期待它们在未来能够取得更多的成就。

三、专家观点



严少安

墨天轮 MVP，Oracle ACE，PostgreSQL ACE，TiDB MOA，拥有二十余项国产数据库认证。公众号“少安事务所”作者。

11 月中国数据库流行度排行榜更新，前三甲 OPT 基本盘稳定，第四到第十产品排名虽略有变化，但分数大步领先于十一名之后的产品。回看过往半年，第 11 到 30 名中，有几款产品排名上涨明显，11.14 举办国产数据库创新生态大会的 YashanDB 排名上升 4 位 (13 ← 17)，8.23 时序数据库 KaiwuDB 2.0 正式开源，其排名上升 7 位 (15 ← 22)，双入围安全可靠评测的虚谷数据库排名上升 4 位 (17 ← 21)、神通数据库排名上升 16 位 (19 ← 35)，助力高校数据库学科建设编入教材《数据库原理及应用》的星瑞格 SinoDB 排名上升 12 位(24 ← 36)，10.13 中国移动磐维数据库一体机发布，磐维数据库过去半年排名上升 15 位 (27 ← 42)，10.24 海量数据产品发布会带来产品体系全新升级，其中包括 Vastbase 数据库，排名上升 40 位 (28 ← 68)。排名大幅上升也意味国产数据库厂商在外宣上花了大力气，好产品是要讲出来的，更是长时间用出来的，不断提升产品质量、多在稳定性和易用性等方面下功夫就是炼内功，内外兼修方得始终。

立冬节气到了，是时候准备总结忙碌的 2024 年。从集中式数据库到分布式数据库，在数据库选型时还是要从实际需求出发，小而美的集中式数据库可以覆盖大多数业务场景，海量级数据交给（云原生）分布式数据库，集中式、分布式数据库都有通过安全可靠评测的产品，DBA 运维省心，客户使用安心。从独栋大楼办公到全国分布式办公地点，集中力量办大事没有错，只是数据库作为基础软件，数据库厂商服务的对象已遍布全国，数据库内核开发人才也分布于不同城市，我们喜闻乐见一些厂商加速在多地开设办公室，有些还允许员工远程办公。

从学习单一数据库到学习多种国产数据库，信创 2.0 时代，对于从业者不能只停留于学习一两种数据库，而是要按需快速上手多种数据库，墨天轮平台提供了诸多课程、知识图谱和学习路径，来墨天轮学数据库，轻轻松松不迷路。从综合大会到各数据库厂商城市站活动，诸如数据技术嘉年华 DTC 这般综合性会议自然不可或缺，近来由厂商搭建线下主题交流会议逐渐增多，或许这种交流形式会更有料、更聊的开。中国数据库不仅要流行于互联网，更需要重返线下，与客户见面，与用户见面，与行业上下游见面。

知识：Oracle 通过透明网关以 DBLINK 形式连接 MogDB 的方法

文 / 罗海雄

Oracle 透明网关是 Oracle 公司的一个连接访问异构数据库的组件。通过透明网关，Oracle 可以以数据库链接 (dblink) 的方式连接异构数据库。

目前，通过 Oracle Database Gateway for ODBC，可以实现在 Oracle 数据库中直接访问 MogDB 数据库

。下面是具体的实现步骤：

1. 下载 Oracle 透明网关软件

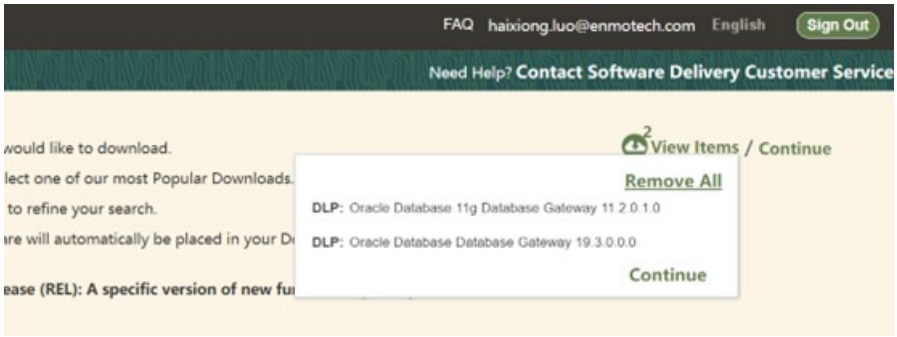
1.1 登录

1.2 搜索 database gateway



1.3 选择合适版本

1.4 选择 Continue

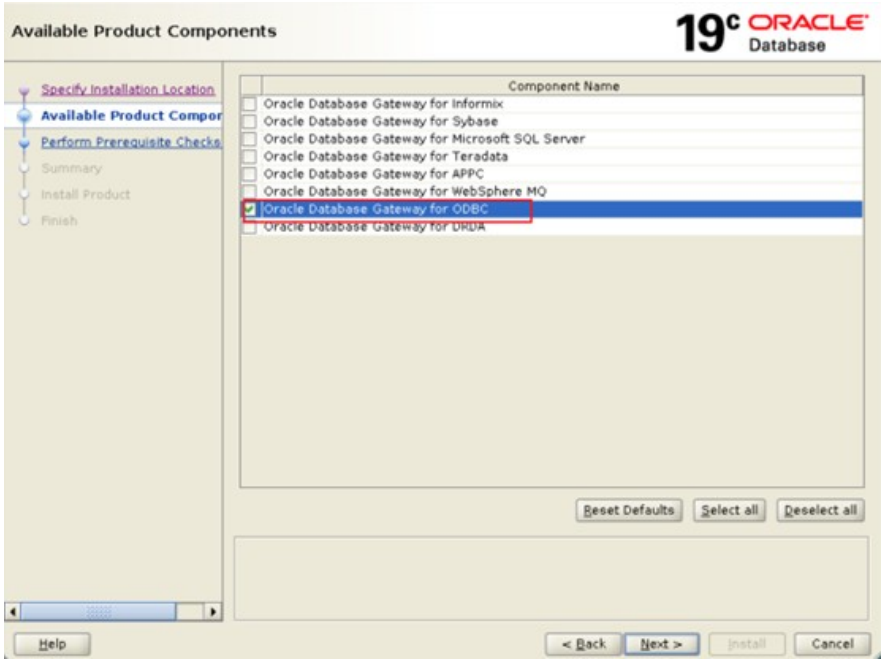


1.5 选择合适操作系统平台

开始下载

2. 安装 Oracle 透明网关软件

不同版本界面可能不一样
解压，并运行 ./runInstaller
关键步骤在于选择 Oracle Database Gateway for ODBC



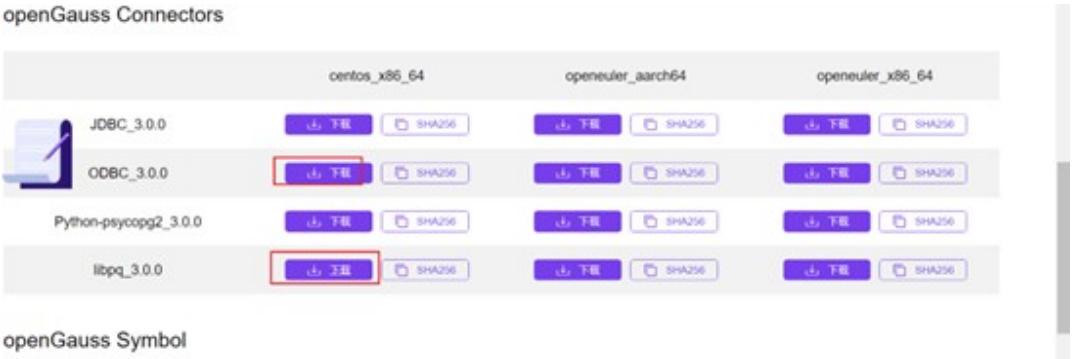
完成安装

3. 安装 unixODBC

以 root 用户直接运行：

yum install unixODBC unixODBC-devel

4. 下载 openGauss ODBC 驱动和 libpq 驱动



并一起解压到 $ORACLE_HOME/hs/ogodbc 下（或者其他地方也可以的）。

5. 配置 Oracle LISTENER/TNSNAME

```
cd $ORACLE_HOME/network/admin


n  listener.ora
LISTENER_HS =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1567))
    )
  )
SID_LIST_LISTENER_HS=
  (SID_LIST=
      (SID_DESC=
         (SID_NAME=mogdb)
         (ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1)
         (ENV="LD_LIBRARY_PATH=/u01/app/oracle/product/19c/dbhome_1/hs/odbc")
         (PROGRAM=dg4odbc)
      )
      (SID_DESC=
         (SID_NAME=mogdb2)
         (ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1)
         (ENV="LD_LIBRARY_PATH=/u01/app/oracle/product/19c/dbhome_1/hs/odbc ")
         (PROGRAM=dg4odbc)
      )
  )
```

其中，

- LISTENER_HS 为监听名字，可随意选择
- 1567 为端口号，可随意选择，不重复即可
- mogdb/mogdb2 为透明网关 SID, 理论上可随意选择（不超过 8 位，不以数字开头）
- /u01/app/oracle/product/19c/dbhome_1 为 ORACLE_HOME, 根据实际情况做相应改动即可

可支持一个监听负责多个数据库，通过多个 SID_DESC 实现

也可以把 SID_DESC 加到现有的 LISTENER 当中，实现默认 LISTENER 同时监听普通连接和透明网关连接。

```
n  tnsnames.ora
mogdbtns =
  (DESCRIPTION=
```

```
    (ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1567))
    (CONNECT_DATA=(SID=mogdb))
    (HS=OK)
  )
Mogdb2tns =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1567))
    (CONNECT_DATA=(SID=mogdb2))
    (HS=OK)
  )
```

其中，

- mogdbtns 为 TNS 名字，可随意选择
- 1567 为端口号，对应 listener 内容
- mogdb 为透明网关 SID, 对应 listener 内容

如果连接多个数据库，则需要建立多个 tnsname。

6. 配置 ODBC.ini

目录和文件名其实可任选，为了方便管理，建议 $ORACLE_HOME/hs/admin

```
cd $ORACLE_HOME/hs/admin
vi odbc.ini
[MOGDB]
Servername=192.168.2.131
Port=10086
Database=postgres
Driver=/u01/app/oracle/product/19c/dbhome_1/hs/odbc/psqlodbcw.so
Username=tpcc
Password=tpcc@123
[MOGDB2]
Servername=192.168.2.131
Port=10199
Database=postgres
Driver=/u01/app/oracle/product/19c/dbhome_1/hs/odbc/psqlodbcw.so
Username=tpcc
Password=tpcc@123
```

其中

- MOGDB/MOGDB2 为 ODBC 的数据源名称，可任选
- Driver 为 ODBC 驱动解压目录下的 psqlodbcw.so
- Servername/Port/ Database/ Username/Password 为目标库相应信息

7. 配置 hs/admin/init.ora

```
cd $ORACLE_HOME/hs/admin

vi init<SID>.ora
```

此处 SID 对应 listerner.ora 里面的 SID_NAME 的定义，如果多个 SID_NAME，则需要多份 init.ora

最核心的几个参数：

```
HS_FDS_CONNECT_INFO = MOGDB

set ODBCINI = /home/oracle/.odbc.ini

HS_FDS_SHAREABLE_NAME = /home/oracle/ogodbc/lib/psqlodbcw.so

HS_FDS_QUOTE_IDENTIFIER = FALSE

set LowerCaseIdentifier = on

HS_FDS_TRACE_LEVEL = 1

HS_NLS_NCHAR = UCS2
```

其中

- HS_FDS_CONNECT_INFO 对应 odbc.ini 里面的数据源名字；
- ODBCINI 对应上一步 odbc.ini 的完整路径及文件名；
- HS_FDS_SHAREABLE_NAME 对应 odbc.ini 的驱动文件；
- HS_FDS_QUOTE_IDENTIFIER = FALSE 解决字段名大小写问题；
- set LowerCaseIdentifier = on 解决字段名大小写问题；
- HS_FDS_TRACE_LEVEL 开始时可以设置大一点，方便 Debug，比如 255，稳定下来可以改为 0 或者 1；
- HS_NLS_NCHAR = UCS2 有时候连接失败需要设置这个参数，应该是字符集不匹配时。

8. 启动 listener 并简单测试

```
lsnrctl start LISTENER_HS
```

名字对应 listener 里的名字

```
tnsping mogdb

Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1567)) (CONNECT_DATA =
```

```
(SERVER = DEDICATED) (SID_NAME = mogdb) (SERVICE_NAME = mogdb)) (HS=OK))

OK (10 msec)
```

注意，此处的 OK 仅代表 LISTENER 正常工作，还未和远程数据库发生关系。

9. 创建 DBLINK

Create database link mogdb_link connect to “user” identified by “pass” using ‘mogdb’;

- mogdb_link 是 dblink 名字
- mogdb’ 对应 tnsnames.ora 里的内容
- 注意前面的 user/pass 加双引号，using 后面的 mogdb 加单引号

10. 测试 DBLINK

```
Select * from pg_class@mogdb_link;

select “relname” from pg_class@mogdb;
```

目前字段名称需要加括号，暂无合适的绕过方法

如果可以，建议修改应用，涉及远程表的部分 SQL 中使用字段名称小写 + 双引号的方式，如果实在不想改应用代码，可联系恩墨工程师进行定制化调整。

11. 遗留问题

- 如果语句报错，后面的访问也会报错

需要显式 rollback 或者断开会话重连。

odbc.ini 里面设置 ForExtensionConnector=1 可能可以解决这个问题，还需要进一步测试验证。

- 表名大小写问题

访问 MogDB 的表的指定字段需要使用小写并加双引号。

如果可以，建议修改应用，涉及远程表的部分 SQL 中使用字段名称小写 + 双引号的方式，如果实在不想改应用代码，可联系恩墨工程师进行定制化调整。

故障：MySQL 8.0 WITH 导致 singal 11 案例分析

文 / 崔虎龙

在使用 MySQL 8.0.26 时，遇到了与 WITH 通用表达式导致 mysqld got signal 11 频繁崩溃问题。按照之前分享过的问题排查思路，阶梯式去查找和分析有用的信息。



下面分享分析问题的完整思路和技术内容。

1. 查找问题

首先，需要查看 MySQL 的错误日志。这个日志文件通常包含了关于崩溃的详细信息，如堆栈跟踪、错误代码和导致崩溃的查询语句。

错误记录信息如下所示：

```
# 错误 1：
[Warning] [MY-011825] [InnoDB] Cannot add field `<temporary>_sup_Agreement_Type_280` in table
`tmp`.`#sql1b13_25d83b_1e` because after adding it, the row size is 8126 which is greater than maximum allowed
size (8126) for a record on index leaf page.
15:16:08 UTC - mysqld got signal 11 ;
。。。
/usr/sbin/mysqld(temptable::Handler::index_init(unsigned int, bool)+0xf) [0x253e98f]
。。。
Trying to get some variables.
Some pointers may be invalid and cause the dump to abort.
Query (7f887c428210): select count(0) from ( with tmp1 a...

# 错误 2：
11:21:10 UTC - mysqld got signal 11 ;
```

```
。。。
/usr/sbin/mysqld(TemptableAggregateIterator::Init()+0xde) [0x12b71fe]
。。。
Trying to get some variables.
Some pointers may be invalid and cause the dump to abort.
Query (7f241669d340): with pecom as ( ...
```

从上述错误中有 3 个关键点信息：

1. signal 11。

2.Temptable 相关处理信息。

3.WITH 表达式 SQL 语句。

2. 分析过程

问题核心点已经找到，那就需要结合知识点进行进一步的分析。

1）singal 11

信号 (signal) 是一种软件中断，它提供了一种处理异步事件的方法，也是进程间惟一的异步通信方式。在 Linux 系统中，根据 POSIX 标准扩展以后的信号机制，不仅可以用来通知某种程序发生了什么事件，还可以给进程传递数据。

信号的来源可以有很多种试，按照产生条件的不同可以分为硬件和软件两种。

1、硬件方式

当用户在终端上按下某键时，将产生信号。如按下 <Ctral + C> 组合键后将产生一个 SIGINT 信号。硬件异常产生信号：除数据、无效的存储访问等。这些事件通常由硬件 (如 :CPU) 检测到，并将其通知给 Linux 操作系统内核，然后内核生成相应的信号，并把信号发送给该事件发生时正在进行的程序。

2、软件方式

用户在终端下调用 kill 命令向进程发送任务信号。进程调用 kill 或 sigqueue 函数发送信号。

在 Linux 中，kill 命令用于发送信号给进程。可以使用kill --help 来查看关于kill 命令的帮助信息。信号列表如下：

```
shell#> kill -l

[root@schouse kevindba]# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
```

```
26) SIGVTALRM       27) SIGPROF       28) SIGWINCH      29) SIGIO      30) SIGPWR
31) SIGSYS    34) SIGRTMIN       35) SIGRTMIN+1     36) SIGRTMIN+2     37) SIGRTMIN+3
38) SIGRTMIN+4       39) SIGRTMIN+5     40) SIGRTMIN+6     41) SIGRTMIN+7     42) SIGRTMIN+8
43) SIGRTMIN+9       44) SIGRTMIN+10    45) SIGRTMIN+11    46) SIGRTMIN+12    47) SIGRTMIN+13
48) SIGRTMIN+14      49) SIGRTMIN+15    50) SIGRTMAX-14    51) SIGRTMAX-13    52) SIGRTMAX-12
53) SIGRTMAX-11      54) SIGRTMAX-10    55) SIGRTMAX-9     56) SIGRTMAX-8     57) SIGRTMAX-7
58) SIGRTMAX-6       59) SIGRTMAX-5     60) SIGRTMAX-4     61) SIGRTMAX-3     62) SIGRTMAX-2
63) SIGRTMAX-1       64) SIGRTMAX
```

信号量编号还是挺多的。但在 MySQL 日常故障中，常出现的信号量有 signal 6,7,8，11。这些信号量都会导致数据库服务崩溃。

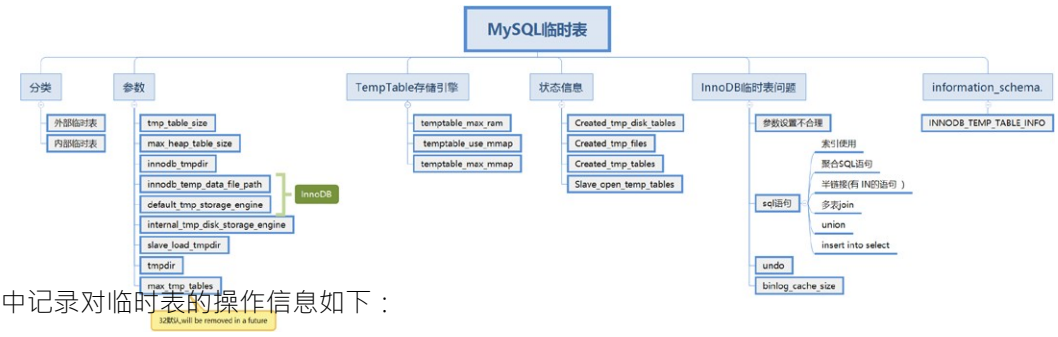
其中signal 11 最常见的。mysqld got signal 11 表示MySQL 服务器进程（mysqld）接收到了一个 SIGSEGV 信号，这是一个指示内存访问违规的错误。通常，这意味着 mysqld 尝试访问不属于它的内存区域，或者试图写入只读内存区域。

信号数值	信号名	说明
1	SIGHUP	当用户退出 Shell 时，由该 Shell 启的发所有进程都退接收到这个信号，默认动作为终止进程。
2	SIGINT	用户按下<Ctrl + C>组合键时，用户端时向正在运行中的由该终端启动的程序发出此信号。默认动作为终止进程
3	SIGQUIT	当用户按下<Ctrl + />组合键时产生该信号，用户终端向正在运行中的由该终端启动的程序发出此信号。默认动作为终止进程并产生 core 文件。
4	SIGILL	CPU 检测到某进程执行了非法指令。默认动作为终止进程并产生 core 文件
5	SIGTRAP	该信号由断点指令或其他 trap 指令产生。默认动作为终止进程并产生 core 文件。
6	SIGABRT	调用 abort 函数时产生该信号。默认动作为终止进程并产生 core 文件。
7	SIGBUS	非法访问内存地址，包括内存地址对齐（alignment）出错，默认动作为终止进程并产生 core 文件。
8	SIGFPE	在发生致命的算术错误时产生。不仅包括浮点运行错误，还包括溢出及除数为 0 等所有的算术错误。默认动作为终止进程并产生 core 文件。
9	SIGKILL	无条件终止进程。本信号不能被忽略、处理和阻塞。默认动作为终止进程。它向系统管理员提供了一种可以杀死任何进程的方法。
10	SIGUSR1	用户定义的信号，即程序可以在程序中定义并使用该信号。默认动作为终止进程。
11	SIGSEGV	指示进程进行了无效的内存访问。默认动作为终止进程并使用该信号。默认动作为终止进程。

备注：在 Linux 中查看信号量命令行是 ipcs

2）Temptable 相关处理信息

MySQL 临时表在很多场景中都会用到，比如用户自己创建的临时表用于保存临时数据，以及 MySQL 内部在执行复杂 SQL 时，需要借助临时表进行分组、排序、去重等操作。如图所示：



错误日志中记录对临时表的操作信息如下：

```
[Warning] [MY-011825] [InnoDB] Cannot add field `<temporary>_sup_Agreement_Type_280`
in table `tmp`.`#sql1b13_25d83b_1e` because after adding it, the row size is 8126
which is greater than maximum allowed size (8126) for a record on index leaf page.


/usr/sbin/mysqld(TemptableAggregateIterator::Init()+0xde) [0x12b71fe]
```

- tmp.#sql1b13_25d83b_1e：是对于 innodb_temp_tablespaces_dir 参数配置的临时文件。
- TempTableAggregateIterator：在 MySQL 实现中，属于内部迭代器或者执行计划中的一部分，用于处理临时表和聚合操作。这通常表明 MySQL 正在使用一种策略来处理复杂的聚合操作。特别是涉及到 GROUP BY 子句或需要进行聚合运算（如 SUM, COUNT, AVG 等）时，可能需要用到临时表来存储中间结果。
- greater than maximum allowed size (8126)：单行的数据记录，不操过8126 字节。在InnoDB 引擎下的限制。

3）WITH 表达式

MySQL8.0 支持公共表表达式common table expression（CTE）。CTE 是临时结果集，存在于单个语句的范围内，也可以在该语句中引用。在很多场景下，可以简化复杂的嵌套查询。CTE 类似于使用 CREATE[TEMPORARY] TABLE 创建的表，CTE 实际成本也可能受到结果集大小的影响。产生许多行的 CTE 可能需要一个足够大的内部临时表，以便从内存格式转换为磁盘格式，并且可能会受到性能损失。

从结合 WITH 语句触发和下面错误日志。可以确定临时结果集处理存在问题。

4） 临时缓存落盘机制

临时缓存落盘机制是 当会话临时表超过设置大小的时，就会落盘。（在 MySQL 8.0.15 及更早版本中，internal_tmp_disk_storage_engine 参数定义了用于磁盘上内部临时表的存储引擎。支持的存储引擎是 InnoDB 和 MyISAM。从 MySQL 8.0.16 开始，MySQL 只对磁盘上的内部临时表使用 InnoDB 存储引擎。不再支持 MYISAM 存储引擎用于此目的。）就是说临时表落磁盘，必定是 InnoDB 临时表引擎。那不想碰到如下问题，就需要保存在内存中。其中 internal_tmp_mem_storage_engine 决定内存中的引擎类型（MEMORY，INNODB 引擎，tmp_table_size 和 max_heap_table_size 控制内存临时表限制。

5） bug

搜索MySQL 的官方Bug 列表和社区论坛，查看是否有其他人报告了类似的问题。能提供一些分析思路和说明。

Showing all 12 (Edit, Save, CSV, Feed)

ID#	Date	Updated	Type	Status	Sev	Version	OS	CPU	Summary
102839	2021-03-06 11:03	2021-03-06 11:38	MySQL Server: DML	Verified (1199 days)	S2	8.0.23	Any	Any	Index out of bounds in regular expression search.
90257	2018-03-29 20:00	2018-04-03 23:52	MySQL Workbench	Verified (2267 days)	S3	6.3.0, 6.3.10	Windows	Any	Alter procedure with cte
91665	2018-07-16 13:45	2018-07-23 17:33	MySQL Server: Data Types	Verified (2160 days)	S3	8.0.11	Windows (Windows 7)	Other (x64)	CTE misbehaves with 38 digit keys
111511	2023-06-21 8:38	2024-04-12 15:00	MySQL Server: Optimizer	Verified (362 days)	S3	8.0.33	Ubuntu (22.04)	x86 (x86_64)	CTE missing scope in set queries
111852	2023-07-23 12:46	2023-07-27 1:15	MySQL Server: Optimizer	Verified (329 days)	S3	8.0.30, 8.0.34	Any	Any	An incorrect number of records is returned
113415	2023-12-14 6:27	2023-12-14 8:51	MySQL Server: DML	Verified (186 days)	S3	8.0, 8.0.35	Any	Any	Unknown column, but can get query result in CTE.
113429	2023-12-14 21:13	2023-12-18 13:33	MySQL Server: DML	Verified (182 days)	S5	8.0	Any	Any	Function call duplicated in CTE
102614	2021-02-17 2:28	2023-07-12 7:46	MySQL Workbench: SQL Editor	Verified (1216 days)	S3	8.0.23, 8.0.33	Linux (Debian 10.8 x86_64)	Any	CTE beautify/format does not work
103890	2021-06-02 13:47	2021-06-03 7:13	MySQL Server: Optimizer	Verified (1110 days)	S3	8.0.23, 8.0.25	Any	Any	wrong result when use cte with subquery
104127	2021-06-28 7:00	2021-06-28 7:19	MySQL Server: Optimizer	Verified (1085 days)	S2	8.0, 8.0.25	Any	Any	output count is unexpected when limit clause is used in CTE
111310	2023-06-07 6:51	2023-06-08 8:30	MySQL Server: Prepared statements	Verified (375 days)	S3	8.0.33	Ubuntu (22.04)	x86 (x86_64)	Prepared statement with CTE memory leak
111559	2023-06-26 9:10	2023-07-03 13:36	MySQL Server: Optimizer	Verified (357 days)	S3	8.0.33	Ubuntu (22.04)	x86 (x86_64)	View with CTE syntax error

分析结论

从上述分析中可以执行得到如下结论。在使用 WITH 表达式时，因为用到了大量的临时表，导致数据落盘，

临时表缓存在内存里使用 MEMORY 引擎，底层临时表引擎因为是 InnoDB 引擎，在处理过程中因为受到 InnoDB 引擎的限制导致崩溃情况发生。

当然解决这方面问题可以采取

1. 整改 WITH 语句，避免使用是临时表。
2. 同时可以更改临时表参数 tmp_table_size，max_heap_table_size 大小。

总结

在 MySQL 运维当中，碰到疑难杂时，需要综合使用多种方法和技术。通过收集和分析崩溃日志、复现和简化问题、检查查询、分析 MySQL 配置和状态、查找已知问题，以及联系支持或寻求帮助，能够逐步缩小问题范围并最终找到解决方案。
在 MySQL 的运维中，遇到复杂或棘手的问题时，确实需要综合运用多种方法和技术，逐步缩小问题范围并最终找到解决方案。重要的是保持耐心和细心，不要急于下结论或进行可能导致更多问题的操作。

经验：提高运维效率，MogDB/openGauss 中如何调用带变量的 SQL 脚本

文 / 罗海雄

前言

作为一个习惯于使用终端进行运维的 DBA, 手上肯定需要一批 SQL 脚本，用于日常的运维和优化工作。这些脚本中，往往需要用户输入一些变量。

相对上一个版本，传字符串形式有更新，更方便了。

那么，在 MogDB/openGauss 中，如何执行脚本？如何给脚本传递变量呢？

1. 执行不带变量的 SQL 脚本

不带变量的 SQL 脚本执行相对比较简单。我们分为两种情况讨论。一种是当前输入界面已经在 gsql 中，去调用 SQL 脚本；另一种是在 SHELL 里面，命令行调用 gsql 并给它传递 SQL 脚本。

1.1 准备一个测试用的 SQL 脚本（不含变量）

```
cat > sess.sql <<EOF
select pid,usename,application_name,xact_start,query_start,waiting,state
from pg_stat_activity
where state = 'active';
EOF
```

1.2 在 gsql 里面调用 SQL 脚本： \i 脚本文件名

gsql 提供一个元命令，\i , 用于调用 SQL 脚本。

比如：

```
gsql -r
\i sess.sql
      pid       | usename | application_name |          xact_start           |          query_start          |
waiting | state
----------------+---------+-----------------+-------------------------------+-------------------------------+---
------+--------
```

```
 22872621315840 | tpcc     | gsql             | 2024-07-10 15:40:21.179175+08 | 2024-07-10 15:40:21.179175+08 | f
| active
 22846861993728 | tpcc     | JobScheduler     |                               |                               | f
| active
```

\i 有几个变种，其中 \ir 是用于SQL 脚本中嵌套 \i 时，使用的路径是以被调用的 SQL 脚本为当前路径的。而 \i+和 \ir+ 则是允许脚本进行加密。关于脚本嵌套和脚本加密，这里就不展开了。

1.3 在 Shell 环境执行 gsql + SQL 脚本

在 Shell 环境执行 gsql + SQL 脚本，则有三种用法：

1.3.1 标准用法： -f 脚本文件名

最标准用法是使用 gsql 的命令行选项 -f, 比如

```
gsql -f sess.sql
      pid       | usename | application_name |          xact_start          |          query_start          | waiting | state
----------------+---------+------------------+------------------------------+------------------------------+---------+--------
 22873709213440 | tpcc    | gsql             | 2024-07-10 15:49:29.524565+08 | 2024-07-10 15:49:29.524565+08 | f       | active
 22846861993728 | tpcc    | JobScheduler     |                               |                               | f       | active
```

1.3.2 SHELL 输入重定向：< 脚本文件名

另一种用法是把 SQL 脚本中的内容模仿标准输入，传给 gsql 命令，也就是使用 输入重定向符号 < + 脚本文件名

如：

```
gsql < sess.sql
      pid       | usename | application_name |          xact_start          |          query_start          | waiting | state
----------------+---------+------------------+------------------------------+------------------------------+---------+--------
 22873349551872 | tpcc    | gsql             | 2024-07-10 15:56:06.868991+08 | 2024-07-10 15:56:06.868991+08 | f       | active
 22846861993728 | tpcc    | JobScheduler     |                               |                               | f       | active
```

1.3.3 利用 gsql -c 选项：-c “\i 脚本文件名”

这个其实利用 gsql 里面的 \i 元命令，其实是绕了一圈，用这种方法应该比较少。

比如：

```
gsql -c "\i sess.sql"
      pid       | usename | application_name |          xact_start          |          query_start          | waiting | state
----------------+---------+------------------+------------------------------+------------------------------+---------+--------
 22872307726080 | tpcc    | gsql             | 2024-07-10 16:00:04.077213+08 | 2024-07-10 16:00:04.077213+08 | f       | active
```

```
 22846861993728 | tpcc    | JobScheduler     |                          |                        | f       | active
```

执行带变量的 SQL 脚本

调用不带变量的 SQL 脚本还是比较简单的。下面进入困难模式，调用的 SQL 需要特定的变量。比如，刚才那个示例 SQL 脚本，输出的是 status='active', 可是，如果希望根据变量来输出不同 status 的结果，就会相对麻烦很多。

2.1 gsql 中的变量使用

幸运的是，gsql 的确是支持变量的。内部称为 "Variables"。

变量的设置有几种方式，分别是：

- gsql 命令行选项：-v 变量名称 = 变量值
- gsql 内赋值：\set 变量名称 变量值
- gsql 内交互式输入：\prompt 变量名称

而调用的时候，使用：变量名 形式进行调用。但是，这里面有个非常重要的点，：变量名 不能放在引号里面，或者说，：变量名 不能用引号括起来，因为对于 gsql 而言，无法判断引号里的冒号是字符串内容的一部分还是变量名的前缀，所以，统一当做字符串内容的一部分来处理。

这个处理逻辑和Oracle 不一样，Oracle 使用 & 符号来作为变量的前缀，在 sqlplus 里set define on（默认选项）时，无论 & 符号是否被放在引号里面，都会进行替换。

2.2 gsql 中的数字类型变量使用

如果需要传入的字符串，这个问题处理起来会很复杂，因此，我们循序渐进，先通过相对简单的传入数字的形式来熟悉变量的使用。
假设现在我们需要检查数据库中空间大小大于 x MB 的对象。
SQL 类似于这样：

```
cat > relsize.sql <<EOF

select relname,relpages*8192/1024/1024 relsize

from pg_class

where relpages * 8192/1024/1024 > :size

order by relsize desc;

EOF
```

2.2.1 gsql 命令行选项：-v 变量名称 = 变量值

从 SHELL 中调用

检查大于 10MB 的

```
gsql -v size=10 -f relsize.sql
                 relname                 |  relsize
-----------------------------------------+-----------
 snap_global_stat_all_indexes            | 45.5546875
 snap_global_statio_all_indexes          |    41.8125
 snap_class_vital_info                   |      40.25
 snap_global_config_settings             |   35.65625
 snap_global_stat_all_tables             |      33.25
 snap_summary_stat_all_indexes           | 32.4453125
 snap_summary_statio_all_indexes         | 31.7265625
 snap_global_statio_all_tables           | 26.1953125
 snap_summary_stat_all_tables            |   21.34375
 snap_summary_statio_all_tables          | 17.9921875
 snap_summary_transactions_running_xacts | 16.5703125
 snap_global_transactions_running_xacts  | 16.4921875
 snap_global_thread_wait_status          | 16.3828125
 pg_toast_2618                           |    13.6875
 snap_global_os_threads                  | 11.3359375
(15 rows)
```

检查大于 30MB 的

```
gsql -f relsize.sql -v size=30
             relname              |  relsize
----------------------------------+-----------
 snap_global_stat_all_indexes     | 45.5546875
 snap_global_statio_all_indexes   |    41.8125
 snap_class_vital_info            |      40.25
 snap_global_config_settings      |   35.65625
 snap_global_stat_all_tables      |      33.25
 snap_summary_stat_all_indexes    | 32.4453125
 snap_summary_statio_all_indexes | 31.7265625
```

1.1.1 gsql 内元命令：\set 变量名称 变量值

从 gsql 中调用，需先设置变量值

检查大于 10MB 的

```
gsql
\set size 10
\i relsize.sql
                 relname                 |  relsize
-----------------------------------------+------------
 snap_global_stat_all_indexes            | 45.5546875
 snap_global_statio_all_indexes          |    41.8125
 snap_class_vital_info                   |      40.25
 snap_global_config_settings             |   35.65625
 snap_global_stat_all_tables             |      33.25
 snap_summary_stat_all_indexes           | 32.4453125
 snap_summary_statio_all_indexes         | 31.7265625
 snap_global_statio_all_tables           | 26.1953125
 snap_summary_stat_all_tables            |   21.34375
 snap_summary_statio_all_tables          | 17.9921875
 snap_summary_transactions_running_xacts | 16.5703125
 snap_global_transactions_running_xacts  | 16.4921875
 snap_global_thread_wait_status          | 16.3828125
 pg_toast_2618                           |    13.6875
 snap_global_os_threads                  | 11.3359375
(15 rows)
```

30MB 同理，不占用篇幅。

2.2.2 gsql 内元命令等待输入 : \prompt 变量名称

检查大于 30MB 的

```
gsql
\prompt size
30
\i relsize.sql
             relname              |  relsize
----------------------------------+------------
 snap_global_stat_all_indexes     | 45.5546875
 snap_global_statio_all_indexes   |    41.8125
 snap_class_vital_info            |      40.25
 snap_global_config_settings      |   35.65625
 snap_global_stat_all_tables      |      33.25
```

```
 snap_summary_stat_all_indexes   | 32.4453125
 snap_summary_statio_all_indexes | 31.7265625
(7 rows)
```

10MB 同理，不占用篇幅。

2.2.3 让脚本包含输入提示 : \prompt 变量名称
除了以上 3 种，其实，还可以在 SQL 脚本中包含输入提示部分。
脚本可以写成 (给个新的名字 relsize_p.sql)

```
cat > relsize_p.sql <<EOF
\echo Please input value of size:
\prompt size
select relname,relpages*8192/1024/1024 relsize
from pg_class
where relpages * 8192/1024/1024 > :size
order by relsize desc;
EOF
```

然后可以 SHELL 调用 gsql

```
gsql -f relsize_p.sql
Please input value of size:
40
            relname             |  relsize
--------------------------------+------------
 snap_global_stat_all_indexes   | 45.5546875
 snap_global_statio_all_indexes |    41.8125
 snap_class_vital_info          |      40.25
(3 rows)
```

或者在 gsql 中调用该 SQL

```
gsql
\i relsize_p.sql
MogDB=# \i relsize_p.sql
Please input value of size:
41
            relname             |  relsize
```

```
--------------------------------+------------
 snap_global_stat_all_indexes   | 45.5546875
 snap_global_statio_all_indexes |    41.8125
(2 rows)
```

之所以特意把脚本改了个名字，是因为这是两种不同的思路，relsize.sql 是可以 SHELL 直接后台调用，无需交互的；而 relsize_p.sql 是强制交互的。

如果大家有需要准备常用带变量的脚本，建议准备两套，交互或者不交互。

2.3 gsql 中的字符串类型变量使用

字符串类型变量使用会复杂一些。

2.3.1 字符串变量的限制

在 gsql 的体系里面，变量是无法被引号括起来，下面做个不成功的尝试。

假设，我们需要检查某个数据库参数的值。计划使用类似这样的 SQL 进行查询

```
select name,setting
from pg_settings
 where name ~ ' 参数名称 '
order by name;
```

计划使用变量，按照 Oracle 的习惯，带变量的 SQL 会写成

```
select name,setting
from pg_settings
where name ~ ':name'
order by name;
```

尝试在 gsql 中跑一下

```
gsql -r
\set name index
select name,setting
from pg_settings
where name ~ ':name'
order by name;
```

返回的 0 行

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/327014155043010043