GBASE

GBase 8c AI 特性指南

www.gbase.cn

GBase 8c AI特性指南，南大通用数据技术股份有限公司

GBase 版权所有©2023，保留所有权利

版权声明

本文档所涉及的软件著作权及其他知识产权已依法进行了相关注册、登记，由南大通用数据技术股份有限公司合法拥有，受《中华人民共和国著作权法》、《计算机软件保护条例》、《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的南大通用数据技术股份有限公司的版权信息由南大通用数据技术股份有限公司合法拥有，受法律的保护，南大通用数据技术股份有限公司对本文档可能涉及到的非南大通用数据技术股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅，并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经南大通用数据技术股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将视为侵权，南大通用数据技术股份有限公司具有依法追究其责任的权利。

本文档中包含的信息如有更新，恕不另行通知。您对本文档的任何问题，可直接向南大通用数据技术股份有限公司告知或查询。

未经本公司明确授予的任何权利均予保留。

通讯方式

南大通用数据技术股份有限公司

天津市高新区华苑产业园区开华道22号普天创新园东塔20-23层(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

商标声明

GBASE® 是南大通用数据技术股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由南大通用数据技术股份有限公司合法拥有，受法律保护。未经南大通用数据技术股份有限公司书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯南大通用数据技术股份有限公司商标权的，南大通用数据技术股份有限公司将依法追究其法律责任。

目　录

目　录 II
1　AI4DB: 数据库自治运维 6
1.1　DBMind 模式说明 8
1.1.1　service 子命令 9
1.1.2　component 子命令 15
1.1.3　set 子命令 16
1.2　DBMind 的支持组件 17
1.2.1　Prometheus Exporter 组件 17
1.2.1.1　概述 17
1.2.1.2　环境部署 17
1.2.1.3　使用指导 18
1.2.1.4　获取帮助 20
1.2.1.5　命令参考 20
1.2.1.6　常见问题处理 24
1.3　DBMind 的 AI 子功能 25
1.3.1　X-Tuner: 参数调优与诊断 25
1.3.1.1　概述 25
1.3.1.2　使用准备 25
1.3.1.3　使用示例 30
1.3.1.4　获取帮助 35
1.3.1.5　命令参考 36
1.3.1.6　常见问题处理 39
1.3.2　Index-advisor: 索引推荐 39
1.3.2.1　单 query 索引推荐 39

1.3.2.2 虚拟索引 41
1.3.2.3 workload 级别索引推荐 43
1.3.3 AI4DB: 慢 SQL 根因分析 46
1.3.3.1 概述 46
1.3.3.2 环境部署 46
1.3.3.3 使用指导 47
1.3.3.4 获取帮助 47
1.3.3.5 命令参考 48
1.3.3.6 常见问题处理 48
1.3.4 AI4DB: 趋势预测 48
1.3.4.1 概述 48
1.3.4.2 环境部署 49
1.3.4.3 使用指导 49
1.3.4.4 获取帮助 49
1.3.4.5 命令参考 50
1.3.4.6 常见问题处理 50
1.3.5 SQLdiag: 慢 SQL 发现 50
1.3.5.1 概述 51
1.3.5.2 使用指导 51
1.3.5.3 获取帮助 53
1.3.5.4 命令参考 54
1.3.5.5 常见问题处理 54
1.3.6 SQL Rewriter: SQL 语句改写 54
1.3.6.1 概述 54
1.3.6.2 使用指导 55
1.3.6.3 获取帮助 56

1.3.6.4 命令参考 57
1.3.6.5 常见问题处理 57
1.3.7 Anomaly detection: 异常检测 57
1.3.7.1 概述 57
1.3.7.2 使用指导 57
1.3.7.3 获取帮助 59
1.3.7.4 命令参考 59
1.3.7.5 常见问题处理 60
1.3.8 Anomaly analysis: 多指标关联 61
1.3.8.1 概述 61
1.3.8.2 使用指导 61
1.3.8.3 获取帮助 61
1.3.8.4 命令参考 62
1.3.8.5 常见问题处理 63
1.4 ABO 优化器 63
1.4.1 智能基数估计 63
1.4.1.1 概述 63
1.4.1.2 前置条件 63
1.4.1.3 使用指导 63
1.4.1.4 最佳实践 64
1.4.1.5 常见问题处理 66
1.4.2 自适应计划选择 66
1.4.2.1 概述 66
1.4.2.2 前置条件 66
1.4.2.3 使用指导 66
1.4.2.4 最佳实践 67

1.4.2.5 常见问题处理 68
2 DB4AI：数据库驱动 AI 68
2.1 原生 DB4AI 引擎 68
2.2 全流程 AI 80
2.2.1 PLPython Fenced 模式 80
2.2.2 DB4AI-Snapshots 数据版本管理 82
2.2.3 DB4AI-Query：模型训练和推断 88

AI 特性

人工智能技术最早可以追溯到上世纪 50 年代，甚至比数据库系统的发展历史还要悠久。但是，由于各种各样客观因素的制约，在很长的一段时间内，人工智能技术并没有得到大规模的应用，甚至还经历了几次明显的低谷期。到了近些年，随着信息技术的进一步发展，从前限制人工智能发展的因素已经逐渐减弱，所谓的 ABC（AI、Big data、Cloud computing）技术也随之而诞生。

AI 与数据库结合是近些年的行业研究热点，GBase 8c 参与了该领域的探索，并取得了阶段性的成果。AI 特性子模块名为 DBMind，相对数据库其他功能更为独立，大致可分为 AI4DB、DB4AI 和 ABO 优化器三个部分。

- AI4DB 就是指用人工智能技术优化数据库的性能，从而获得更好地执行表现；也可以通过人工智能的手段实现自治、免运维等。主要包括自调优、自诊断、自安全、自运维、自愈等子领域；
- DB4AI 就是指打通数据库到人工智能应用的端到端流程，通过数据库来驱动 AI 任务，统一人工智能技术栈，达到开箱即用、高性能、节约成本等目的。例如通过 SQL-like 语句实现推荐系统、图像检索、时序预测等功能，充分发挥数据库的高并行、列存储等优势，既可以避免数据和碎片化存储的代价，又可以避免因信息泄漏造成的安全风险；

本章节所涉及的功能独立存在于数据库安装目录($GAUSSHOME)的 bin/dbmind 目录中，各个子功能存在于 dbmind 的子目录 components 中。提供 gs_dbmind 命令行供用户调用。与此同时，对于数据库内置 AI 的功能（如 DB4AI），以 SQL 语法和系统函数的形式呈现。

1 AI4DB：数据库自治运维

如上文所述，AI4DB 主要用于对数据库进行自治运维和管理，从而帮助数据库运维人 员减少运维工作量。在实现上，DBMind 的 AI4DB 框架具有监控和服务化的性质，同时也提供即时 AI 工具包，提供开箱即用的 AI 运维功能（如索引推荐）。AI4DB 的监控平台以开源的 Prometheus 为主，DBMind 提供监控数据生产者 exporter，可与 Prometheus 平台完成对接。DBMind 的 AI4DB 服务架构如下图所示：

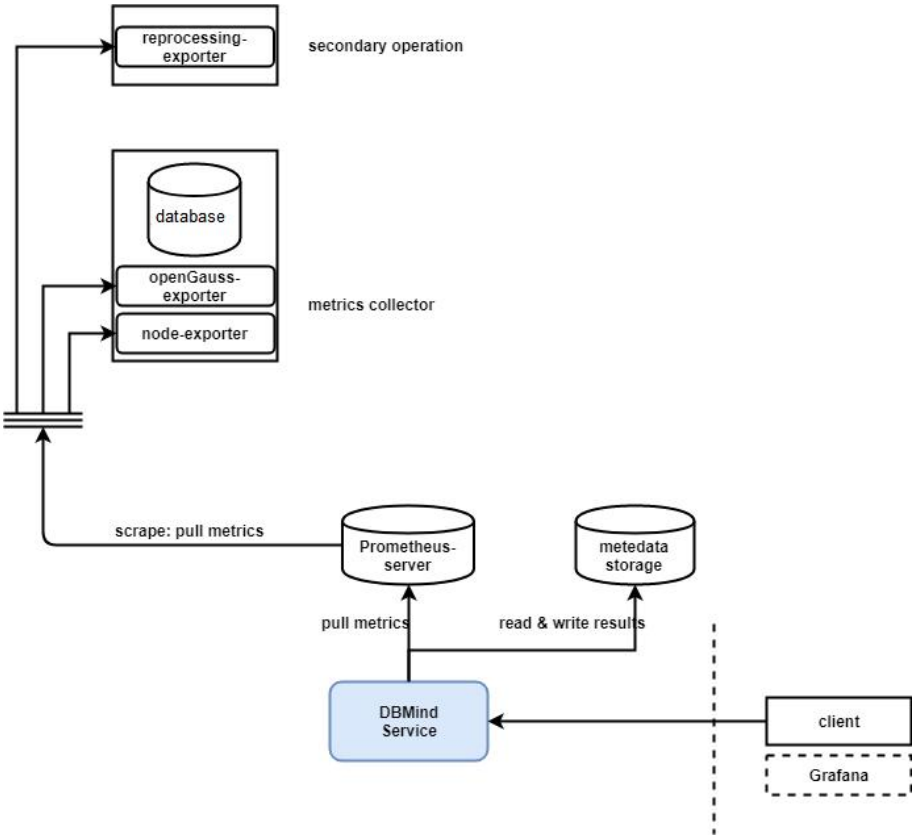

图 1-1 DBMind AI4DB 服务架构

图中各关键组件说明：

- DBMind Service：DBMind 后台服务，可用于定期离线计算，包括慢 SQL 根因分析、时序预测等；
- Prometheus-server：Prometheus 监控指标存储的服务器；
- metadatabase：DBMind 在离线计算结束后，将计算结果存储在此处，支持 GBase 8c、GBase 8c、SQLite 等数据库；
- client：用户读取 DBMind 离线计算结果的客户端，目前仅实现命令行客户端；若采用 GBase 8c 等数据库存储计算 DBMind 计算结果，则用户可以自行配置 Grafana 等可视化工具对该结果进行可视化；
- GBase 8c-exporter：用户从数据库节点上采集监控指标，供 DBMind 服务进行计算；
- node-exporter：Prometheus 官方提供的 exporter，可用于监控该节点的系统指标，如 CPU 和内存使用情况；
- reprocessing-exporter：用于对 Prometheus 采集到的指标进行二次加工处理，例如计

算 CPU 使用率等。

环境配置

DBMind 外置 AI 功能需要运行在 Python 3.6 版本及以上，需要的第三方依赖包记录在 AI 功能根目录（$GAUSSHOME/bin/dbmind）的 requirements.txt 文件中（包括 requirements-x86.txt 与 requirements-arrch64.txt,用户可根据自己平台类型选择）中，可以通过 pip install 命令安装依赖，如：

```
pip install requirements-x86.txt
```

如果用户没有安装齐全所需的依赖，则当用户执行 gs_dbmind 命令时，会再次提醒用户安装第三方依赖。需要注意，该文件提供了 DBMind 所需的第三方依赖，若用户环境存在第三方包冲突等情况，可由用户根据实际情况进行处理。

1.1 DBMind 模式说明

用户可通过 gs_dbmind 命令调用 AI4DB 的全部功能，该命令可实现下列基本功能：

- 服务功能：service 子命令，包括创建并初始化配置目录、启动后台服务、关闭后台服务等；
- 调用组件：component 子命令，AI4DB 功能（如索引推荐、参数调优等）可通过该模式进行即时调用；
- 设置参数：set 子命令，通过该命令，可以一键修改配置目录中的配置文件值；当然，用户也可以通过文本编辑器进行手动修改；

用户可以通过 --help 选项获得上述模式的帮助信息，例如：

```
gs_dbmind –help

usage: [-h] [--version] {service,set,component} ...

GBase 8c DBMind: An autonomous platform for GBase 8c

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

available subcgbaseands:
  {service,set,component}
                        type '<subcgbaseand> -h' for help on a specific subcgbaseand
```

service	send a cgbaseand to DBMind to change the status of the service
set	set a parameter
component	pass cgbaseand line arguments to each sub-component.

表 1-1 gs_dbmind 选项基本说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
--version	版本号	-
service	服务功能相关的子命令	-
component	调用组件的子命令	-
set	修改配置文件的子命令	-

1.1.1service 子命令

该子命令可用于对配置目录进行初始化，同时也可以实现启动和停止后台任务。

配置目录初始化

用户可通过 gs_dbmind service setup 子命令进行配置目录的初始化。该配置文件中可包括 DBMind 的配置文件、日志等内容。该目录中的部分文件说明：

- dbmind.conf：DBMind 的参数配置文件，用户可通过 gs_dbmind set 命令进行修改，也可通过文本编辑器进行手动修改。
- dynamic_config.db：DBMind 服务保存在本地节点的元信息，主要包括算法的超参数、监控阈值等；该文件为 DBMind 服务元信息，不可由用户直接配置。
- metric_map.conf：监控指标映射表，可用于适配到不同采集平台中。例如，在 DBMind 中，监控到的系统 cpu 使用率名为 os_cpu_usage，而用户自行实现的指标采集工具将 cpu 使用率命名为 my_cpu_usage_rate. 则在该种情况下，如果想要 DBMind 代表 cpu 使用率的指标名为 my_cpu_usage_rate，则需要修改该配置选项。即添加“os_cpu_usage = my_cpu_usage_rate”配置项进行映射。对于普通用户，建议直接使用 DBMind 配套的采集组件和方案，则不涉及修改该配置文件。
- logs：该目录中用于存储 DBMind 服务产生的日志。用户可通过两种方式进行配置目录

的初始化，一种为交互式，另一种为非交互式。例如，待初始化的配置目录名为 confpath，则分别通过下述方法进行配置：

- 交互式模式

```
gs_dbmind service setup -c confpath --interactive
```

执行完毕上述命令后，用户可通过命令行终端对配置项进行交互式配置。

- 非交互式模式

非交互式模式总共分为三个步骤，即启动配置，修改配置项，初始化配置。其中第二个步骤需要用户通过文本编辑器手动编辑配置文件。具体步骤如下：

步骤 1 启动配置，执行下述命令：

```
gs_dbmind service setup -c confpath
```

步骤 2 执行完上述命令后，会在 confpath 目录下生成 dbmind.conf 配置文件，用户需要利用文本编辑器进行手动修改。需要修改的配置部分为“指标数据源数据库信息区【TSDB】”、“预测结果存储数据库信息区【METADATABASE】”和“自监控参数配置区【SELF-MONITORING】”相关参数的说明如下：

```
# TSDB 部分用于指定监控数据库系统的指标存储位置，目前只支持 Prometheus.
# 此处，必填项为 Prometheus 的 IP 地址和端口号，其他选项（如 username, password, ssl 证书信息）取决于用户配置，非必须。
[TSDB]
name = prometheus # The type of time-series database. Options: prometheus.
host = # Address of time-series database.
port = # Port to connect to time-series database.
username = (null) # User name to connect to time-series database.
password = (null) # Password to connect to time-series database.
ssl_certfile = (null) # The certificate file for ssl connections.
ssl_keyfile = (null) # Certificate private key file.
ssl_keyfile_password = (null) # Password for ssl keyfile.
ssl_ca_file = (null)    # CA certificate to validate requests.

# METADATABASE 部分用于指定 DBMind 生成的分析结果的存储位置。
# 当前支持的数据库类型有 SQLite, GBase 8c、GBase 8c 以及 PostgreSQL. 如果使用 GBase 8c 数据库的话，注意 Python 驱动 psycopg2 的兼容性问题，用户可以选择使用 GBase 8c 官方提供的驱动，也可以通过自行编译或修改 GUC 参数进行适配。
# 其他信息为连接到该数据库的连接信息，注意用户需要有数据库创建权限。
[METADATABASE]
dbtype = sqlite # Database type. Options: sqlite,GBase 8c, GBase 8c, postgresql.
```

```
host = # Address of meta-data database.
port = # Port to connect to meta-data database.
username = # User name to connect to meta-data database.
password = (null) # Password to connect to meta-data database.
database = # Database name to connect to meta-data database.

# WORKER 用于指定 DBMind 可以使用的 worker 子进程数量，如果写 0 则会进行自适应，即尽可能多地使用 CPU 资源。
[WORKER]
process_num = 0    # Number of worker processes on a local node. Less than or equal to zero means adaptive.

# AGENT 部分用于指定 DBMind 连接到 GBase 8c Agent 的信息。通过使用该 Agent，可以让 DBMind 获取到被监控实例的即时状态，从而提高分析准确性。同时，也可以向数据库实例下发一些变更动作，如结束某条慢 SQL 语句（这取决于此处配置的用户是否有足够的权限）。
# 该 master_url 地址即为 Agent 的地址，由于 GBase 8c-exporter 承担了 Agent 的角色，故改地址也就是 GBase 8c-exporter 的地址。
# 同时，GBase 8c-exporter 是支持 Https 协议的，所以，此处也可以根据配置指定 SSL 证书。
[AGENT]
master_url =    # The agent URL of the master node. e.g., https://127.0.0.1:9187.
username = # Username to login the monitoring database. Credential for agent.
password = # Password to login the monitoring database. Credential for agent.
ssl_certfile = (null) # The certificate file for ssl connections.
ssl_keyfile = (null) # Certificate private key file.
ssl_keyfile_password = (null) # Password for ssl keyfile.
ssl_ca_file = (null)    # CA certificate to validate requests.

# SELF-MONITORING 表示“自监控”配置，用于配置监控数据库实例时的参数。
# detection_interval 表示周期性检查任务的执行频次，单位是秒；
# last_detection_time 表示每一次检查任务使用的最近数据长度；
# forecasting_future_time 表示时序预测特性预测未来时间的长度；
# golden_kpi 表示“黄金 KPI”，即重点关注的监控指标；
# result_storage_retention 表示诊断结果的最长保存时间。
[SELF-MONITORING]
detection_interval = 600    # Unit is second. The interval for performing health examination on the GBase 8c through monitoring metrics.
last_detection_time = 600    # Unit is second. The time for last detection.
forecasting_future_time = 3600    # Unit is second. How long the KPI in the future for forecasting. Meanwhile, this is the period for the forecast.
```

```
# The following golden_kpi of monitoring system is vital.
golden_kpi = os_cpu_usage, os_mem_usage, os_disk_usage, gaussdb_qps_by_instance    # DBMind only measures and detects the golden metrics in the anomaly detection processing.
result_storage_retention = 604800    # Unit is second. How long should the results retain? If retention is more than the threshold, DBMind will delete them.

# SELF-OPTIMIZATION 表示“自优化”配置，可以修改下述参数配置，对 DBMind 的优化结果进行干预，一般使用默认值即可。
# optimization_interval 的优化任务执行间隔；
# max_reserved_period 优化结果的最大保存时间；
# max_index_num 索引建议结果上限；
# max_index_storage 推荐出的索引的页面占用磁盘空间的上限；
# max_template_num 索引推荐使用的 SQL 模板记录 SQL 语句的上限；
# kill_slow_query 是否启动慢 SQL 自动查杀，如果启动慢 SQL 自动查杀，可以通过 set 子命令设置查杀阈值，例如将查杀阈值设置为 70 秒，该值应为正整数，单位是秒：
# gs_dbmind set slow_sql_threshold max_elapsed_time 70
[SELF-OPTIMIZATION]
optimization_interval = 86400    # Unit is second. The interval for generating report.
max_reserved_period = 100 # Unit is day. Maximum retention time.
max_index_num = 10 # Maximum number of advised indexes.
max_index_storage = 100 # Unit is MB.
max_template_num = 5000 # Maximum number of templates.
kill_slow_query = false    # Whether to actively check and kill slow query. The default elapsed time of a slow query to be killed is 1 minute.

# LOG 表示设置 DMBind 的日志记录信息。
[LOG]
maxbytes = 10485760 # Default is 10Mb. Maximum size of a single log file. If maxbytes is zero, the file grows indefinitely.
backupcount = 1 # Number of backups of log files.
level = INFO    # Options: DEBUG, INFO, WARNING, ERROR.

#  下列内容表示给用户进行交互配置时的提示信息，用户无需配置。
[CgbaseENT]
worker = The form of executing compute-intensive tasks. Tasks can be executed locally or distributed to multiple nodes for execution.
tsdb = Configure the data source for time series data, which come from monitoring the GBase 8c instance.
metadatabase = Configure the database to record meta-data, which the database can store meta-data for the forecasting and diagnosis process. The database should be an GBase 8c instance.
```

```
self-monitoring = Set up parameters for monitoring and diagnosing GBase 8c instance.
self-optimization = Set up parameters for GBase 8c optimization.
```

步骤 3 待用户手动修改完上述参数后，需要执行下述命令进行配置项的初始化。在该阶段中，DBMind 会初步检查配置项的正确性、加密配置项中出现的明文密码、同时初始化用于存储结果数据的元数据库。

```
gs_dbmind service setup --initialize -c confpath
```

步骤 4 完成配置目录初始化过程，可基于该配置目录启动 DBMind 后台服务。

----结束

说明

(1) 配置文件注释信息用于在交互模式下对用户进行提示，有特殊含义不要手动修改或删除；

(2) 需要确保配置项的值与注释信息之间通过空格符分割，否则系统会将注释信息识别为配置项的值；

(3) 配置项中的特殊字符，如果需要转移，则通过转义符“百分号”（%）来转义，例如，用户配置的密码为 "password%"，则应通过“百分号”进行转义，即 "password%%"。

启动服务

当用户完成配置目录的初始化后，可基于此配置目录启动 DBMind 后台服务。例如配置目录为 confpath，则启动命令如下：

```
gs_dbmind service start -c confpath
```

当执行上述命令后，会提示服务已启动。在未指定任何附加参数时，该命令默认会启动所有的后台任务。如果用户只想启动某一个后台任务，需要添加参数 --only-run. 例如，用户只想启动慢 SQL 根因分析服务，则为：

```
gs_dbmind service start -c confpath --only-run slow_query_diagnosis
```

关闭服务

关闭服务与启动服务类似，其命令行结构更加简单，只需指定配置目录的地址即可。例如配置目录为 confpath，则为：

```
gs_dbmind service stop -c confpath
```

DBMind 服务会在后台执行完正在运行的任务后自行退出。

⚠ 注意

(1) [METADATABASE]中的元数据库用户需要具有在该数据库下的创表和数据插入更新权限，否则工具执行会出现异常。

(2) 当前不支持同一配置文件下分开启动多个服务。

(3) 工具提供了 requirement.txt 文件，用户可以通过该文件安装所需的第三方依赖。

命令参考

用户可以通过 --help 选项获得该模式的帮助信息，例如：

```
gs_dbmind service --help
usage: service [-h] -c DIRECTORY [--only-run {slow_query_diagnosis,forecast}] [--interactive
| --initialize]
{setup,start,stop}
positional arguments:
{setup,start,stop}
perform an action for service
optional arguments:
-h, --help       show this help message and exit
-c DIRECTORY, --conf DIRECTORY
set the directory of configuration files
--only-run {slow_query_diagnosis,forecast}
explicitly set a certain task running in the backend
--interactive   configure and initialize with interactive mode
--initialize     initialize and check configurations after configuring.
```

表 1-2 gs_dbmind service 子命令说明

参数	参数说明	取值范围
action	动作参数	setup：初始化配置目录。 start：服务启动。 stop：服务停止。
-c，--conf	配置文件目录地址	——
--initialize	配置参数初始化	——

--interactive	交互式输入配置参数	——
--only-run	选择只运行的模块	forecast：预测模块。 slow_query_diagnosis ： 慢 SQL 根因分析模块。
-h, --help	帮助命令	——

1.1.2component 子命令

该子命令可以用于启动 DBMind 的组件，包括可用于监控指标的 exporter，以及 AI 功能等。该命令可以将用户通过命令行传入的命令转发给对应的子组件，故不同的子组件命令需参考其功能的对应说明，详见后文各个子组件对应章节，此处不再赘述。

命令参考

用户可以通过 --help 选项获得该模式的帮助信息，例如：

```
gs_dbmind component --help
usage: component [-h] COMPONENT_NAME ...
positional arguments:
COMPONENT_NAME choice a component to start. ['extract_log', 'forecast', 'index_advisor',
'GBase 8c_exporter', 'reprocessing_exporter', 'slow_query_diagnosis', 'sqldiag', 'xtuner']
ARGS    arguments for the component to start
optional arguments:
-h, --help      show this help message and exit
```

表 1-3 gs_dbmind component 子命令说明

参数	参数说明	取值范围
COMPONENT_NAME	子组件名	extract_log, forecast, index_advisor, GBase 8c_exporter, reprocessing_exporter, slow_query_diagnosis, sqldiag, xtuner
ARGS	子组件的参数	参考子组件的命令说明

-h, --help	帮助命令	-

1.1.3set 子命令

该命令用于修改配置文件 dbmind.conf 中的参数值，与用户手动修改配置文件 dbmind.conf 一般无差异。例如修改配置目录 confpath 中的配置文件 dbmind.conf 中 TSDB 配置部分，host 参数的值，并将其设置为 127.0.0.1。则可通过下述命令实现：

```
gs_dbmind set TSDB host 127.0.0.1 -c confpath
```

在修改上述普通参数时，与手动修改配置文件 dbmind.conf 无差异。但是，当用户想要修改密码项时，则必须通过该命令才可以实现修改，否则通过用户手动修改的配置文件无法实现对明文密码的加密，例如：

```
gs_dbmind set METADATABASE password xxxxxx -c confpath
```

说明

该命令对于字符串是大小写敏感的，如果输错则可能出现执行过程错误。

命令参考

用户可以通过 --help 选项获得该模式的帮助信息，例如：

```
gs_dbmind set --help
usage: set [-h] -c DIRECTORY section option target
positional arguments:
section   which section (case sensitive) to set option    which option to set
target    the parameter target to set
optional arguments:
-h, --help       show this help message and exit
-c DIRECTORY, --conf DIRECTORY
set the directory of configuration files
```

表 1-4 模块命令行参数说明：python dbmind/ set xxx

参数	参数说明	取值范围
-h, –help	帮助命令	-
-c，–conf	配置文件目录 confpath	-
section	设置区	-

option	设置项	-
target	设置值	-

1.2 DBMind 的支持组件

支持组件是指 DBMind 提供的用于支撑整个服务、解决方案能够部署和实施的模块。它们本身不是 AI 功能，却是整个服务体系中非常重要的一环，用于支撑整个自治运维解决方案的快速实施，如用于采集数据库指标的 exporter 等。

1.2.1Prometheus Exporter 组件

1.2.1.1 概述

Prometheus 是业内非常流行的开源监控系统，同时本身也是一款时序数据库。Prometheus 的采集端称之为 exporter，用来收集被监控模块的指标项。为了与 Prometheus 平台完成对接，AI 工具分别实现了两款 exporter，分别是用来采集数据库指标的 GBase 8c-exporter，以及对采集到的指标进行二次加工的 reprocessing- exporter。

须知

- Prometheus 和 exporter 是业内流行的监控和采集平台，部署在内网环境中，不对外部暴露接口，仅供内部监控平台使用。因此，为了增强该平台的安全性，一般需要用户或运维人员配置防火墙等，以便隔离外部访问，从而增强监控平台的安全性。
- Prometheus 平台在默认情况下，采用 Http 协议、并且没有任何安全访问限制。这是因为，该平台一般部署在内网环境中，攻击风险可控。如果用户希望提高安全性，可自行修改 Prometheus 的 TLS 配置选项，但仍不建议对外部直接暴露访问接口。

1.2.1.2 环境部署

步骤 1 通过命令行参数启动对应的 exporter 进程，同时在本地机器创建侦听端口号。

步骤 2 在 Promethues 的服务器端修改配置文件 prometheus.yml，将启动的 exporter 信息添加进去，例如：

```
scrape_configs:
...
- job_name: 'GBase 8c_exporter' static_configs:
```

```
- targets: ['127.0.0.1:9187']
...
```

----结束

其中，提供的 exporter 组件默认采用 Https 通信协议，因此需要用户默认提供 ssl 证书和秘钥文件，并通过--ssl-keyfile 以及--ssl-certfile 提供。若用户不希望使用 Https 协议， 则可以通过--disable-https 选项禁用该模式。

说明

由于 GBase 8c 默认模式下的通信加密协议与 PostgreSQL 不兼容，导致通过 PyPI 源安装的 psycopg2-binary，由于是基于 PostgreSQL 编译的 Python 驱动，而默认无法连接至数据库。

1.2.1.3 使用指导

用户可通过 gs_dbmind 命令启动对应的 exporter。下面为用户演示一个完整的 Prometheus 监控平台的搭建过程。

步骤 1 部署 GBase 8c-exporter

启动 GBase 8c-exporter，采用默认侦听端口号 9187，侦听地址为 192.168.1.100，不采用 https 协议，则命令可以为：

```
gs_dbmind component GBase 8c_exporter --url postgresql://user:password@ip:port/dbname
--web.listen- address 192.168.1.100 --disable-https
```

步骤 2 部署 reprocessing-exporter

启动 reprocessing-exporter，采用默认侦听端口号 8181，侦听地址为 192.168.1.101，Prometheus-server IP 与端口号为 192.168.1.100:9090，采用 https 协议，则命令可以为：

```
gs_dbmind component reprocessing_exporter 192.168.1.100 9090 --web.listen-address
192.168.1.101 --ssl- keyfile server.key --ssl-certfile server.crt
```

步骤 3 部署 node-exporter

一般地，对于 Prometheus 监控平台都需要部署 node-exporter 用于监控 Linux 操作系统，后文提到的部分 AI 功能也需要依赖 node-exporter 采集 Linux 系统指标，故也需要用户来部署；只不过，该 node-exporter 为 Prometheus 自带组件，

需要用户自行下载安装，下载地址为 https://prometheus.io/download/#node_exporter，

使用方法详见：https://prometheus.io/docs/guides/node-exporter/#installing-and-running-the-node-exporter。

用户可解压压缩包后直接运行该 node-exporter 进程，其默认端口号为 9100，启动命令行为：

```
./node_exporter
```

步骤 4 配置 Prometheus-server，修改配置文件 prometheus.yml，添加下述内容：

```
scrape_configs:
...
job_name: 'GBase 8c_exporter' static_configs:
- targets: ['192.168.1.100:9187']
job_name: 'reprocessing_exporter' scheme: https
tls_config: ca_file: xxx.crt
static_configs:
- targets: ['127.0.0.1:8181']
job_name: 'node_exporter' static_configs:
- targets: ['127.0.0.1:9100']
...
```

----结束

注意

- GBase 8c-exporter 中连接数据库的用户需要 monitor admin 或以上权限，否则会出现部分指标无法采集的情况。
- GBase 8c-exporter 会从 dbe_perf.statement_history 中抽样慢 SQL 信息，dbe_perf.statement_history 视图慢 SQL 记录与 GUC 参数 log_min_duration_statement 和 track_stmnt_stat_level 相关，其中 log_min_duration_statement 是慢 SQL 阈值，单位毫秒，具体值由用户设置；track_stmnt_stat_level 是 SQL 记录级别，默认为'OFF,L0'，即只记录慢 SQL 信息，级别为 L0，用户在详细了解参数意义与作用情况下谨慎修改。
- GBase 8c-exporter 采集数据库相关信息，主要包括部分系统表和视图中的数据（具体参见代码中 GBase 8c_exporter 中的配置文件），node-exporter 采集系统 指标信息，主要与系统磁盘、CPU 等相关，reprocessing_exporter 基于 prometheus-server 中的某些指标（具体参见代码中 reprocessing_exporter 中的配置文件）进行二次加工，最

终提供加工后的数据供用户使用。

- prometheus-server 在拉取 exporter 数据时有超时机制，超时时间由 scrape_timeout（默认 10s）控制，因此当 exporter 采集数据量较大时，用户可根据实际情况增大 scrape_timeout 以防止超时报错,另外需要注意的是 scrape_interval(采集间隔) 不能比 scrape_timeout 小，否则会出现异常。
- 如果数据库时区设置和系统不相同,可能会出现时间相关指标时间与系统时间不一致的情况，因此需要将用户时区与系统保持同步。
- 当使用 https 通信时，工具会检测证书与密钥文件权限以及证书有效期，如果文件权限大于 600 则会出现报警，证书有效期小于 90 天会出现报警。
- 当存在指标重复采集时，GBase 8c-exporter 会出现异常，异常信息会打印到日志中。
- GBase 8c-exporter 的 --config 、 --disable-settings-metrics 、 --disable-statement-history-metrics 三个参数需要注意，其存在以下几种情况：
    - 用户不指定其中任何参数,则工具会同时对 yamls 目录下的三个配置文件中的指标进行采集。
    - 用户显式指定--config,则工具不会采集 yamls 目录下 default.yml 中的指标，而会采集用户指定配置文件中的指标，同时 pg_settings.yml 和 statements.yml 正常 采集，此时用户需要注意指定的配置文件中的指标和 pg_settings.yml、statements.yml 中的指标不能存在重复采集的现象。
    - 用户显式指定--disable-settings-metrics,则工具不会采集 yamls 目录下 pg_settings.yml 中的指标，用户显式指定--disable-statement-history-metrics,则工具不会采集 yamls 目录下 statements.yml (慢 SQL 相关) 中的指标。

1.2.1.4 获取帮助

用户可以通过--help 命令获取帮助信息，例如：

```
gs_dbmind component GBase 8c_exporter --help gs_dbmind component reprocessing_exporter --help
```

1.2.1.5 命令参考

reprocessing-exporter 的使用帮助详情：

```
gs_dbmind component reprocessing_exporter --help usage: [-h] [--disable-https] [--ssl-keyfile SSL_KEYFILE]
```

```
[--ssl-certfile SSL_CERTFILE]
[--web.listen-address WEB.LISTEN_ADDRESS] [--web.listen-port WEB.LISTEN_PORT]
[--collector.config COLLECTOR.CONFIG] [--log.filepath LOG.FILEPATH] [--log.level
{debug,info,warn,error,fatal}] [--version]
prometheus_host prometheus_port
Reprocessing Exporter: A re-processing module for metrics stored in the Prometheus server.
positional arguments:
prometheus_host   from which host to pull data prometheus_port      the port to connect to the
Prometheus host
optional arguments:
-h, --help       show this help message and exit
--disable-https      disable Https schema
--ssl-keyfile SSL_KEYFILE
set the path of ssl key file
--ssl-certfile SSL_CERTFILE
set the path of ssl certificate file
--web.listen-address WEB.LISTEN_ADDRESS
address on which to expose metrics and web interface
--web.listen-port WEB.LISTEN_PORT
listen port to expose metrics and web interface
--collector.config COLLECTOR.CONFIG
according to the content of the yaml file for metric collection
--log.filepath LOG.FILEPATH
the path to log
--log.level {debug,info,warn,error,fatal}
only log messages with the given severity or above. Valid levels: [debug, info, warn, error, fatal]
--version       show program's version number and exit
```

表 1-5 reprocessing-exporter 的命令行参数详情表

参数	参数说明	取值范围
prometheus_host	Prometheus-server 的 IP 地址	-
prometheus_port	Prometheus-server 的服务侦听端口号	1024-65535
-h, –help	帮助选项	-
–disable-https	禁用 Https 协议	-
–ssl-keyfile	Https 协议使用的证书私钥文件路径	-

–ssl-certfile	Https 协议使用的证书文件路径	-
–web.listen-address	该 exporter 服务的绑定 IP	-
–web.listen-port	该 exporter 服务的侦听端口	1024-65535
–collector.config	显性指定的待采集指标配置文件路径	-
–log.filepath	日志文件保存路径，默认保存在当前目录下	-
–log.level	日志文件的打印级别，默认为 INFO 级别	debug，info，warn，error, fatal
–version	显示版本信息	-

GBase 8c-exporter 的使用帮助详情：

```
gs_dbmind component GBase 8c_exporter --help
usage: [-h] --url URL [--config CONFIG] [--constant-labels CONSTANT_LABELS]
[--web.listen-address WEB.LISTEN_ADDRESS]
[--web.listen-port WEB.LISTEN_PORT]
[--web.telemetry-path WEB.TELEMETRY_PATH] [--disable-cache] [--disable-settings-metrics]
[--disable-statement-history-metrics] [--disable-https] [--ssl-keyfile SSL_KEYFILE]
[--ssl-certfile SSL_CERTFILE] [--parallel PARALLEL] [--log.filepath LOG.FILEPATH]
[--log.level {debug,info,warn,error,fatal}] [--version] GBase 8c Exporter (DBMind):
Monitoring for GBase 8c.
optional arguments:
-h, --help       show this help message and exit
--url URL        GBase 8c database target url.
--config CONFIG path to config dir or file.
--constant-labels CONSTANT_LABELS
a list of label=value separated by cgbasea(,).
--web.listen-address WEB.LISTEN_ADDRESS
address on which to expose metrics and web interface
--web.listen-port WEB.LISTEN_PORT
listen port to expose metrics and web interface
--web.telemetry-path WEB.TELEMETRY_PATH
path under which to expose metrics.
--disable-cache      force not using cache.
--disable-settings-metrics
not collect pg_settings.yml metrics.
```

```
--disable-statement-history-metrics
not collect statement-history metrics (including slow queries).
--disable-https       disable Https schema
--ssl-keyfile SSL_KEYFILE
set the path of ssl key file
--ssl-certfile SSL_CERTFILE
set the path of ssl certificate file
--parallel PARALLEL not collect pg_settings.yml metrics.
--log.filepath LOG.FILEPATH
the path to log
--log.level {debug,info,warn,error,fatal}
only log messages with the given severity or above. Valid levels: [debug, info, warn, error, fatal]
--version       show program's version number and exit
```

表 1-6 GBase 8c-exporter 的命令行参数详情表

参数	参数说明	取值范围
–url	GBase 8c server 的连接地址，例如 postgres://user:pwd@host:port/dbname	-
–constant-labels	常量标签，将采集到的指标项中强行添加该标签列表	1024-65535
-h, –help	帮助选项	-
–disable-https	禁用 Https 协议	-
–ssl-keyfile	Https 协议使用的证书私钥文件路径	-
–ssl-certfile	Https 协议使用的证书文件路径	-
–web.listen-address	该 exporter 服务的绑定 IP	-
–web.listen-port	该 exporter 服务的侦听端口	1024-65535
–web.telemetry-path	该 exporter 采集指标的 URI 地址，默认	-

	为 /metrics	
–config	显性指定的待采集指标配置文件路径	-
–log.filepath	日志文件保存路径，默认保存在当前目录下	-
–log.level	日志文件的打印级别，默认为 INFO 级别	debug, info, warn, error, fatal
–version	显示版本信息	-
–disable-cache	禁止使用缓存	-
–disable-settings-metrics	禁止采集 pg_settings 表的值	-
–disable-statement-history-metrics	禁止收集 statement_history 表中的慢 SQL 信息	-
–parallel	连接到 GBase 8c 的数据库连接池的大小	正整数

1.2.1.6 常见问题处理

(1) 提示需要用户提供–ssl-keyfile 与–ssl-certfile 选项:

上述 exporter 默认采用 Https 模式通信，因此需要用户指定证书及其私钥文件的路径。相反，如果用户只想采用 Http 模式，则需要显性给定–disable-https 选项，从而禁用 Https 协议。

(2) 提示用户需要输入 PEM 密码 (Enter PEM pass phrase) :

如果用户采用 Https 模式，并给定了证书及其秘钥文件的路径，且该秘钥文件是经过加密的，则需要用户输入该加密私钥证书文件的密码。

1.3 DBMind 的 AI 子功能

1.3.1X-Tuner：参数调优与诊断

1.3.1.1 概述

X-Tuner 是一款数据库集成的参数调优工具，通过结合深度强化学习和全局搜索算法等 AI 技术，实现在无需人工干预的情况下，获取最佳数据库参数配置。本功能不强制与数据库环境部署到一起，支持独立部署，脱离数据库安装环境独立运行。

1.3.1.2 使用准备

前提条件与使用事项

- 数据库状态正常、客户端能够正常连接、且要求数据库内导入数据，以便调优程序可以执行 benchmark 测试调优效果。
- 使用本工具需要指定登录到数据库的用户身份，要求该登录到数据库上的用户具有足够的权限，以便可以获得充足的数据库状态信息。
- 使用登录到数据库宿主机上的 Linux 用户，需要将$GAUSSHOME/bin 添加到 PATH 环境变量中，即能够直接运行 gsql、gs_guc、gs_ctl 等数据库运维工具。
- 本工具支持以三种模式运行，其中 tune 和 train 模式要求用户配置好 benchmark 运行环境，并导入数据，本工具将会通过迭代运行 benchmark 来判断修改后的参数是否有性能提升。
- recgbaseend 模式建议在数据库正在执行 workload 的过程中执行，以便获得更准确的实时 workload 信息。
- 本工具默认带有 TPC-C、TPC-H、TPC-DS 以及 sysbench 的 benchmark 运行脚本样 例，如果用户使用上述 benchmark 对数据库系统进行压力测试，则可以对上述配置文件进行适度修改或配置。如果需要适配用户自己的业务场景，需要您参照 benchmark 目录中的 template.py 文件编写驱动您自定义 benchmark 的脚本文件。

原理简介

调优程序是一个独立于数据库内核之外的工具，需要提供数据库及其所在实例的用户名和登录密码信息，以便控制数据库执行 benchmark 进行性能测试；在启动调优程序前，要求用户测试环境交互正常，能够正常跑通 benchmark 测试脚本、能够正常连接数据库。

说明

如果需要调优的参数中，包含重启数据库后才能使修改生效的参数，那么在调优过程中数据库将会重启多次。如果用户的数据库正在执行作业，请慎用 train 与 tune 模式。

调优程序 X-Tuner 包含三种运行模式，分别是：

- recgbaseend：通过用户指定的用户名等信息登录到数据库环境中，获取当前正在运行的 workload 特征信息，根据上述特征信息生成参数推荐报告。报告当前数据库中不合理的参数配置和潜在风险等；输出根据当前正在运行的 workload 行为和特征；输出推荐的参数配置。该模式是秒级的，不涉及数据库的重启操作，其他模式可能需要反复重启数据库。
- train：通过用户提供的 benchmark 信息，不断地进行参数修改和 benchmark 的执行。通过反复的迭代过程，训练强化学习模型，以便用户在后面通过 tune 模式加载该模型进行调优。
- tune：使用优化算法进行数据库参数的调优，当前支持两大类算法，一种是深度强化学习，另一种是全局搜索算法（全局优化算法）。深度强化学习模式要求先运行 train 模式，生成训练后的调优模型，而使用全局搜索算法则不需要提前进行训练，可以直接进行搜索调优。

须知

如果在 tune 模式下，使用深度强化学习算法，要求必须有一个训练好的模型，且要求训练该模型时的参数与进行调优时的参数列表（包括 max 与 min）必须一致。

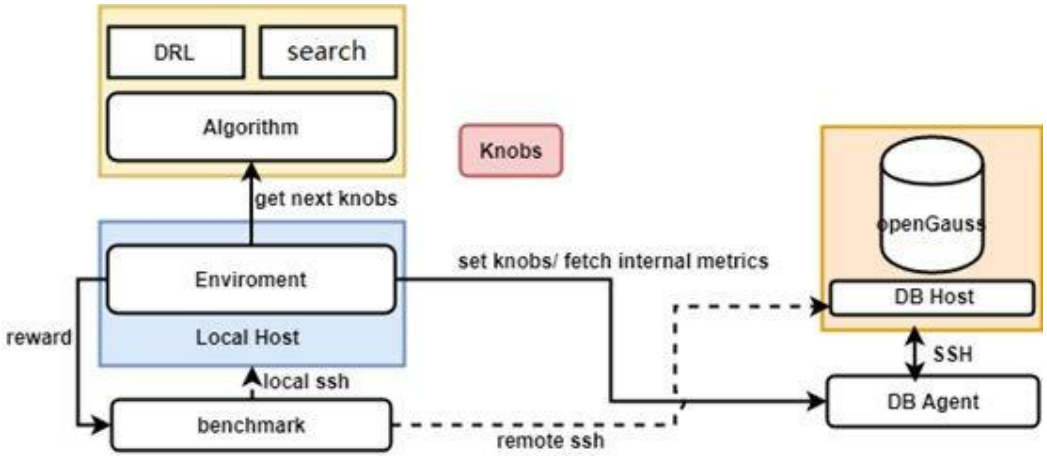

图 1-2 X-Tuner 结构图

- X-Tuner 的整体架构如图 11-2 X-Tuner 结构图所示，系统可以分为：

- DB 侧：通过 DB_Agent 模块对数据库实例进行抽象，通过该模块可以获取数据库内部的状态信息、当前数据库参数、以及设置数据库参数等。DB 侧包括登录数据库环境使用的 SSH 连接。
- 算法侧：用于调优的算法包，包括全局搜索算法（如贝叶斯优化、粒子群算法等）和深度强化学习（如 DDPG）。
- X-Tuner 主体逻辑模块：通过 Enviroment 模块进行封装，每一个 step 就是一次调优过程。整个调优过程通过多个 step 进行迭代。
- benchmark：由用户指定的 benchmark 性能测试脚本，用于运行 benchmark 作业，通过跑分结果反映数据库系统性能优劣。

  说明

  应确保 benchmark 脚本跑分结果越大表示性能越好。

  例如 TPCH 这种衡量 SQL 语句整体执行时长的 benchmark，可以通过取总体执行时间的相反数作为 benchmark 的输出分数。

X-Tuner 的运行和安装方法

执行下述命令即可获取 xtuner 功能帮助

```
gs_dbmind component xtuner --help
```

用户可据此给定不同的命令行执行相应的功能。

X-Tuner 的配置文件说明

X-Tuner 在运行前需要加载配置文件，用户可以通过 --help 命令查看默认加载的配置文件绝对路径：

```
...
-x TUNER_CONFIG_FILE, --tuner-config-file TUNER_CONFIG_FILE This is the path of the
core configuration file of the X-Tuner. You can specify the path of the new
configuration file. The default path is /path/to/xtuner/xtuner.conf.
You can modify the configuration file to control the tuning process.
...
```

修改配置文件的配置项可以指引 X-Tuner 执行不同的动作，用户可以根据自己的不同需求来修改配置文件的内容，配置文件的配置项说明详见表 11-8。如果需要修改配置文件的加载路径，则可以通过选项-x 命令行选项来指定。

Benchmark 的选择与配置

Benchmark 的驱动脚本存放路径为 X-Tuner 目录（即 $GAUSSHOME/bin/dbmind/components/xtuner，下同）的子目录 benchmark 中。X-Tuner 自带常用的 benchmark 驱动脚本，例如基于时间周期的探测脚本(默认)、TPC-C、TPC-H 等。X-Tuner 通过调用 benchmark/init.py 文件中 get_benchmark_instance()命令来加载不同的 benchmark 驱动脚本，获取 benchmark 驱动实例。其中，benchmark 驱动脚本的格式说明如下：

- 驱动脚本文件名：表示 benchmark 的名字，该名字用于表示驱动脚本的唯一性，可通过在 X-Tuner 的配置文件中的配置项 benchmark_script 来指定选择加载哪个 benchmark 驱动脚本。
- 驱动脚本内容三要素：path 变量、cmd 变量以及 run 函数。下面分别介绍驱动脚本的内容三要素：
    - path 变量：表示 benchmark 脚本的存放地址，可以直接在驱动脚本中修改，也可以通过配置文件的 benchmark_path 配置项来指定。
    - cmd 变量：表示执行 benchmark 脚本需要运行的命令，可以直接在驱动脚本中修改，也可以通过配置文件的 benchmark_cmd 配置项来指定。cmd 中的文本允许使用占位符，用于获取某些运行 cmd 命令时的必要信息，使用示例参见 TPC-H 驱动脚本示例。这些占位符包括：
        - {host}：数据库宿主机的 IP 地址
        - {port}：数据库实例的侦听端口号
        - {user}：登录数据库系统上的用户名
        - {password}：与登录数据库系统上的用户相匹配的密码
        - {db}：正在进行调优的数据库名
    - run 函数：该函数的函数签名为：

```
def run(remote_server, local_host) -> float:
```

其中，返回数据类型为 float，表示 benchmark 执行后的评估分数值，要求该值越大表示性能越好，例如使用 TPC-C 跑分结果 tpmC 即可作为返回值，TPC-H 的全部 SQL 语句执行总时间的相反数（取相反数后可保证返回值越大则性能越好）也可作为返回值。

remote_server 变量是 X-Tuner 程序传递给脚本使用的远端主机（数据库宿主机）的 shell 命令接口，local_host 变量是 X-Tuner 程序传递给脚本使用的本地主机（运行 X-Tuner 脚本的主机）的 shell 命令接口。上述 shell 命令接口提供的方法包括：

exec_cgbaseand_sync(cgbaseand, timeout) 功能：该方法用于在主机上执行 shell 命令。参数列表：
cgbaseand 必选，数据类型可以是 str, 以及元素为 str 类型的 list 或 tuple; timeout 可选，表示命令执行的超时时长，单位是秒。
返回值：
返回二元组 (stdout, stderr)，stdout 表示标准输出流结果，stderr 表示标准错误流结果，数据类型均为 str. exit_status
功能：该属性表示最近一条 shell 命令执行后的退出状态码(exit status code)。
说明：一般情况，退出状态码为 0 表示执行正常，非 0 表示存在错误。

Benchmark 驱动脚本示例说明

(1) TPC-C 驱动脚本

```
from tuner.exceptions import ExecutionError
# WARN: You need to download the benchmark-sql test tool to the system, # replace the PostgreSQL JDBC driver with the GBase 8c driver,
# and configure the benchmark-sql configuration file.
# The program starts the test by running the following cgbaseand:
path = '/path/to/benchmarksql/run' # TPC-C 测试脚本 benchmark-sql 的存放路径
cmd = "./runBenchmark.sh props.gs" # 自定义一个名为 props.gs 的 benchmark-sql 测试配置文件
def run(remote_server, local_host):
# 切换到 TPC-C 脚本目录下，清除历史错误日志，然后运行测试命令。
# 此处最好等待几秒钟，因为 benchmark-sql 测试脚本生成最终测试报告是通过一个 shell 脚本实现的， 整个过程会有延迟，
# 为了保证能够获取到最终的 tpmC 数值报告，我们这里选择等待 3 秒钟。
stdout, stderr = remote_server.exec_cgbaseand_sync(['cd %s' % path, 'rm -rf benchmarksql-error.log', cmd, 'sleep 3'])
# 如果标准错误流中有数据，则报异常退出。
if len(stderr) > 0:
raise ExecutionError(stderr)
# 寻找最终 tpmC 结果
tpmC = None
split_string = stdout.split() # 对标准输出流结果进行分词。
for i, st in enumerate(split_string):
# 在 5.0 版本的 benchmark-sql 中，tpmC 最终测试结果数值在 ‘(NewOrders)’关键字的后两位，正常情况下，找到该字段后直接返回即可。
if "(NewOrders)" in  st: tpmC = split_string[i + 2] break
stdout, stderr = remote_server.exec_cgbaseand_sync( "cat %s/benchmarksql-error.log" % path)
nb_err = stdout.count("ERROR:") # 判断整个 benchmark 运行过程中，是否有报错，记录报错的错误数
```

```
return float(tpmC) - 10 * nb_err # 这里将报错的错误数作为一个惩罚项，惩罚系数为 10，越高的惩罚系数表示越看中报错的数量.
```

(2) TPC-H 驱动脚本

```
import time
from tuner.exceptions import ExecutionError
# WARN: You need to import data into the database and SQL statements in the following path will be executed.
# The program automatically collects the total execution duration of these SQL statements. path = '/path/to/tpch/queries' # 存放 TPC-H 测试用的 SQL 脚本目录
cmd = "gsql -U {user} -W {password} -d {db} -p {port} -f {file}" # 需要运行 TPC-H 测试脚本的命令，一般使用'gsql -f 脚本文件' 来运行
def run(remote_server, local_host):
# 遍历当前目录下所有的测试用例文件名
find_file_cmd = "find . -type f -name '*.sql'"
stdout, stderr = remote_server.exec_cgbaseand_sync(['cd %s' % path, find_file_cmd]) if len(stderr) > 0:
raise ExecutionError(stderr) files = stdout.strip().split('\n') time_start = time.time()
for file in files:
# 使用 file 变量替换 {file}，然后执行该命令行。perform_cmd = cmd.format(file=file)
stdout, stderr = remote_server.exec_cgbaseand_sync(['cd %s' % path, perform_cmd]) if len(stderr) > 0:
print(stderr)
# 代价为全部测试用例的执行总时长
cost = time.time() - time_start
# 取相反数，适配 run 函数的定义：返回结果越大表示性能越好。
return - cost
```

1.3.1.3 使用示例

X-Tuner 支持三种模式，分别是获取参数诊断报告的 recgbaseend 模式、训练强化学习模型的 train 模式、以及使用算法进行调优的 tune 模式。上述三种模式可以通过命令行参数来区别，通过配置文件来指定具体的细节。

配置数据库连接信息

三种模式连接数据库的配置项是相同的，有两种方式：一种是直接通过命令行输入详细的连接信息，另一种是通过 JSON 格式的配置文件输入，下面分别对两种指定数据库连接信息的方法进行说明。

(1) 通过命令行指定：

分别传递 --db-name --db-user --port --host --host-user 参数，可选 --host-ssh-port 参数，例如：

```
gs_dbmind component xtuner recgbaseend --db-name postgres --db-user gbase --port 5678
--host 192.168.1.100 --host-user gbase
```

(2) 通过 JSON 格式的连接信息配置文件指定：

JSON 配置文件的示例如下，并假设文件名为 connection.json：

```
{
"db_name": "postgres", # 数据库名
"db_user": "dba",              # 登录到数据库上的用户名"host": "127.0.0.1", # 数据库宿主
机的 IP 地址"host_user": "dba",              # 登录到数据库宿主机的用户名"port": 5432, #
数据库的侦听端口号"ssh_port": 22          # 数据库宿主机的 SSH 侦听端口号
}
```

则可通过 -f connection.json 传递。

说明

为了防止密码泄露，配置文件和命令行参数中默认都不包含密码信息，用户在输入上述连接信息后，程序会采用交互式的方式要求用户输数据库密码以及操作系统登录用户的密码。

recgbaseend 模式使用示例

对 recgbaseend 模式生效的配置项为 scenario，若为 auto，则自动检测 workload 类型。执行下述命令，获取诊断结果：

```
gs_dbmind component xtuner recgbaseend -f connection.json
```

则可以生成诊断报告如下：

```
Start to recommend knobs. Just a moment, please.
************************** Knob Recommendation Report **************************
INFO:
+--------------------------------------+----------------------+
|                Metric                |        Value         |
+--------------------------------------+----------------------+
|             workload_type            |          tp          |
|         average_connection_age       |          0           |
|         dirty_background_bytes       |          0           |
|             temp_file_size           |          0           |
|          current_connections         |         0.0          |
|          current_locks_count         |         0.0          |
|      current_prepared_xacts_count    |         0.0          |
|          rollback_commit_ratio       | 0.09168372786632421  |
|                uptime                |  0.122942879722222   |
| checkpoint_proactive_triggering_ratio |  0.488598416181662   |
|         fetched_returned_ratio       |  0.9915911452033203  |
|             cache_hit_rate           |  0.9979742937232552  |
|            read_write_ratio          |  123.86665549830312  |
|           all_database_size          |  134154882.48046875  |
|          search_modify_ratio         |  187.59523392981777  |
|                ap_index              |  2.3759498376861847  |
|            current_free_mem          |       31161892       |
|             os_mem_total             |       32779460       |
|        checkpoint_avg_sync_time      |   381.359603091308   |
|  checkpoint_dirty_writing_time_window |        450.0         |
|             max_processes            |          46          |
|          track_activity_size         |         46.0         |
|            write_tup_speed           |   6810.36309197048   |
|                used_mem              |     73988850.25      |
|             os_cpu_count             |          8           |
|              block_size              |         8.0          |
|            read_tup_speed            |   845237.440716804   |
|      shared_buffer_toast_hit_rate    |  98.16007359705611   |
|      shared_buffer_tidx_hit_rate     |  99.11667280088332   |
|       shared_buffer_idx_hit_rate     |  99.74473859023848   |
|      shared_buffer_heap_hit_rate     |  99.81099543813004   |
|          enable_autovacuum           |         True         |
|               is_64bit               |         True         |
|                is_hdd                |         True         |
|            load_average              | [1.89, 3.175, 3.005] |
+--------------------------------------+----------------------+
p.s: The unit of storage is kB.
WARN:
[0]. The number of CPU cores is a little small. Please do not run too high concu
rrency. You are recommended to set max_connections based on the number of CPU co
res. If your job does not consume much CPU, you can also increase it.
[1]. The value of wal_buffers is a bit high. Generally, an excessively large val
ue does not bring better performance. You can also set this parameter to -1. The
 database automatically performs adaptation.
BAD:
[0]. The database runs for a short period of time, and the database description
may not be accumulated. The recommendation result may be inaccurate.
************************** Recommended Knob Settings ***************************
+---------------------------+-----------+---------+----------+---------+
|            name           | recommend |   min   |   max    | restart |
+---------------------------+-----------+---------+----------+---------+
|       shared_buffers      |  1638973  |  614614 | 1884818  |   True  |
|      max_connections      |     43    |    24   |   500    |   True  |
|    effective_cache_size   |  1638973  | 1638973 | 24584595 |  False  |
|       wal_buffers         |   51217   |   2048  |  51217   |   True  |
|      random_page_cost     |    3.0    |    2.0  |    3.0   |  False  |
| default_statistics_target |    100    |    10   |    150   |  False  |
+---------------------------+-----------+---------+----------+---------+
```

图 1-3 recgbaseend 模式生成的报告示意图

在上述报告中，推荐了该环境上的数据库参数配置，并进行了风险提示。报告同时生成了当前 workload 的特征信息，其中有几个特征是比较有参考意义的：

- temp_file_size：产生的临时文件数量，如果该结果大于 0，则表明系统使用了临 时文件。使用过多的临时文件会导致性能不佳，如果可能的话，需要提高 work_mem 参数的配置。
- cache_hit_rate：shared_buffer 的缓存命中率，表明当前 workload 使用缓存的效率。
- read_write_ratio：数据库作业的读写比例。
- search_modify_ratio：数据库作业的查询与修改数据的比例。
- ap_index：表明当前 workload 的 AP 指数，取值范围是 0 到 10，该数值越大，表明越偏向于数据分析与检索。
- workload_type：根据数据库统计信息，推测当前负载类型，分为 AP、TP 以及 HTAP 三种类型。
- checkpoint_avg_sync_time：数据库在 checkpoint 时，平均每次同步刷新数据到 磁盘的时长，单位是毫秒。
- load_average：平均每个 CPU 核心在 1 分钟、5 分钟以及 15 分钟内的负载。一般 地，该数值在 1 左右表明当前硬件比较匹配 workload、在 3 左右表明运行当前作业压力比较大，大于 5 则表示当前硬件环境运行该 workload 压力过大（此时一般建议减少负载或升级硬件）。

  说明

  - recgbaseend 模式会读取数据库中的 pg_stat_database 以及 pg_stat_bgwriter 等系统表中的信息，需要登录到数据库上的用户具有足够的权限(建议为管理员权限，可通过 alter user username sysadmin；授予 username 相应的权限）。
  - 由于某些系统表会一直记录统计信息，这可能会对负载特征识别造成干扰，因此建议最好先清空某些系统表的统计信息，运行一段时间的 workload 后再使用 recgbaseend 模式进行诊断，以便获得更准确的结果。清除统计信息的方法为：

```
select pg_stat_reset_shared('bgwriter');
select pg_stat_reset();
```

train 模式使用示例

该模式是用来训练深度强化学习模型的，与该模式有关的配置项为：

- rl_algorithm：用于训练强化学习模型的算法，当前支持设置为 ddpg.
- rl_model_path：训练后生成的强化学习模型保存路径。

- rl_steps：训练过程的最大迭代步数。
- max_episode_steps：每个回合的最大步数。
- scenario：明确指定的 workload 类型，如果为 auto 则为自动判断。在不同模式下，推荐的调优参数列表也不一样。
- tuning_list：用户指定需要调哪些参数，如果不指定，则根据 workload 类型自动推荐应该调的参数列表。如需指定，则 tuning_list 表示调优列表文件的路径。一个调优列表配置文件的文件内容示例如下：

```
{
"work_mem": { "default": 65536,
"min": 65536,
"max": 655360,
"type": "int", "restart": false
},
"shared_buffers": { "default": 32000,
"min": 16000,
"max": 64000,
"type": "int", "restart": true
},
"random_page_cost": { "default": 4.0,
"min": 1.0,
"max": 4.0,
"type": "float", "restart": false
},
"enable_nestloop": { "default": true, "type": "bool", "restart": false
}
}
```

待上述配置项配置完成后，可以通过下述命令启动训练：

```
gs_dbmind component xtuner train -f connection.json
```

训练完成后，会在配置项 rl_model_path 指定的目录中生成模型文件。

tune 模式使用示例

tune 模式支持多种算法，包括基于强化学习（Reinforcement Learning，RL）的 DDPG 算法、基于全局搜索算法（Global OPtimization algorithm，GOP）算法的贝叶斯优化算法（Bayesian Optimization）以及粒子群算法（Particle Swarm Optimization，PSO）。

与 tune 模式相关的配置项为：

- tune_strategy：指定选择哪种算法进行调优，支持 rl（使用强化学习模型进行调优）、gop（使用全局搜索算法）以及 auto（自动选择）。若该参数设置为 rl，则 rl 相关的配置项生效。除前文提到过的 train 模式下生效的配置项外，test_episode 配置项也生效，该配置项表明调优过程的最大回合数，该参数直接影响了调优过程的执行时间（一般地，数值越大越耗时）。
- gop_algorithm：选择何种全局搜索算法，支持 bayes 以及 pso。
- max_iterations：最大迭代轮次，数值越高搜索时间越长，效果往往越好。
- particle_nums：在 PSO 算法上生效，表示粒子数。
- scenario 与 tuning_list 见上文 train 模式中的描述。

待上述配置项配置完成后，可以通过下述命令启动调优：

```
gs_dbmind component xtuner tune -f connection.json
```

⚠ 注意

在使用 tune 和 train 模式前，用户需要先导入 benchmark 所需数据并检查 benchmark 能否正常跑通。调优过程结束后，调优程序会自动恢复调优前的数据库参数配置。

1.3.1.4 获取帮助

启动调优程序之前，可以通过如下命令获取帮助信息：

```
gs_dbmind component xtuner --help
```

输出帮助信息结果如下：

```
usage: [-h] [--db-name DB_NAME] [--db-user DB_USER] [--port PORT] [--host HOST]
[--host-user HOST_USER]
[--host-ssh-port HOST_SSH_PORT] [-f DB_CONFIG_FILE] [-x TUNER_CONFIG_FILE]
[-v]
{train,tune,recgbaseend}
X-Tuner: a self-tuning tool integrated by GBase 8c. positional arguments:
{train,tune,recgbaseend}
Train a reinforcement learning model or tune database
by model. And also can recgbaseend best_knobs according to your workload.
optional arguments:
-h, --help       show this help message and exit
-f DB_CONFIG_FILE, --db-config-file DB_CONFIG_FILE
```

You can pass a path of configuration file otherwise you should enter database information by cgbaseand arguments manually. Please see the template file share/server.json.template.
-x TUNER_CONFIG_FILE, --tuner-config-file TUNER_CONFIG_FILE This is the path of the core configuration file of the X-Tuner. You can specify the path of the new configuration file. The default path is xtuner.conf. You can modify the configuration file to control the tuning process.
-v, --version show program's version number and exit
Database Connection Information:
--db-name DB_NAME The name of database where your workload running on.
--db-user DB_USER Use this user to login your database. Note that the user must have sufficient permissions.
--port PORT Use this port to connect with the database.
--host HOST The IP address of your database installation host.
--host-user HOST_USER
The login user of your database installation host.
--host-ssh-port HOST_SSH_PORT
The SSH port of your database installation host.

1.3.1.5 命令参考

表 1-7 命令行参数

参数	参数说明	取值范围
mode	指定调优程序运行的模式	train，tune，recgbaseend
--tuner-config-file, -x	X-Tuner 的核心参数配置文件路径，默认路径为安装目录下的 xtuner.conf	-
--db- config-file,-f	调优程序的用于登录到数据库宿主机上的连接信息配置文件路径，若通过该文件配置数据库连接信息，则下述数据库连接信息可省略	-
--db-name	指定需要调优的数据库名	-
--db-user	指定以何用户身份登陆到调优的数据库上	-
--host	数据库实例的宿主机 IP	-

--host-user	指定以何用户身份登陆到数据库实例的宿主机上，要求改用户名的环境变量中可以找到 gsql、gs_ctl 等数据库运维工具。	-
--host-ssh- port	数据库实例所在宿主机的 SSH 端口号，可选，默认为 22	-
--help, -h	返回帮助信息	-
--version, -v	返回当前工具版本号	-

表 1-8 配置文件中的参数详解

参数名	参数说明	取值范围
logfile	生成的日志存放路径	-
output_tuning_result	可选，调优结果的保存路径	-
verbose	是否打印详情	on, off
recorder_file	调优中间信息的记录日志存放路径	-
tune_strategy	调优模式下采取哪种策略	rl, gop, auto
drop_cache	是否在每一个迭代轮次中进行 drop cache，drop cache 可以使 benchmark 跑分结果更加稳定。若启动该参数，则需要将登录的系统用户加入到 /etc/ sudoers 列表中，同时为其增加 NOPASSWD 权限（由于该权限可能过高，建议临时启用该权限，调优结束后关闭）。	on, off
used_mem_penalty_term	数据库使用总内存的惩罚系数，用于防止通过无限量占用内存而换取的性能表现。该数值越大，惩罚力度越大。	建议 0 ~ 1

rl_algorithm	选择何种 RL 算法	ddpg
rl_model_path	RL 模型保存或读取路径，包括保存目录名与文件名前缀。在 train 模式下该路径用于保存模型，在 tune 模式下则用于读取模型文件	-
rl_steps	深度强化学习算法迭代的步数	-
max_episode_steps	每个回合的最大迭代步数	-
test_episode	使用 RL 算法进行调优模式的回合数	-
gop_algorithm	采取何种全局搜索算法	bayes, pso
max_iterations	全局搜索算法的最大迭代轮次（并非确定数值，可能会根据实际情况多跑若干轮）	-
particle_nums	PSO 算法下的粒子数量	-
benchmark_script	使用何种 benchmark 驱动脚本，该选项指定加载 benchmark 路径下同名文件，默认支持 TPC-C、TPC- H 等典型 benchmark	tpcc, tpch, tpcds, sysbench ...
benchmark_path	benchmark 脚本的存储路径，若没有配置该选项， 则使用 benchmark 驱动脚本中的配置	-
benchmark_cmd	启动 benchmark 脚本的命令，若没有配置该选项， 则使用 benchmark 驱动脚本中的配置	-
benchmark_period	仅对 period benchmark 有效，表明整个 benchmark 的测试周期是多少，单位是秒	-
scenario	用户指定的当前 workload 所属的类型	tp, ap, htap

tuning_list	准备调优的参数列表文件，可参考 share/knobs.json.template 文件	-

1.3.1.6 常见问题处理

- 数据库实例连接失败：请检查数据库实例的情况，是否数据库实例出现了问题或安全权限配置（pg_hba.conf 文件中的配置项）不正确。
- 重启失败：请检查数据库实例健康情况，确保数据库实例工作正常。
- 跑 TPC-C 作业时发现性能越来越慢：TPC-C 等高并发场景下的压力测试，往往伴随着大量的数据修改。由于每一次测试并非是幂等的（TPC-C 数据库数据量的增加、没有进行 vacuum full 清理掉失效元组、数据库没有触发 checkpoint、没有进行 drop cache 等），因此一般建议 TPC-C 等伴随着较多数据写入的 benchmark 应该 每隔一段时间（视具体并发量、执行时长的不同而异）重新导入一次数据，比较简单的方法是备份$PGDATA 目录。
- TPC-C 跑作业时，TPC-C 驱动脚本报异常 “TypeError: float() argument must be a string or a number, not 'NoneType'”（不能将 None 转换为 float 类型）：这是因为没有获取到 TPC-C 的压测返回结果，造成该问题的原因比较多，请首先手动检测是否能够跑通 TPC-C 并能够获取返回结果。若无上述问题，则建议将 TPC-C 驱动脚本中的命令列表中的 "sleep" 命令延迟时间设得更大一些。

1.3.2Index-advisor：索引推荐

本节介绍索引推荐的功能，共包含三个子功能：单 query 索引推荐、虚拟索引和 workload 级别索引推荐。

1.3.2.1 单 query 索引推荐

单 query 索引推荐功能支持用户在数据库中直接进行操作，本功能基于查询语句的语义信息和数据库的统计信息，对用户输入的单条查询语句生成推荐的索引。本功能涉及的函数接口如下。

表 1-9 单 query 索引推荐功能的接口

函数名	参数	功能

gs_index_advise	SQL 语句字符串	针对单条查询语句生成推荐索引

说明

- 本功能仅支持单条 SELECT 类型的语句，不支持其他类型的 SQL 语句。
- 本功能暂不支持列存表、段页式表、普通视图、物化视图、全局临时表以及密态数据库。

使用方法

使用上述函数，获取针对该 query 生成的推荐索引，推荐结果由索引的表名和列名组成。

例如：

```
gsql=> select "table", "column" from gs_index_advise('SELECT c_discount from
bmsql_customer where c_w_id = 10');
        table          |    column
-----------------+----------
 bmsql_customer | c_w_id
(1 row)
```

上述结果表明：应当在 bmsql_customer 的 c_w_id 列上创建索引，例如可以通过下述 SQL 语句创建索引：

```
CREATE INDEX idx on bmsql_customer(c_w_id);
```

某些 SQL 语句，也可能被推荐创建联合索引，例如：

```
gsql=# select "table", "column" from gs_index_advise('select name, age, sex from t1 where
age >= 18 and age < 35 and sex = "f";');
 table | column
-------+------------
 t1        | age, sex
(1 row)
```

则上述语句表明应该在表 t1 上创建一个联合索引（age，sex），则可以通过下述命令创建：

```
CREATE INDEX idx1 on t1(age, sex);
```

针对分区表可推荐具体索引类型，例如：

```
gsql=# select "table", "column", "indextype" from gs_index_advise('select name, age, sex from
range_table where age = 20;');
 table | column | indextype
-------+--------+-----------
```

```
 t1      | age      | global
(1 row)
```

说明

系统函数 gs_index_advise()的参数是文本型，如果参数中存在如单引号（’）等特殊字符，可以使用单引号（’）进行转义，可参考上述示例。

1.3.2.2 虚拟索引

虚拟索引功能支持用户在数据库中直接进行操作，本功能将模拟真实索引的建立，避免真实索引创建所需的时间和空间开销，用户基于虚拟索引，可通过优化器评估该索引对指定查询语句的代价影响。

本功能涉及的系统函数接口如下表所示：

表 1-10 虚拟索引功能的接口

函数名	参数	功能
hypopg_create_index	创建索引语句的字符串	创建虚拟索引。
hypopg_display_index	无	显示所有创建的虚拟索引信息。
hypopg_drop_index	索引的 oid	删除指定的虚拟索引。
hypopg_reset_index	无	清除所有虚拟索引。
hypopg_estimate_size	索引的 oid	估计指定索引创建所需的空间大小。

本功能涉及的 GUC 参数如下：

表 1-11 虚拟索引功能的 GUC 参数

参数名	功能	默认值
enable_hypo_index	是否开启虚拟索引功能	off

使用步骤

(1) 使用函数 hypopg_create_index 创建虚拟索引。例如：

```
gsql=> select * from hypopg_create_index('create index on bmsql_customer(c_w_id)');
```

```
 indexrelid |                    indexname
------------+-------------------------------------
        329726 | <329726>btree_bmsql_customer_c_w_id
(1 row)
```

(2) 开启 GUC 参数 enable_hypo_index，该参数控制数据库的优化器进行 EXPLAIN 时是否考虑创建的虚拟索引。通过对特定的查询语句执行 explain，用户可根据优化器给出的执行计划评估该索引是否能够提升该查询语句的执行效率。例如：

```
gsql=> set enable_hypo_index = on;
SET
```

开启 GUC 参数前，执行 EXPLAIN + 查询语句：

```
gsql=> explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                                        QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on bmsql_customer    (cost=0.00..52963.06 rows=31224 width=4)
    Filter: (c_w_id = 10)
(2 rows)
```

开启 GUC 参数后，执行 EXPLAIN + 查询语句：

```
gsql=> explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 [Bypass]
 Index Scan using <329726>btree_bmsql_customer_c_w_id on bmsql_customer
(cost=0.00..39678.69 rows=31224 width=4)
    Index Cond: (c_w_id = 10)
(3 rows)
```

通过对比两个执行计划可以观察到，该索引预计会降低指定查询语句的执行代价，用户可考虑创建对应的真实索引。

(3) （可选）使用函数 hypopg_display_index 展示所有创建过的虚拟索引。例如：

```
gsql=> select * from hypopg_display_index();
                    indexname                    | indexrelid |        table          |
column
-----------------------------------------------+------------+----------------+------------------
 <329726>btree_bmsql_customer_c_w_id            |        329726 | bmsql_customer | (c_w_id)
 <329729>btree_bmsql_customer_c_d_id_c_w_id |        329729 | bmsql_customer | (c_d_id,
c_w_id)
(2 rows)
```

(4) （可选）使用函数 hypopg_estimate_size 估计虚拟索引创建所需的空间大小（单位：字节）。例如：

```
gsql=> select * from hypopg_estimate_size(329730);
 hypopg_estimate_size
----------------------
                15687680
(1 row)
```

(5) 删除虚拟索引。

使用函数 hypopg_drop_index 删除指定 oid 的虚拟索引。例如：

```
gsql=> select * from hypopg_drop_index(329726);
 hypopg_drop_index
-------------------
 t
(1 row)
```

使用函数 hypopg_reset_index 一次性清除所有创建的虚拟索引。例如：

```
gsql=> select * from hypopg_reset_index();
 hypopg_reset_index
--------------------

(1 row)
```

说明

- 执行 EXPLAIN ANALYZE 不会涉及虚拟索引功能。
- 创建的虚拟索引是数据库实例级别的，各个会话（session）之间可共享设置，关闭会话后虚拟索引仍可存在，但是重启数据库后将被清空。
- 本功能暂不支持视图、物化视图、列存表。

1.3.2.3 workload 级别索引推荐

对于 workload 级别的索引推荐，用户可通过运行数据库外的脚本使用此功能，本功能将包含有多条 DML 语句的 workload 作为输入，最终生成一批可对整体 workload 的执行表现进行优化的索引。同时，本功能提供从日志中抽取业务数据 SQL 流水的功能。

前提条件

- 数据库状态正常、客户端能够正常连接。

- 当前执行用户下安装有 gsql 工具，该工具路径已被加入到 PATH 环境变量中。
- 若使用本功能提供的业务数据抽取功能，需提前将要收集的节点的 GUC 参数按如下设置：

```
log_min_duration_statement = 0
log_statement= 'all'
```

说明

业务数据抽取完毕建议将上述 GUC 参数复原，否则容易导致日志文件膨胀。

业务数据抽取脚本使用步骤

(1) 按前提条件中要求设置相关 GUC 参数。

(2) 执行根据日志抽取 SQL 语句的功能，命令如下：

```
gs_dbmind component extract_log [l LOG_DIRECTORY] [f OUTPUT_FILE] [p
LOG_LINE_PREFIX] [-d DATABASE] [-U USERNAME][--start_time] [--sql_amount]
[--statement] [--json] [--max_reserved_period] [--max_template_num]
```

其中的输入参数依次为：

- LOG_DIRECTORY：pg_log 的存放目录。
- OUTPUT_PATH：输出 SQL 流水文件文件的保存路径，即抽取出的业务数据存放的文件路径。
- LOG_LINE_PREFIX：指定每条日志信息的前缀格式。
- DATABASE：（可选）数据库名称，不指定默认所有数据库。
- USERNAME：（可选）用户名称，不指定默认所有用户。
- start_time：（可选）日志收集的开始时间，不指定默认所有文件。
- sql_amount：（可选）收集 SQL 数量的最大值，不指定默认收集所有 SQL。
- statement：（可选）表示收集 pg_log 日志中 statement 标识开头的 SQL，不指定默认不收集。
- json：（可选）指定收集日志的文件存储格式为 SQL 归一化后的 json，不指定默认格式每条 SQL 占一行。
- max_reserved_period：（可选）指定 json 模式下，增量收集日志中保留的模板的最大的更新时长，不指定默认都保留，单位/天。
- max_template_num：（可选）指定 json 模式下保留的最大模板数量，不指定默认都保

留。

使用示例：

```
gs_dbmind component extract_log $GAUSSLOG/pg_log/dn_6001 sql_log.txt
'%m %c %d %p %a %x %n %e' -d postgres -U gbase --start_time '2021-07-06 00:00:00'
--statement
```

说明

若指定-d/-U 参数，日志打印每条日志信息的前缀格式需包含%d、%u，若需要抽取事务，必须指定%p，详见 log_line_prefix 参数。max_template_num 参数设置建议不超 5000 条，避免 workload 索引推荐执行时间过长。

(3) 将 1 中设置的 GUC 参数还原为设置前的值。

索引推荐脚本使用步骤

(1) 准备好包含有多条 DML 语句的文件作为输入的 workload，文件中每条语句占据一行。用户可从数据库的离线日志中获得历史的业务语句。

(2) 运行本功能，命令如下：

```
gs_dbmind component index_advisor [p PORT] [d DATABASE] [f FILE] [--h HOST] [-U
USERNAME] [-W PASSWORD][--schema SCHEMA]
[--max_index_num MAX_INDEX_NUM][--max_index_storage MAX_INDEX_STORAGE]
[--multi_iter_mode] [--multi_node]  [--json] [--driver] [--show_detail]
```

其中的输入参数依次为：

- PORT：连接数据库的端口号。
- DATABASE：连接数据库的名字。
- FILE：包含 workload 语句的文件路径。
- HOST：（可选）连接数据库的主机号。
- USERNAME：（可选）连接数据库的用户名。
- PASSWORD：（可选）连接数据库用户的密码。
- SCHEMA：模式名称。
- MAX_INDEX_NUM：（可选）最大的索引推荐数目。
- MAX_INDEX_STORAGE：（可选）最大的索引集合空间大小。
- multi_iter_mode：（可选）算法模式，可通过是否设置该参数来切换算法。

- json：（可选）指定 workload 语句的文件路径格式为 SQL 归一化后的 json，默认格式每条 SQL 占一行。
- driver：（可选）指定是否使用 python 驱动器连接数据库，默认 gsql 连接。
- show_detail：（可选）是否显示当前推荐索引集合的详细优化信息。

例如：

```
gs_dbmind component index_advisor 6001 postgres tpcc_log.txt --schema public
--max_index_num 10 --multi_iter_mode
```

推荐结果为一批索引，以多个创建索引语句的格式显示在屏幕上，结果示例。

```
create index ind0 on public.bmsql_stock(s_i_id,s_w_id);
create index ind1 on public.bmsql_customer(c_w_id,c_id,c_d_id);
create index ind2 on public.bmsql_order_line(ol_w_id,ol_o_id,ol_d_id);
create index ind3 on public.bmsql_item(i_id);
create index ind4 on public.bmsql_oorder(o_w_id,o_id,o_d_id);
create index ind5 on public.bmsql_new_order(no_w_id,no_d_id,no_o_id);
create index ind6 on public.bmsql_customer(c_w_id,c_d_id,c_last,c_first);
create index ind7 on public.bmsql_new_order(no_w_id);
create index ind8 on public.bmsql_oorder(o_w_id,o_c_id,o_d_id);
create index ind9 on public.bmsql_district(d_w_id);
```

说明

multi_node 参数需严格按照当前数据库架构进行指定，否则推荐结果不全，甚至导致无推荐结果。

1.3.3AI4DB：慢 SQL 根因分析

1.3.3.1 概述

慢 SQL 一直是数据运维中的痛点问题，如何有效诊断慢 SQL 根因是当前一大难题，工具结合 GBase 8c 自身特点融合了现网 DBA 慢 SQL 诊断经验，该工具可以支持慢 SQL 根因 15+，能同时按照可能性大小输出多个根因并提供针对性的建议。

1.3.3.2 环境部署

- 数据库运行正常。
- 指标采集系统运行正常。

1.3.3.3 使用指导

假设用户已经初始化配置文件目录 confpath，则可以通过下述命令实现本特性的功能：

- 仅启动慢 SQL 诊断功能（输出 Top3 根因），启动命令如下（更多用法参考对 service 子命令的说明）：

```
gs_dbmind service start -c confpath --only-run slow_query_diagnosis
```

- 用户交互式慢 SQL 诊断，命令如下：

```
gs_dbmind component slow_query_diagnosis show -c confpath --query SQL --start-time
timestamps0
--end-time timestamps1
```

- 用户手动清理历史预测结果，命令如下：

```
gs_dbmind component slow_query_diagnosis clean -c confpath --retention-days DAYS
```

- 停止已启动的服务，命令如下：

```
gs_dbmind service stop -c confpath
```

1.3.3.4 获取帮助

模块命令行说明：

```
gs_dbmind component slow_query_diagnosis --help usage: [-h] -c DIRECTORY [--query
SLOW_QUERY]
[--start-time TIMESTAMP_IN_MICROSECONDS]
[--end-time TIMESTAMP_IN_MICROSECONDS] [--retention-days DAYS]
{show,clean}
Slow Query Diagnosis: Analyse the root cause of slow query positional arguments:
{show,clean} choose a functionality to perform
optional arguments:
-h, --help       show this help message and exit
-c DIRECTORY, --conf DIRECTORY
set the directory of configuration files
--query SLOW_QUERY set a slow query you want to retrieve
--start-time TIMESTAMP_IN_MICROSECONDS
set the start time of a slow SQL diagnosis result to be retrieved
--end-time TIMESTAMP_IN_MICROSECONDS
set the end time of a slow SQL diagnosis result to be retrieved
--retention-days DAYS
clear historical diagnosis results and set the maximum number of days to retain data
```

1.3.3.5 命令参考

表 1-12 gs_dbmind component slow_query_diagnosis 命令行说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
action	动作参数	show：结果展示 clean：清理结果
-c，--conf	配置目录	-
--query	慢 SQL 文本	*
--start-time	显示开始时间的时间戳，单位毫秒	非负整数
--end-time	显示结束时间的时间戳，单位毫秒	非负整数
--retention-days	清理天数级结果	非负实数

1.3.3.6 常见问题处理

- 如果用户对没有执行过的慢 SQL 执行交互式诊断命令，则无法给出诊断结果。
- exporter 指标采集功能没有启动时运行慢 SQL 诊断功能，此时功能无法正常运行。
- 配置文件中的参数重新设置后，需要重新启动服务进程才能生效。

1.3.4AI4DB：趋势预测

1.3.4.1 概述

趋势预测功能模块主要实现基于历史时序数据预测未来时序变化趋势。该模块框架解耦，可以实现不同预测算法的灵活替换，并且该模块功能可以实现不同特征时序的算法自动选择，支持线性特征时序预测 LR 回归算法和非线性特征预测 ARIMA 算法。目前该模块可以覆盖线性时序、非线性时序和周期时序的准确预测。

1.3.4.2 环境部署

指标采集系统运行正常。

1.3.4.3 使用指导

假设用户已经初始化配置文件目录 confpath，则可以通过下述命令实现本特性的功能：

- 仅启动趋势预测功能，启动命令如下（更多用法参考对 service 子命令的说明）：

```
gs_dbmind service start -c confpath --only-run forecast
```

- 用户交互式趋势预测，查看 timestamps0 到 timestamps1 时间段内的预测结果，命令如下：

```
gs_dbmind component forecast show -c confpath --start-time timestamps0 --end-time
timestamps1
```

- 用户手动清理手动清理历史预测结果，命令如下：

```
gs_dbmind component forecast clean -c confpath --retention-days DAYS
```

- 停止已启动的服务，命令如下：

```
gs_dbmind service stop -c confpath
```

1.3.4.4 获取帮助

模块命令行说明：

```
gs_dbmind component forecast --help
usage: [-h] -c DIRECTORY [--metric-name METRIC_NAME] [--host HOST] [--start-time
TIMESTAMP_IN_MICROSECONDS] [--end-time TIMESTAMP_IN_MICROSECONDS]
[--retention-days DAYS]
{show,clean}
Workload Forecasting: Forecast monitoring metrics positional arguments:
{show,clean} choose a functionality to perform
optional arguments:
-h, --help       show this help message and exit
-c DIRECTORY, --conf DIRECTORY
set the directory of configuration files
--metric-name METRIC_NAME
set a metric name you want to retrieve
--host HOST set a host you want to retrieve
--start-time TIMESTAMP_IN_MICROSECONDS
```

```
set a start time of for retrieving
--end-time TIMESTAMP_IN_MICROSECONDS
set a end time of for retrieving
--retention-days DAYS
clear historical diagnosis results and set the maximum number of days to retain data
```

1.3.4.5 命令参考

表 1-13 gs_dbmind component forecast 命令行说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
action	动作参数	show：结果展示 clean：清理结果
-c，--conf	配置目录	-
--metric-name	指定显示指标名，用于过滤	-
--host	指定服务 IP 和端口号，用于过滤	-
--start-time	显示开始时间的时间戳，单位毫秒	非负实数
--end-time	显示结束时间的时间戳，单位毫秒	非负实数
--retention-days	保留结果天数	非负实数

1.3.4.6 常见问题处理

- 综合实际业务与模型预测效果考虑，趋势预测时长建议不要太短，建议大于 3600 秒（如果指标采集周期为 15 秒，则数据量为 240 个），否则预测效果会变差，并且数据量极小时，服务会异常，因此默认配置为 3600 秒
- 配置文件中的参数重新设置后，需要重新启动服务进程才能生效。

1.3.5SQLdiag：慢 SQL 发现

SQLdiag 是 GBase 8c 中 SQL 语句执行时长预测工具。现有的预测技术主要基于执行计

划的预测方法，但这些预测方案仅适用于 OLAP 场景且可以获取执行计划的任务，对于 OLTP 或者 HTAP 这样的快速、简单查询是没有太多使用价值的。与上述方案不同，SQLdiag 着眼于数据库的历史 SQL 语句，通过对历史 SQL 语句的执行表现进行总结归纳，将之再用于推断新的未知业务上。由于短时间内数据库 SQL 语句执行时长不会有太大的差距，SQLdiag 可以从历史数据中检测出与已执行 SQL 语句相似的语句结果集，并基于 SQL 向量化技术通过 SQL 模板化和深度学习这两种方法来预测 SQL 语句执行时长。本工具有如下优点：

(1) 不需要 SQL 语句的执行计划，对数据库性能不会有任何的影响。

(2) 使用场景广泛，目前业内的很多算法局限性比较高，比如只适用于 OLTP 或者 OLAP，而 SQLdiag 使用场景广泛。

(3) 该工具容易理解，只需要简单的操作，就可以训练出自己的预测模型。

本工具的典型应用场景是对一批即将上线的 SQL 语句进行透视，提前识别风险。

1.3.5.1 概述

SQLdiag 是一个 SQL 语句执行时间预测工具，通过模板化方法或者深度学习方法，实现在不获取 SQL 语句执行计划的前提下，依据语句逻辑相似度与历史执行记录，预测 SQL 语句的执行时间并以此发现异常 SQL。

1.3.5.2 使用指导

前提条件

- 需要保证用户提供训练数据。
- 如果用户通过提供的工具收集训练数据，则需要启用 WDR 功能，涉及到的参数为 track_stmt_stat_level 和 log_min_duration_statement，具体情况见下面小结。
- 为保证预测准确率，用户提供的历史语句日志应尽可能全面并具有代表性。

SQL 流水采集方法

本工具需要用户提前准备数据，训练数据格式如下，每个样本通过换行符分隔：

```
SQL,EXECUTION_TIME
```

预测数据格式如下：

```
SQL
```

其中 SQL 表示 SQL 语句的文本，EXECUTION_TIME 表示 SQL 语句的执行时间，样例数据

见 sample_data 中的 train.csv 和 predict.csv。

用户可以按照要求格式自己收集训练数据，工具也提供了脚本自动采集（load_sql_from_rd），该脚本基于 WDR 报告获取 SQL 信息，涉及到的参数有 log_min_duration_statement 和 track_stmt_stat_level:

- 其中 log_min_duration_statement 表示慢 SQL 阈值，如果为 0 则全量收集，时间单位为毫秒;
- track_stmt_stat_level 表示信息捕获的级别，建议设置为 track_stmt_stat_level='L0,L0'

参数开启后，可能占用一定的系统资源，但一般不大。持续的高并发场景可能产生 5%以内的损耗，数据库并发较低的场景，性能损耗可忽略。下述脚本存在于 sqldiag 根目录（$GAUSSHOME/bin/components/sqldiag）中。

```
使用脚本获取训练集方式：
load_sql_from_wdr.py [-h] --port PORT --start_time START_TIME
                                    --finish_time FINISH_TIME [--save_path SAVE_PATH]
例如：
    python load_sql_from_wdr.py --start_time "2021-04-25 00:00:00" --finish_time
"2021-04-26 14:00:00" --port 5432   --save_path ./data.csv
```

操作步骤

(1) 提供历史日志以供模型训练

(2) 进行训练与预测操作。

```
基于模板法的训练与预测：
    gs_dbmind component sqldiag [train, predict] -f FILE --model template --model-path
template_model_path
基于 DNN 的训练与预测：
    gs_dbmind component sqldiag [train, predict] -f FILE --model dnn --model-path
dnn_model_path
```

使用方法示例

使用提供的测试数据进行模板化训练:

```
gs_dbmind component sqldiag train -f ./sample_data/train.csv --model template
--model-path ./template
```

使用提供的测试数据进行模板化预测:

```
gs_dbmind component sqldiag predict -f ./sample_data/predict.csv --model template
--model-path ./template --predicted-file ./result/t_result
```

使用提供的测试数据进行模板化模型更新：

```
gs_dbmind component sqldiag finetune -f ./sample_data/train.csv --model template
--model-path ./template
```

使用提供的测试数据进行 DNN 训练：

```
gs_dbmind component sqldiag train -f ./sample_data/train.csv --model dnn
--model-path ./dnn_model
```

使用提供的测试数据进行 DNN 预测：

```
gs_dbmind component sqldiag predict -f ./sample_data/predict.csv --model dnn
--model-path ./dnn_model --predicted-file
```

使用提供的测试数据进行 DNN 模型更新：

```
gs_dbmind component sqldiag finetune -f ./sample_data/train.csv --model dnn
--model-path ./dnn_model
```

1.3.5.3 获取帮助

使用 SQLdiag 工具前，您可以通过以下指令获取帮助。

```
gs_dbmind component sqldiag --help
```

显示如下帮助信息：

```
usage:     [-h] [-f CSV_FILE] [--predicted-file PREDICTED_FILE]
                    [--model {template,dnn}] --model-path MODEL_PATH
                    [--config-file CONFIG_FILE]
                    {train,predict,finetune}
SQLdiag integrated by GBase 8c.
positional arguments:
  {train,predict,finetune}
                                The training mode is to perform feature extraction and
                                model training based on historical SQL statements. The
                                prediction mode is to predict the execution time of a
                                new SQL statement through the trained model.
optional arguments:
  -h, --help                    show this help message and exit
  -f CSV_FILE, --csv-file CSV_FILE
                                The data set for training or prediction. The file
                                format is CSV. If it is two columns, the format is
```

```
                        (SQL statement, duration time). If it is three
                        columns, the format is (timestamp of SQL statement
                        execution time, SQL statement, duration time).
--predicted-file PREDICTED_FILE
                        The file path to save the predicted result.
--model {template,dnn}
                        Choose the model model to use.
--model-path MODEL_PATH
                        The storage path of the model file, used to read or
                        save the model file.
--config-file CONFIG_FILE
```

1.3.5.4 命令参考

表 1-14 命令行参数说明

参数	参数说明	取值范围
-f	训练或预测文件位置	-
–predicted-file	预测结果存储位置	-
–model	模型选择	template, dnn
–model-path	训练模型存储位置	-

1.3.5.5 常见问题处理

- 训练场景失败：请检查历史日志文件路径是否正确，且文件格式符合上文规定。

预测场景失败：请检查模型路径是否正确。确保待预测负载文件格式正确。

1.3.6 SQL Rewriter: SQL 语句改写

1.3.6.1 概述

SQL Rewriter 是一个 SQL 改写工具，根据预先设定的规则，将查询语句转换为更为高效或更为规范的形式，使得查询效率得以提升。

说明：

- 本功能不适用包含子查询的语句；
- 本功能只支持 SELECT 语句和 DELETE 对整个表格删除的语句；
- 本功能包含 12 个改写规则，对不符合改写规则的语句，不会进行处理；
- 本功能会对原始查询语句和改写后语句进行屏幕输出，不建议对包含涉敏感信息的 SQL 语句进行改写；
- union 转 union all 规则避免了去重，从而提升了查询性能，所得结果有可能存在冗余；
- 语句中如包含 ‘order by’ + 指定列名或 ‘group by’ + 指定列名，无法适用 SelfJoin 规则。

1.3.6.2 使用指导

前提条件

数据库状态正常、连接正常。

使用方法示例

以 tpcc 数据库为例：

```
gs_dbmind component sql_rewriter 5030 tpcc queries.sql --db-host 127.0.0.1
--db-user myname --schema public
```

queries.sql 为需要改写的 SQL，内容如下：

```
select cfg_name from bmsql_config group by cfg_name having cfg_name='1';
delete from bmsql_config;
delete from bmsql_config where cfg_name='1';
```

结果为多个改写后的查询语句，显示在屏幕（无法改写的语句，显示为空），如下：

```
+-----------------------------------------------------------------------------
+-------------------------------+
| Raw SQL                                                                    |
Rewritten SQL                 |
+-----------------------------------------------------------------------------
+-------------------------------+
```

```
| select cfg_name from bmsql_config group by cfg_name having cfg_name='1'; |
SELECT cfg_name                                                           |
                                                                          |
| FROM bmsql_config                                                       |
                                                                          |
| WHERE cfg_name = '1';                                                   |
| delete from bmsql_config;                                               |
TRUNCATE TABLE bmsql_config;                                              |
| delete from bmsql_config where cfg_name='1';                            |
|
+-------------------------------------------------------------------------+
+--------------------------+
```

1.3.6.3 获取帮助

使用 SQL Rewriter 前，您可以通过以下指令获取帮助。

```
gs_dbmind component sql_rewriter --help
```

显示如下帮助信息：

```
usage:    [-h] [--db-host DB_HOST] [--db-user DB_USER] [--schema SCHEMA]
          db_port database file

SQL Rewriter

positional arguments:

  db_port                Port for database

  database               Name for database

  file                   File containing SQL statements which need to rewrite

optional arguments:

  -h, --help             show this help message and exit

  --db-host DB_HOST      Host for database

  --db-user DB_USER      Username for database log-in

  --schema SCHEMA        Schema name for the current business data
```

密码通过管道输入或交互式输入，对于免密用户，任意输入都可通过检验。

1.3.6.4 命令参考

表 1 命令行参数说明

参数名称	释义
db_port	数据库端口号
database	数据库名称
file	包含多个查询语句的文件路径
db-host	（可选）数据库主机号
db-user	（可选）数据库用户名
schema	（可选，模式为 public）模式

1.3.6.5 常见问题处理

SQL 无法改写：请查看 SQL 是否符合改写规则或 SQL 语法是否正确。

1.3.7 Anomaly detection：异常检测

1.3.7.1 概述

Anomaly detection 异常检测模块主要基于统计方法来实现时序数据来发现数据中存在的可能的异常情况。该模块框架解耦，可以实现不同异常检测算法的灵活替换，而且该模块功能可以根据时序数据的不同特征来自动选择算法，支持异常值检测、阈值检测、箱型图检测、梯度检测、增长率检测、波动率检测和状态转换检测。

1.3.7.2 使用指导

假设指标采集系统运行正常，并且用户已经初始化了配置文件目录 confpath，则可以通

过下述命令实现本特性的功能：

- 仅启动异常检测功能：

```
gs_dbmind service start --conf confpath --only-run anomaly_detection
```

- 对于某一指标，在全部节点上，从 timestamps1 到 timestamps1 时间段内的数据进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2
```

- 对于某一指标，在特定节点上，从 timestamps1 到 timestamps1 时间段内的数据进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --anomaly anomaly_type
```

- 对于某一指标，在全部节点上，从 timestamps1 到 timestamps1 时间段内的数据，以特定异常检测方式进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --anomaly anomaly_type
```

- 对于某一指标，在特定节点，从 timestamps1 到 timestamps1 时间段内的数据，以特定异常检测方式进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --anomaly anomaly_type
```

- 对于某一指标，在特定节点，从 timestamps1 到 timestamps1 时间段内的数据，以特定异常检测方式进行可视化展示：

```
gs_dbmind component anomaly_detection --conf confpath --action plot --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --anomaly anomaly_type
```

- 停止已启动的服务：

```
gs_dbmind service stop --conf confpath
```

说明： 在输入 anomaly detection 的参数时，start-time 至少要比 end-time 早 30 秒以上。

1.3.7.3 获取帮助

模块命令行说明：

```
gs_dbmind component anomaly_detection --help
```

显示如下帮助信息：

```
usage: anomaly_detection.py [-h] --action {overview,plot} -c CONF -m METRIC -s
                            START_TIME -e END_TIME [-H HOST] [-a ANOMALY]

Workload Anomaly detection: Anomaly detection of monitored metric.

optional arguments:
  -h, --help            show this help message and exit
  --action {overview,plot}
                        choose a functionality to perform
  -c CONF, --conf CONF  set the directory of configuration files
  -m METRIC, --metric METRIC
                        set the metric name you want to retrieve
  -s START_TIME, --start-time START_TIME
                        set the start time of for retrieving in ms
  -e END_TIME, --end-time END_TIME
                        set the end time of for retrieving in ms
  -H HOST, --host HOST  set a host of the metric, ip only or ip and port.
  -a ANOMALY, --anomaly ANOMALY
                        set a anomaly detector of the metric(increase_rate,
                        level_shift, spike, threshold)

Process finished with exit code 0
```

1.3.7.4 命令参考

表 1 命令行参数说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
--action	动作参数	overview：概览 plot：可视化
-c, --conf	配置文件目录	-
-m, --metric-name	指定显示指标名	-
-H, --host	指定数据来源地址信息，通过地址信息进行过滤	-ip 地址或者 ip 地址加端口号
-a, --anomaly	指定异常检测方式，用于过滤	-
-s, --start-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S.	正整数或日期时间格式
-e, --end-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S.	正整数或日期时间格式

1.3.7.5 常见问题处理

- 概览场景失败：请检查配置文件路径是否正确，且配置文件信息完整。检查指标名称是否准确，检查 host 地址是否正确，检查异常检测类型是否准确，检查起止时间内指标是否存在对应数据。
- 可视化场景失败：请检查配置文件路径是否正确，且配置文件信息完整。检查指标名称是否准确，检查 host 地址是否正确，检查异常检测类型是否准确，检查起止时间内指标是否存在对应数据。

1.3.8 Anomaly analysis：多指标关联

1.3.8.1 概述

Anomaly analysis 多指标关联模块主要基于分析时序数据的 Pearson 相关系数来发现与已知异常关联性最强的指标。该模块框架解耦，支持的时序数据库包括 Prometheus 和 InfluxDB。

1.3.8.2 使用指导

假设指标采集系统运行正常，并且用户已经初始化了配置文件目录 confpath，则可以通过下述命令实现本特性的功能：

对于某一指标，在特定节点上，分析其他指标与该指标从 timestamps1 到 timestamps1 时间段内的数据的相关性：

```
gs_dbmind component anomaly_analysis --conf confpath --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address
```

对于某一指标，在特定节点上，分析其他指标与该指标从 timestamps1 到 timestamps1 时间段内的数据的相关性，并将分析结果保存至为 csv 文件：

```
gs_dbmind component anomaly_analysis --conf confpath --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --csv-dump-path csv_path
```

说明：在输入 anomaly_analysis 的参数时，start-time 至少要比 end-time 早 30 秒以上。

1.3.8.3 获取帮助

- 模块命令行说明：

```
gs_dbmind component anomaly_detection --help
```

- 显示如下帮助信息：

```
usage: anomaly_analysis.py [-h] -c CONF -m METRIC -s START_TIME -e END_TIME -H
                           HOST [--csv-dump-path CSV_DUMP_PATH]

Workload Anomaly analysis: Anomaly analysis of monitored metric.

optional arguments:
```

```
  -h, --help                    show this help message and exit
  -c CONF, --conf CONF     set the directory of configuration files
  -m METRIC, --metric METRIC
                                set the metric name you want to retrieve
  -s START_TIME, --start-time START_TIME
                                set the start time of for retrieving in ms, supporting
                                UNIX-timestamp with microsecond or datetime format
  -e END_TIME, --end-time END_TIME
                                set the end time of for retrieving in ms, supporting
                                UNIX-timestamp with microsecond or datetime format
  -H HOST, --host HOST     set a host of the metric, ip only or ip and port.
  --csv-dump-path CSV_DUMP_PATH
                                dump the result csv file to the dump path if it is
                                specified.
```

1.3.8.4 命令参考

表 1 命令行参数说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
-c, --conf	配置文件目录	-
-m, --metric	指定显示指标名	-
-H, --host	指定数据来源地址信息，通过地址信息进行过滤	-ip 地址或者 ip 地址加端口号

参数	参数说明	取值范围
-s, --start-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S.	正整数或日期时间格式
-e, --end-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S.	正整数或日期时间格式
--csv-dump-path	结果导出的 csv 文件路径	-

1.3.8.5 常见问题处理

分析场景失败，请检查配置文件路径是否正确，且配置文件信息完整。检查指标名称是否准确，检查 host 地址是否正确，检查起止时间内指标是否存在对应数据。

1.4 ABO 优化器

1.4.1 智能基数估计

1.4.1.1 概述

智能基数估计使用库内贝叶斯网络模型对多列数据样本联和分布进行建模，从而能够对多列等值查询提供更加准确的基数估计。更加准确的基数估计能够显著提高优化器对于计划和算子的选择的准确性，从而提高数据库整体吞吐量。

1.4.1.2 前置条件

数据库运行正常，GUC 参数 enable_ai_stats 设置为 on，multi_stats_type 设置为'BAYESNET'或者'ALL'。

1.4.1.3 使用指导

- 设置采样方式为按照采样率采样，即设置 GUC 参数 default_statistics_target 为[-100, -1]之间的整数，表示采样百分比。
- 使用 ANALYZE(([column_name,])) 进行数据统计和模型创建。
- 输入查询，如果查询涉及到的等值查询列上有统计模型创建，那么会自动使用统计模型

进行选择率估计。

- 不再需要智能统计模型的时候，使用 ALTER TABLE [table_name] DELETE STATISTICS (([column_name,]))进行统计信息以及模型删除。

其他使用的方法详见 SQL 接口章节 ALTER TABLE 和 ANALYZE | ANALYSE。

1.4.1.4 最佳实践

生成如下数据表：

```
benchmark=# \d part;
id                  | integer                  |
p_brand             | character varying(256)   |
p_type              | character varying(256)   |
p_container         | character varying(256)   |
p_mfgr              | character varying(256)   |
```

插入 10,000,000 行数据：

```
benchmark=# select count(1) from part1;
10000000
```

在数据表上创建四种不同的多列索引：

```
benchmark=# select * from pg_indexes where tablename='part1';
public | part1 | brand_type_container | | CREATE INDEX brand_type_container ON part1 USING btree (p_brand, p_type, p_container) TABLESPACE pg_default
public | part1 | brand_type_mfgr | | CREATE INDEX brand_type_mfgr ON part1 USING btree (p_brand, p_type, p_mfgr) TABLESPACE pg_default
public | part1 | brand_container_mfgr | | CREATE INDEX brand_container_mfgr ON part1 USING btree (p_brand, p_container, p_mfgr) TABLESPACE pg_default
public | part1 | type_container_mfgr | | CREATE INDEX type_container_mfgr ON part1 USING btree (p_type, p_container, p_mfgr) TABLESPACE pg_default
```

针对数据表生成一批包含多列等值条件的查询，如下：

```
explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34' AND p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';
```

分别测试不创建多列统计信息和创建 ABO 统计信息场景下的执行计划：

```
benchmark=# explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34' AND p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';

Bitmap Heap Scan on part1 (cost=5.30..336.06 rows=17 width=56) (actual time=0.953..7.061 rows=103 loops=1)

  Recheck Cond: (((p_brand)::text = 'Brand#34'::text) AND ((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG CASE'::text))

  Filter: ((p_mfgr)::text = 'Manufacturer#2'::text)

  Rows Removed by Filter: 773

  Heap Blocks: exact=871

  -> Bitmap Index Scan on brand_type_container (cost=0.00..5.30 rows=84 width=0) (actual time=0.704..0.704 rows=876 loops=1)

        Index Cond: (((p_brand)::text = 'Brand#34'::text) AND ((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG CASE'::text))

Total runtime: 7.213 ms

benchmark=# explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34' AND p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';

Bitmap Heap Scan on part1 (cost=10.59..723.97 rows=210 width=56) (actual time=0.112..0.434 rows=103 loops=1)

  Recheck Cond: (((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG CASE'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))

  Filter: ((p_brand)::text = 'Brand#34'::text)

  Rows Removed by Filter: 64
```

```
  Heap Blocks: exact=167
  ->  Bitmap Index Scan on type_container_mfgr  (cost=0.00..10.54 rows=183 width=0) (actual time=0.081..0.081 rows=167 loops=1)
        Index Cond: (((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG CASE'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
Total runtime: 0.533 ms
```

通过以上操作可以看出，在这个场景下，ABO 基数估计加速了查询 10+倍。

1.4.1.5 常见问题处理

如果遇到异常场景导致模型无法创建，ABO 优化器会只创建传统统计信息，请根据对应的告警信息进行相应处理。

1.4.2 自适应计划选择

1.4.2.1 概述

自适应计划选择作用于使用通用缓存计划进行计划执行的场景。通过使用范围线性扩张进行缓存计划探索，通过范围覆盖匹配进行计划选择。自适应计划选择弥补了传统单一缓存计划无法根据查询条件参数进行变化带来的性能问题，并且避免了频繁调用查询优化。

1.4.2.2 前置条件

数据库运行正常，GUC 参数"enable_cachedplan_mgr"为 on，启动自适应计划选择功能。

1.4.2.3 使用指导

现网环境下，对存在缓存计划问题的 query 使用 hint 开启计划自适应管理能力：

```
select /*+ choose_adaptive_gplan */ * from tab where c1 = xxx;
```

JDBC 客户端默认会将以上带 hint 的 SQL 转换为 PBE 模型，并建立查询模板。除直接修改 SQL 外，hint 还可通过 sqlpatch 能力进行添加。

gsql 环境下，可以使用手动创建查询模板的模式进行：

```
prepare test_stmt as select /*+ choose_adaptive_gplan */ * from tab where c1 = $1;
```

1.4.2.4 最佳实践

多索引自适应选择支持，举例如下：

```
create table t1(c1 int, c2 int, c3 int, c4 varchar(32), c5 text);

create index t1_idx2 on t1(c1,c2,c3,c4);

create index t1_idx1 on t1(c1,c2,c3);

insert into t1( c1, c2, c3, c4, c5) SELECT (random()*(2*10^9))::integer , (random()*(2*10^9))::integer, (random()*(2*10^9))::integer, (random()*(2*10^9))::integer, repeat('abc', i%10) ::text from generate_series(1,1000000) i;

insert into t1( c1, c2, c3, c4, c5) SELECT (random()*1)::integer, (random()*1)::integer, (random()*1)::integer, (random()*(2*10^9))::integer, repeat('abc', i%10) ::text from generate_series(1,1000000) i;
```

性能对比：

随机参数：c1~ random(1, 20); c2~ random(1, 20); c3~ random(1, 20); c4 ~ random(2, 10000)

线程数 50，客户端 50，执行时长 60s

方法	语句	tps
gplan	prepare k as select * from t1 where c1=$1 and c2=$2 and c3=$3 and c4=$4;	35126
cplan	prepare k as select /*+ use_cplan */ * from t1 where c1=$1 and c2=$2 and c3=$3 and c4=$4;	75817
gplan 选择	prepare k as select /*+ choose_adaptive_gplan */ * from t1 where c1=$1 and c2=$2 and c3=$3 and c4=$4;	175681

1.4.2.5 常见问题处理

对于过于复杂的慢查询由于特征范围限制，可能无法使用本特性正确进行计划选择，建议直接使用 CPLAN 进行查询计划生成。

2 DB4AI：数据库驱动 AI

DB4AI 是指利用数据库的能力驱动 AI 任务，实现数据存储、技术栈的同构。通过在数据库内集成 AI 算法，令 GBase 8c 具备数据库原生 AI 计算引擎、模型管理、AI 算子、AI 原生执行计划的能力，为用户提供普惠 AI 技术。不同于传统的 AI 建模流程，DB4AI “一站式”建模可以解决数据在各平台的反复流转问题，同时简化开发流程，并可通过数据库规划出最优执行路径，让开发者更专注于具体业务和模型的调优上，具备同类产品不具备的易用性与性能优势。

2.1 原生 DB4AI 引擎

GBase 8c 支持了原生 DB4AI 能力，通过引入原生 AI 算子，简化操作流程，充分利用数据库优化器、执行器的优化与执行能力，获得高性能的数据库内模型训练能力。更简化的模型训练与预测流程、更高的性能表现，让开发者在更短时间内能更专注于模型的调优与数据分析上，而避免了碎片化的技术栈与冗余的代码实现。

关键字解析

表 2-1 DB4AI 语法及关键字

名称		描述
语法	CREATE MODEL	创建模型并进行训练，同时保存模型。
	PREDICT BY	利用已有模型进行推断。
	DROP MODEL	删除模型。
关键字	TARGET	训练/推断任务的目标列名。
	FEATURES	训练/推断任务的数据特征列名。
	MODEL	训练任务的模型名称。

使用指导

(1) 本版本支持的算法概述。

当前版本的 DB4AI 新增支持算法如下:

表 2-2 支持算法

优化算法	算法
GD	logistic_regression
	linear_regression
	svm_classification
	PCA
	multiclass
Kmeans	kmeans
xgboost	xgboost_regression_logistic
	xgboost_binary_logistic
	xgboost_regression_squarederror
	xgboost_regression_gamma

(2) 模型训练语法说明。

- CREATE MODEL

使用“CREATE MODEL”语句可以进行模型的创建和训练。模型训练 SQL 语句，选用公开数据集鸢尾花数据集 iris。

- 以 multiclass 为例，训练一个模型。从 tb_iris 训练集中指定 sepal_length, sepal_width,petal_length,petal_widt 为特征列，使用 multiclass 算法，创建并保存模型 iris_classification_model。

```
gsql=# CREATE MODEL iris_classification_model USING xgboost_regression_logistic
FEATURES sepal_length, sepal_width,petal_length,petal_width TARGET target_type < 2
FROM tb_iris_1 WITH nthread=4, max_depth=8;
MODEL CREATED. PROCESSED 1
```

上述命令中：

- CREATE MODEL 语句用于模型的训练和保存。
- USING 关键字指定算法名称。
- FEATURES 用于指定训练模模型的特征，需根据训练数据表的列名添加。
- TARGET 指定模型的训练目标，它可以是训练所需数据表的列名，也可以是一个表达式，例如：price > 10000。
- WITH 用于指定训练模型时的超参数。当超参未被用户进行设置的时候，框架会使用默认数值。

针对不同的算子，框架支持不同的超参组合：

表 2-3 算子支持的超参

算子	超参
GD (logistic_regression、linear_regression、svm_classification)	optimizer(char); verbose(bool); max_iterations(int); max_seconds(double); batch_size(int); learning_rate(double); decay(double); tolerance(double) 其中，SVM 限定超参 lambda(double)
Kmeans	max_iterations(int); num_centroids(int); tolerance(double); batch_size(int); num_features(int); distance_function(char); seeding_function(char); verbose(int);seed(int)
GD(pca)	batch_size(int);max_iterations(int);max_seconds(int);tolerance(float8);verbose(bool);number_components(int);seed(int)
GD(multiclass)	classifier(char)

	注意：multiclass 的其他超参种类取决于选择的分类器中类
xgboost_regression_logistic 、xgboost_binary_logistic 、xgboost_regression_squarederror 、xgboost_regression_gamma	batch_size(int);booster(char);tree_method(char);eval_metric(char*);seed(int);nthread(int);max_depth(int);gamma(float8);eta(float8);min_child_weight(int);verbosity(int)

当前各个超参数设置的默认值和取值范围如下：

表 2-4 超参的默认值以及取值范围

算子	超参(默认值)	取值范围	超参描述
GD: logistic_regression 、linear_regression 、svm_classification、pca	optimizer = gd（梯度下降法）	gd/ngd（自然梯度下降）	优化器
	verbose = false	T/F	日志显示
	max_iterations = 100	(0, 10000]	最大迭代次数
	max_seconds = 0 (不对运行时长设限制)	[0,INT_MAX_VALUE]	运行时长
	batch_size = 1000	(0, 1048575]	一次训练所选取的样本数
	learning_rate = 0.8	(0,	学习率

		DOUBLE_MAX_VALUE]	
	decay = 0.95	(0, DOUBLE_MAX_VALUE]	权值衰减率
	tolerance = 0.0005	(0, DOUBLE_MAX_VALUE]	公差
	seed = 0（对 seed 取随机值）	[0, INT_MAX_VALUE]	种子
	just for linear、SVM：kernel = “linear”	linear/gaussian/polynomial	核函数
	just for linear、SVM：components = MAX(2*features, 128)	[0, INT_MAX_VALUE]	高维空间维数
	just for linear、SVM：gamma = 0.5	(0, DOUBLE_MAX_VALUE]	gaussian 核函数参数
	just for linear、SVM：degree = 2	[2, 9]	polynomial 核函数参数
	just for linear、SVM：coef0 = 1.0	[0, DOUBLE_MAX_VALUE]	polynomial 核函数的参数
	just for SVM：lambda = 0.01	(0, DOUBLE_MAX_VALUE)	正则化参数
	just for pca：	（0，INT_MAX_VALUE]	降维的目标维

	number_components		度
GD: multiclass	classifier="svm_classification"	svm_classification\logistic_regression	多分类任务的分类器
Kmeans	max_iterations = 10	[1, 10000]	最大迭代次数
	num_centroids = 10	[1, 1000000]	簇的数目
	tolerance = 0.00001	(0,1]	中心点误差
	batch_size = 10	[1,1048575]	一次训练所选取的样本数
	num_features = 2	[1, INT_MAX_VALUE]	输入样本特征数
	distance_function = "L2_Squared"	L1\L2\L2_Squared\Linf	正则化方法
	seeding_function = "Random++"	"Random++"\"KMeans||"	初始化种子点方法
	verbose = 0U	{ 0, 1, 2 }	冗长模式
	seed = 0U	[0, INT_MAX_VALUE]	种子
xgboost:	n_iter=10	(0, 10000]	迭代次

xgboost_regression_logistic、 xgboost_binary_logistic、 xgboost_regression_gamma 、 xgboost_regression_squarederror			数
	batch_size=10000	(0, 1048575]	一次训练所选取的样本数
	booster=“gbtree”	gbtree\gblinear\dart	booster 种类
	tree_method=“auto”	auto\exact\approx\hist\gpu_hist 注意：gpu_hist 参数需要相应的库 GPU 版本，否则 DB4AI 平台不支持该值。	树构建算法
	eval_metric=“rmse”	rmse\rmsle\map\mae\auc\aucpr	验证数据的评估指标
	seed=0	[0, 100]	种子
	nthread=1	(0, MAX_MEMORY_LIMIT]	并发量
	max_depth=5	(0, MAX_MEMORY_LIMIT]	树的最大深度，该超参仅对树型 booster 生效。
	gamma=0.0	[0, 1]	叶节点上进行进一步分区所

			需的最小损失减少
	eta=0.3	[0, 1]	更新中使用的步长收缩，以防止过拟合
	min_child_weight=1	[0, INT_MAX_VALUE]	孩子节点中所需的实例权重的最小总和
	verbosity=1	0 (silent)\1 (warning)\2 (info)\3 (debug)	打印信息的详细程度
MAX_MEMORY_LIMIT = 最大内存加载的元组数量			
GS_MAX_COLS = 数据库单表最大属性数量			

- 模型保存成功，则返回创建成功信息：

```
MODEL CREATED. PROCESSED x
```

(3) 查看模型信息。

当训练完成后模型会被存储到系统表 gs_model_warehouse 中。系统表 gs_model_warehouse 可以查看到关于模型本身和训练过程的相关信息。

关于模型的详细描述信息以二进制的形式存储在系统表中，用户可用过使用函数 gs_explain_model 完成对模型的查看，语句如下：

```
gsql=# select * from gs_explain_model("iris_classification_model");
```

```
DB4AI MODEL
-------------------------------------------------------------
Name: iris_classification_model
Algorithm: xgboost_regression_logistic
Query: CREATE MODEL iris_classification_model
USING xgboost_regression_logistic
FEATURES sepal_length, sepal_width,petal_length,petal_width
TARGET target_type < 2
FROM tb_iris_1
WITH nthread=4, max_depth=8;
Return type: Float64
Pre-processing time: 0.000000
Execution time: 0.001443
Processed tuples: 78
Discarded tuples: 0
n_iter: 10
batch_size: 10000
max_depth: 8
min_child_weight: 1
gamma: 0.0000000000
eta: 0.3000000000
nthread: 4
verbosity: 1
seed: 0
booster: gbtree
tree_method: auto
eval_metric: rmse
rmse: 0.2648450136
model size: 4613
```

(4) 利用已存在的模型做推断任务。

使用“SELECT”和“PREDICT BY”关键字利用已有模型完成推断任务。

查询语法：SELECT…PREDICT BY…(FEATURES…)…FROM…;

```
gsql=# SELECT id, PREDICT BY iris_classification (FEATURES
sepal_length,sepal_width,petal_length,petal_width) as "PREDICT" FROM tb_iris limit 3;

id   | PREDICT
-----+---------
  84 |       2
  85 |       0
```

```
 86 |          0
(3 rows)
```

针对相同的推断任务，同一个模型的结果是大致稳定的。且基于相同的超参数和训练集训练的模型也具有稳定性，同时 AI 模型训练存在随机成分（每个 batch 的数据分布、随机梯度下降），所以不同的模型间的计算表现、结果允许存在小的差别。

(5) 查看执行计划。

使用 explain 语句可对“CREATE MODEL”和“PREDICT BY”的模型训练或预测过程中的执行计划进行分析。Explain 关键字后可直接拼接 CREATE MODEL/ PREDICT BY 语句（子句），也可接可选的参数，支持的参数如下：

表 2-5 EXPLAIN 支持的参数

参数名	描述
ANALYZE	布尔型变量，追加运行时间、循环次数等描述信息
VERBOSE	布尔型变量，控制训练的运行信息是否输出到客户端
COSTS	布尔型变量
CPU	布尔型变量
DETAIL	布尔型变量，不可用。
NODES	布尔型变量，不可用
NUM_NODES	布尔型变量，不可用
BUFFERS	布尔型变量
TIMING	布尔型变量
PLAN	布尔型变量
FORMAT	可选格式类型：TEXT / XML / JSON / YAML

示例：

```
gsql=# Explain CREATE MODEL patient_logisitic_regression USING logistic_regression
FEATURES second_attack, treatment TARGET trait_anxiety > 50 FROM patients WITH
batch_size=10, learning_rate = 0.05;
                                    QUERY PLAN
------------------------------------------------------------------------
 Train Model - logistic_regression    (cost=0.00..0.00 rows=0 width=0)
   ->  Materialize    (cost=0.00..41.08 rows=1776 width=12)
         ->  Seq Scan on patients    (cost=0.00..32.20 rows=1776 width=12)
(3 rows)
```

(6) 异常场景。

- 训练阶段。
  - 场景一：当超参数的设置超出取值范围，模型训练失败，返回 ERROR，并提示错误，例如：

```
gsql=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES
second_attack,treatment TARGET trait_anxiety    FROM patients WITH optimizer='aa';
ERROR:    Invalid hyperparameter value for optimizer. Valid values are: gd, ngd.
```

  - 场景二：当模型名称已存在，模型保存失败，返回 ERROR，并提示错误原因，例如：

```
gsql=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES
second_attack,treatment TARGET trait_anxiety    FROM patients;
ERROR:    The model name "patient_linear_regression" already exists in gs_model_warehouse.
```

  - 场景三：FEATURE 或者 TARGETS 列是*，返回 ERROR，并提示错误原因，例如：

```
gsql=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES *
TARGET trait_anxiety    FROM patients;
ERROR:    FEATURES clause cannot be *
------------------------------------------------------------------------------------------------------------------------
----
gsql=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES
second_attack,treatment TARGET *    FROM patients;
ERROR:    TARGET clause cannot be *
```

  - 场景四：对于无监督学习方法使用 TARGET 关键字，或者在监督学习方法中不适用 TARGET 关键字，均会返回 ERROR，并提示错误原因，例如：

```
gsql=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES
second_attack,treatment FROM patients;
```

```
ERROR:  Supervised ML algorithms require TARGET clause
-------------------------------------------------------------------------------------------------------------
CREATE MODEL patient_linear_regression USING linear_regression TARGET trait_anxiety
FROM patients;
ERROR:  Supervised ML algorithms require FEATURES clause
```

- 场景五:当进行分类任务时 TARGET 列的分类只有 1 种情况,会返回 ERROR,并提示错误原因，例如:

```
gsql=# CREATE MODEL ecoli_svmc USING multiclass FEATURES f1, f2, f3, f4, f5, f6, f7
TARGET cat FROM (SELECT * FROM db4ai_ecoli WHERE cat='cp');
ERROR:  At least two categories are needed
```

- 场景六: DB4AI 在训练过程中会过滤掉含有空值的数据，当参与训练的模型数据为空的时候，会返回 ERROR，并提示错误原因，例如:

```
gsql=# create model iris_classification_model using xgboost_regression_logistic features
message_regular target error_level from error_code;
ERROR:  Training data is empty, please check the input data.
```

- 场景七: DB4AI 的算法对于支持的数据类型是有限制的。当数据类型不在支持白名单中，会返回 ERROR，并提示非法的 oid，可通过 pg_type 查看 OID 确定非法的数据类型，例如:

```
gsql=# CREATE MODEL ecoli_svmc USING multiclass FEATURES f1, f2, f3, f4, f5, f6, f7,
cat TARGET cat FROM db4ai_ecoli ;
ERROR:  Oid type 1043 not yet supported
```

- 场景八: 当 GUC 参数 statement_timeout 设置了时长，训练超时执行的语句将被终止: 执行 CREATE MODEL 语句。训练集的大小、训练轮数(iteration)、提前终止条件(tolerance、max_seconds)、并行线程数(nthread)等参数都会影响训练时长。当时长超过数据库限制，语句被终止模型训练失败。

- 模型解析。
  - 场景九: 当模型名在系统表中查找不到，数据库会报 ERROR，例如:

```
gsql=# select gs_explain_model("ecoli_svmc");
ERROR:  column "ecoli_svmc" does not exist
```

- 推断阶段。
  - 场景十: 当模型名在系统表中查找不到，数据库会报 ERROR，例如:

```
gsql=# select id, PREDICT BY patient_logistic_regression (FEATURES
second_attack,treatment) FROM patients;
ERROR:  There is no model called "patient_logistic_regression".
```

- 场景十一：当做推断任务 FEATURES 的数据维度和数据类型与训练集存在不一致，将报 ERROR，并提示错误原因，例如：

```
gsql=# select id, PREDICT BY patient_linear_regression (FEATURES second_attack) FROM
patients;
ERROR:  Invalid number of features for prediction, provided 1, expected 2
CONTEXT:  referenced column: patient_linear_regression_pred
----------------------------------------------------------------------------------------------------------------
-----------------
gsql=# select id, PREDICT BY patient_linear_regression (FEATURES
1,second_attack,treatment) FROM patients;
ERROR:  Invalid number of features for prediction, provided 3, expected 2
CONTEXT:  referenced column: patient_linear_regression_pre
```

说明

DB4AI 特性需要读取数据参与计算，不适用于密态数据库等情况。

2.2 全流程 AI

传统的 AI 任务往往具有多个流程，如数据的收集过程包括数据的采集、数据清洗、数据存储等，在算法的训练过程中又包括数据的预处理、训练、模型的保存与管理等。其中，对于模型的训练过程，又包括超参数的调优过程。诸如此类机器学习模型生命周期的全过程，可大量集成于数据库内部。在距离数据存储侧最近处进行模型的训练、管理、优化等流程，在数据库端提供 SQL 语句式的开箱即用的 AI 全声明周期管理的功能，称之为全流程 AI。

GBase 8c 实现了部分全流程 AI 的功能，将在本章节中详细展开。

2.2.1PLPython Fenced 模式

在 fenced 模式中添加 plpython 非安全语言。在数据库编译时需要将 python 集成进数据库中，在 configure 阶段加入--with-python 选项。同时也可指定安装 plpython 的 python 路径，添加选项--with-includes='/python-dir=path'。

在启动数据库之前配置 GUC 参数 unix_socket_directory ，指定 unix_socket 进程间通讯的文件地址。用户需要提前在 user-set-dir-path 下创建文件夹，并将文件夹权限修改为可读可写可执行状态。

```
unix_socket_directory = '/user-set-dir-path'
```

配置完成，启动数据库。

将 plpython 加入数据库编译，并设置好 GUC 参数 unix_socket_directory 后，在启动数据库的过程中，自动创建 fenced-Master 进程。在数据库不进行 python 编译的情况下，fenced 模式需要手动拉起 master 进程，在 GUC 参数设置完成后，输入创建 master 进程命令。

启动 fenced-Master 进程，命令为：

```
gaussdb --fenced -k /user-set-dir-path -D /user-set-dir-path &
```

完成 fence 模式配置，针对 plpython-fenced UDF 数据库将在 fenced-worker 进程中执行 UDF 计算。

使用指导

- 创建 extension
  - 当编译的 plpython 为 python2 时：

```
gsql=# create extension plpythonu;
CREATE EXTENSION
```

  - 当编译的 plpython 为 python3 时：

```
gsql=# create extension plpython3u;
CREATE EXTENSION
```

下面示例是以 python2 为例。

- 创建 plpython-fenced UDF
- 查看 UDF 信息

```
gsql=# select * from pg_proc where proname='pymax';
gsql=# create or replace function pymax(a int, b int)
gsql-# returns INT
gsql-# language plpythonu fenced
gsql-# as $$
gsql$# import numpy
gsql$# if a > b:
gsql$#   return a;
gsql$# else:
gsql$#   return b;
gsql$# $$;
```

```
CREATE FUNCTION
```

- 运行 UDF
  - 创建一个数据表：

```
gsql=# create table temp (a int ,b int) ;
CREATE TABLE
gsql=# insert into temp values (1,2),(2,3),(3,4),(4,5),(5,6);
INSERT 0 5
```

  - 运行 UDF：

```
gsql=# select pymax(a,b) from temp;
pymax
-------
2
3
4
5
6
(5 rows)
```

2.2.2DB4AI-Snapshots 数据版本管理

DB4AI-Snapshots 是 DB4AI 模块用于管理数据集版本的功能。通过 DB4ai-Snapshots 组件，开发者可以简单、快速地进行特征筛选、类型转换等数据预处理操作，同时还可以像 git 一样对训练数据集进行版本控制。数据表快照创建成功后可以像视图一样进行使用，但是一经发布后，数据表快照便固化为不可变的静态数据，如需修改该数据表快照的内容，需要创建一个版本号不同的新数据表快照。

DB4AI-Snapshots 的生命周期

DB4AI-Snapshots 的状态包括 published、archived 以及 purged。其中，published 可以用于标记该 DB4AI-Snapshots 已经发布，可以进行使用。archived 表示当前 DB4AI-Snapshots 处于“存档期”，一般不进行新模型的训练，而是利用旧数据对新的模型进行验证。purged 则是该 DB4AI-Snapshots 已经被删除的状态，在数据库系统中无法再检索到。

需要注意的是快照管理功能是为了给用户提供统一的训练数据，不同团队成员可以使用给定的训练数据来重新训练机器学习模型，方便用户间协同。为此私有用户和三权分立状态(enableSeparationOfDuty=ON)等涉及不支持用户数据转写等情况将不支持 Snapshot 特性。

用户可以通过“CREATE SNAPSHOT”语句创建数据表快照，创建好的快照默认即为 published 状态。可以采用两种模式创建数据表快照，即为 MSS 以及 CSS 模式，它们可以通过 GUC 参数 db4ai_snapshot_mode 进行配置。对于 MSS 模式，它是采用物化算法进行实现的，存储了原始数据集的数据实体；CSS 则是基于相对计算算法实现的，存储的是数据的增量信息。数据表快照的元信息存储在 DB4AI 的系统目录中。可以通过 db4ai.snapshot 系统表查看到。

可以通过“ARCHIVE SNAPSHOT”语句将某一个数据表快照标记为 archived 状态，可以通过“PUBLISH SNAPSHOT”语句将其再度标记为 published 状态。标记数据表快照的状态，是为了帮助数据科学家进行团队合作使用的。

当一个数据表快照已经丧失存在价值时，可以通过“PURGE SNAPSHOT”语句删除它，以便永久删除其数据并恢复存储空间。

DB4AI-Snapshots 使用指导

(1) 创建表以及插入表数据。

数据库内存在已有的数据表，可根据该已有的数据表创建对应的数据表快照。为了后续演示，在此处新建一个名为 t1 的数据表，并向其中插入测试数据。

```
create table t1 (id int, name varchar);
insert into t1 values (1, 'zhangsan');
insert into t1 values (2, 'lisi');
insert into t1 values (3, 'wangwu');
insert into t1 values (4, 'lisa');
insert into t1 values (5, 'jack');
```

通过 SQL 语句，查询搭配数据表内容。

```
SELECT * FROM t1;
id |     name
----+----------
  1 | zhangsan
  2 | lisi
  3 | wangwu
  4 | lisa
  5 | jack
(5 rows)
```

(2) 使用 DB4AI-Snapshots。

- 创建 DB4AI-Snapshots

- 示例 1：CREATE SNAPSHOT…AS

示例如下，其中，默认版本分隔符为 “@”，默认子版本分割符为 “.”，该分割符可以分别通过 GUC 参数 db4ai_snapshot_version_delimiter 以及 db4ai_snapshot_version_separator 进行设置。

```
create snapshot s1@1.0 cgbaseent is 'first version' as select * from t1;
schema |   name
--------+--------
 public | s1@1.0
(1 row)
```

上述结果提示已经创建了数据表 s1 的快照，版本号为 1.0。创建好后的数据表快照可以像使用一般视图一样进行查询，但不支持通过“INSERT INTO”语句进行更新。例如下面几种语句都可以查询到数据表快照 s1 的对应版本 1.0 的内容：

```
SELECT * FROM s1@1.0;
SELECT * FROM public.s1@1.0;
SELECT * FROM public . s1 @ 1.0;
id |     name
----+----------
  1 | zhangsan
  2 | lisi
  3 | wangwu
  4 | lisa
  5 | jack
(5 rows)
```

可以通过下列 SQL 语句修改数据表 t1 的内容：

```
UPDATE t1 SET name = 'tom' where id = 4;
insert into t1 values (6, 'john');
insert into t1 values (7, 'tim');
```

再检索数据表 t1 的内容时，发现虽然数据表 t1 的内容已经发生变化，但是数据表快照 s1@1.0 版本的查询结果并未发生变化。由于数据表 t1 的数据已经发生了改变，如果将当前数据表的内容作为版本 2.0，则可创建快照 s1@2.0，创建的 SQL 语句如下：

```
create snapshot s1@2.0 as select * from t1;
```

通过上述例子，我们可以发现，数据表快照可以固化数据表的内容，避免中途对数据的改动造成机器学习模型训练时的不稳定，同时可以避免多用户同时访问、修改同一个表时造成的锁冲突。

- 示例 2：CREATE SNAPSHOT…FROM

SQL 语句可以对一个已经创建好的数据表快照进行继承，利用在此基础上进行的数据修改产生一个新的数据表快照。例如：

```
create snapshot s1@3.0 from @1.0 cgbaseent is 'inherits from @1.0' using (INSERT
VALUES(6, 'john'), (7, 'tim'); DELETE WHERE id = 1);
schema |    name
--------+--------
 public | s1@3.0
(1 row)
```

其中，“@”为数据表快照的版本分隔符，from 子句后加上已存在的数据表快照，用法为“@”+版本号，USING 关键字后加入可选的几个操作关键字（INSERT …/UPDATE …/DELETE …/ALTER …），其中 “INSERT INTO”以及“DELETE FROM”语句中的“INTO”、“FROM”等与数据表快照名字相关联的子句可以省略，具体可以参考 AI 特性函数。

示例中，基于前述 s1@1.0 快照，插入 2 条数据，删除 1 条新的数据，新生成的快照 s1@3.0，检索该 s1@3.0：

```
SELECT * FROM s1@3.0;
id |     name
----+----------
  2 | lisi
  3 | wangwu
  4 | lisa
  5 | jack
  6 | john
  7 | tim
(7 rows)
```

- 删除数据表快照 SNAPSHOT

```
purge snapshot s1@3.0;
schema |    name
--------+--------
 public | s1@3.0
(1 row)
```

此时，已经无法再从 s1@3.0 中检索到数据了，同时该数据表快照在 db4ai.snapshot 视图中的记录也会被清除。删除该版本的数据表快照不会影响其他版本的数据表快照。

- 从数据表快照中采样

示例：从 snapshot s1 中抽取数据，使用 0.5 抽样率。

```
sample snapshot s1@2.0 stratify by name as nick at ratio .5;
schema |       name
--------+------------
 public | s1nick@2.0
(1 row)
```

可以利用该功能创建训练集与测试集，例如：

```
SAMPLE SNAPSHOT s1@2.0    STRATIFY BY name AS _test AT RATIO .2, AS _train AT
RATIO .8 CgbaseENT IS 'training';
schema |           name
--------+----------------
 public | s1_test@2.0
 public | s1_train@2.0
(2 rows)
```

- 发布数据表快照

采用下述 SQL 语句将数据表快照 s1@2.0 标记为 published 状态：

```
publish snapshot s1@2.0;
schema |    name
--------+--------
 public | s1@2.0
(1 row)
```

- 存档数据表快照

采用下述语句可以将数据表快照标记为 archived 状态：

```
archive snapshot s1@2.0;
schema |    name
--------+--------
 public | s1@2.0
(1 row)
```

可以通过 db4ai-snapshots 提供的视图查看当前数据表快照的状态以及其他信息：

```
select * from db4ai.snapshot;
id | parent_id | matrix_id | root_id | schema |       name       | owner    |            cgbaseands
| cgbaseent | published | archived |                created | row_count
----+-----------+-----------+---------+--------+------------+--------+-----------------------------------
--+---------+-----------+----------+---------------------------+-----------
```

```
  1 |                |                |           1 | public | s1@2.0         | gbase | {"select *","from t1
where id > 3",NULL} |                | t              | f              | 2021-04-17 09:24:11.139868 |
2
  2 |              1 |                |           1 | public | s1nick@2.0 | gbase | {"SAMPLE nick .5
{name}"}                        |                | f              | f              | 2021-04-17 10:02:31.73923
|              0
```

(3) 异常场景

数据表或 db4ai-snapshots 不存在时。

```
purge snapshot s1nick@2.0;
publish snapshot s1nick@2.0;
---------
ERROR:  snapshot public."s1nick@2.0" does not exist
CONTEXT:  PL/pgSQL function db4ai.publish_snapshot(name,name) line 11 at assignment

archive snapshot s1nick@2.0;
----------
ERROR:  snapshot public."s1nick@2.0" does not exist
CONTEXT:  PL/pgSQL function db4ai.archive_snapshot(name,name) line 11 at assignment
```

删除 snapshot 时，有依赖该快照的其他 snapshot，需先确保删除对本快照所依赖的其他快照。

```
purge snapshot s1@1.0;
ERROR:  cannot purge root snapshot 'public."s1@1.0"' having dependent snapshots
HINT:  purge all dependent snapshots first
CONTEXT:  referenced column: purge_snapshot_internal
SQL statement "SELECT db4ai.purge_snapshot_internal(i_schema, i_name)"
PL/pgSQL function db4ai.purge_snapshot(name,name) line 71 at PERFORM
```

(4) 相关 GUC 参数

- db4ai_snapshot_mode：

Snapshot 有 2 种模式：MSS（物化模式，存储数据实体）和 CSS（计算模式，存储增量信息）。Snapshot 可在 MSS 和 CSS 之间切换快照模式，默认是 MSS 模式。

- db4ai_snapshot_version_delimiter：

该参数为数据表快照版本分隔符。“@”为数据表快照的默认版本分隔符。

- db4ai_snapshot_version_separator

该参数为数据表快照子版本分隔符。“.”为数据表快照的默认版本分隔符。

(5) DB4AI Schema 下的数据表快照详情 db4ai.snapshot。

```
gsql=# \d db4ai.snapshot
                              Table "db4ai.snapshot"
  Column     |              Type              |            Modifiers
-----------+----------------------------+---------------------------
 id             | bigint                          |
 parent_id | bigint                          |
 matrix_id | bigint                          |
 root_id      | bigint                          |
 schema      | name                              | not null
 name          | name                              | not null
 owner         | name                              | not null
 cgbaseands   | text[]                          | not null
 cgbaseent     | text                            |
 published | boolean                         | not null default false
 archived    | boolean                         | not null default false
 created      | timestamp without time zone | default pg_systimestamp()
 row_count | bigint                          | not null
Indexes:
    "snapshot_pkey" PRIMARY KEY, btree (schema, name) TABLESPACE pg_default
    "snapshot_id_key" UNIQUE CONSTRAINT, btree (id) TABLESPACE pg_default
```

说明

命名空间 DB4AI 是本功能的私有域，不支持在 DB4AI 的命令空间下创建函数索引(functional index）。

2.2.3 DB4AI-Query：模型训练和推断

GBase 8c 当前版本支持了原生 DB4AI 能力，通过引入原生 AI 算子，简化操作流程，充分利用数据库优化器、执行器的优化与执行能力，获得高性能的数据库内模型训练能力。更简化的模型训练与预测流程、更高的性能表现，让开发者在更短时间内能更专注于模型的调优与数据分析上，而避免了碎片化的技术栈与冗余的代码实现。

关键字解析

表 1 DB4AI 语法及关键字

	名称	描述

	名称	描述
语句	CREATE MODEL	创建模型并进行训练，同时保存模型。
	PREDICT BY	利用已有模型进行推断。
关键字	TARGET	训练/推断任务的目标列名。
	FEATURES	训练/推断任务的数据特征列名。
	MODEL	训练任务的模型名称。

使用指导

1. 本版本支持的算法概述。

当前版本的 DB4AI 支持基于 SGD 算子的逻辑回归（目前支持二分类任务）、线性回归和支持向量机算法（分类任务），以及基于 K-Means 算子的 Kmeans 聚类算法。

2. 模型训练语法说明。

- CREATE MODEL

使用“CREATE MODEL”语句可以进行模型的创建和训练。模型训练 SQL 语句，现有一个数据集为 kmeans_2d，该表的数据内容如下：

```
Postgres=# select * from kmeans_2d;
 id |                position
----+---------------------------------
  1 | {74.5268815685995,88.2141939294524}
  2 | {70.9565760521218,98.8114827475511}
  3 | {76.2756086327136,23.8387574302033}
  4 | {17.8495847294107,81.8449544720352}
  5 | {81.2175785354339,57.1677675866522}
  6 | {53.97752255667,49.3158342130482}
  7 | {93.2475341879763,86.934042100329}
  8 | {72.7659293473698,19.7020415100269}
  9 | {16.5800288529135,75.7475957670249}
 10 | {81.8520747194998,40.3476078575477}
```

```
 11 | {76.796671198681,86.3827232690528}
 12 | {59.9231450678781,90.9907738864422}
 13 | {70.161884885747,19.7427458665334}
 14 | {11.1269539105706,70.9988166182302}
 15 | {80.5005071521737,65.2822235273197}
 16 | {54.7030725912191,52.151339428965}
 17 | {103.059707058128,80.8419883321039}
 18 | {85.3574452036992,14.9910179991275}
 19 | {28.6501615960151,76.6922890325077}
 20 | {69.7285806713626,49.5416352967732}
(20 rows)
```

该表的字段 position 的数据类型为 double precision[].

- 以 Kmeans 为例，训练一个模型。从 kmeans_2d 训练集中指定 position 为特征列，使用 kmeans 算法，创建并保存模型 point_kmeans。

```
Postgres=# CREATE MODEL point_kmeans USING kmeans FEATURES position FROM kmeans_2d WITH num_centroids=3;
NOTICE:  Hyperparameter max_iterations takes value DEFAULT (10)
NOTICE:  Hyperparameter num_centroids takes value 3
NOTICE:  Hyperparameter tolerance takes value DEFAULT (0.000010)
NOTICE:  Hyperparameter batch_size takes value DEFAULT (10)
NOTICE:  Hyperparameter num_features takes value DEFAULT (2)
NOTICE:  Hyperparameter distance_function takes value DEFAULT (L2_Squared)
NOTICE:  Hyperparameter seeding_function takes value DEFAULT (Random++)
NOTICE:  Hyperparameter verbose takes value DEFAULT (0)
NOTICE:  Hyperparameter seed takes value DEFAULT (0)
MODEL CREATED. PROCESSED 1
```

上述命令中：

1. “CREATE MODEL”语句用于模型的训练和保存。
2. USING 关键字指定算法名称。
3. FEATURES 用于指定训练模模型的特征，需根据训练数据表的列名添加。
4. TARGET 指定模型的训练目标，它可以是训练所需数据表的列名，也可以是一个表达式，例如：price > 10000。

5. WITH 用于指定训练模型时的超参数。当超参未被用户进行设置的时候，框架会使用默认数值。

针对不同的算子，框架支持不同的超参组合：

表 2 算子支持的超参

算子	超参

GD
 	
(logistic_regression、linear_regression、svm_classification)
	optimizer(char*); verbose(bool);
	max_iterations(int); max_seconds(double);
	batch_size(int); learning_rate(double);
	decay(double); tolerance(double)
	
	其中，SVM 限定超参 lambda(double)

Kmeans	max_iterations(int); num_centroids(int);
	tolerance(double); batch_size(int);
	num_features(int); distance_function(char*);
	seeding_function(char*); verbose(int);seed(int)

当前各个超参数设置的默认值和取值范围如下：

表 3 超参的默认值以及取值范围

算子	超参（默认值）	取值范围	超参描述
GD (logistic_regression、linear_regression、svm_classification)	optimizer = gd（梯度下降法）	gd/ngd（自然梯度下降）	优化器
	verbose = false	T/F	日志显示
	max_iterations = 100	(0, INT_MAX_VALUE]	最大迭代次数
	max_seconds = 0（不对运行时长设限制）	[0, INT_MAX_VALUE]	运行时长

算子	超参（默认值）	取值范围	超参描述
	batch_size = 1000	(0, MAX_MEMORY_LIMIT]	一次训练所选取的样本数
	learning_rate = 0.8	(0, DOUBLE_MAX_VALUE]	学习率
	decay = 0.95	(0, DOUBLE_MAX_VALUE]	权值衰减率
	tolerance = 0.0005	(0, DOUBLE_MAX_VALUE]	公差
	seed = 0（对 seed 取随机值）	[0, INT_MAX_VALUE]	种子
	just for SVM: lambda = 0.01	(0, DOUBLE_MAX_VALUE)	正则化参数
Kmeans	max_iterations = 10	[1, INT_MAX_VALUE]	最大迭代次数
	num_centroids = 10	[1, MAX_MEMORY_LIMIT]	簇的数目
	tolerance = 0.00001	(0,1)	中心点误差
	batch_size = 10	[1, MAX_MEMORY_LIMIT]	一次训练所选取的样本数
	num_features = 2	[1, GS_MAX_COLS]	输入样本特征数

算子	超参（默认值）	取值范围	超参描述
	distance_function = ″L2_Squared″	L1\L2\L2_Squared \Linf	正则化方法
	seeding_function = ″Random++″	″Random++″\″KMeans||″	初始化种子点方法
	verbose = 0U	{ 0, 1, 2 }	冗长模式
	seed = 0U	[0, INT_MAX_VALUE]	种子
MAX_MEMORY_LIMIT = 最大内存加载的元组数量			
GS_MAX_COLS = 数据库单表最大属性数量			

- 模型保存成功，则返回创建成功信息：

```
MODEL CREATED. PROCESSED x
```

3. 查看模型信息。

当训练完成后模型会被存储到系统表 gs_model_warehouse 中。系统表 gs_model_warehouse 可以查看到关于模型本身和训练过程的相关信息。

用户可以通过查看系统表的方式查看模型，例如查看模型名为“point_kmeans”的 SQL 语句如下：

```
postgres=# select * from gs_model_warehouse where modelname='point_kmeans';
-[                                                                 RECORD
1 ]---------+---------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
--------------------------------------------------------------
modelname                | point_kmeans
modelowner               | 10
createtime               | 2021-04-30 17:30:39.59044
processedtuples          | 20
discardedtuples          | 0
pre_process_time         | 6.2001e-05
exec_time                | .000185272
iterations               | 5
outputtype               | 23
modeltype                | kmeans
query                    | CREATE MODEL point_kmeans USING kmeans FEATURES
position FROM kmeans_2d WITH num_centroids=3;
modeldata                |
weight                   |
hyperparametersnames                                                  |
{max_iterations,num_centroids,tolerance,batch_size,num_features,dista
nce_function,seeding_function,verbose,seed}
hyperparametersvalues | {10,3,1e-05,10,2,L2_Squared,Random++,0,0}
hyperparametersoids      | {23,23,701,23,23,1043,1043,23,23}
coefnames                                                             |
{original_num_centroids,actual_num_centroids,dimension,distance_function_id,see
d,coordinates}
coefvalues                                                            |
{3,3,2,2,572368998,"(77.282589,23.724434)(74.421616,73.239455)(18.551682,76.32
0914)"}
coefoids                 |
trainingscoresname       |
trainingscoresvalue      |
modeldescribe                                                         |
{"id:1,objective_function:542.851169,avg_distance_to_centroid:108.570234,min_d
istance_to_centroid:1.027078,max_distance_to_centroid:297.210108,std_dev_distan
ce_to_centroid:105.053257,cluster_size:5","id:2,objective_function:5825.982139,
avg_distance_to_centroid:529.634740,min_distance_to_centroid:100.270449,max_dis
tance_to_centroid:990.300588,std_dev_distance_to_centroid:285.915094,cluster_si
ze:11","id:3,objective_function:220.792591,avg_distance_to_centroid:55.198148,
min_distance_to_centroid:4.216111,max_distance_to_centroid:102.117204,std_dev_d
istance_to_centroid:39.319118,cluster_size:4"}
```

4. 利用已存在的模型做推断任务。

使用“SELECT”和“PREDICT BY”关键字利用已有模型完成推断任务。

查询语法：SELECT…PREDICT BY…(FEATURES…)…FROM…;

```
postgres=# SELECT id, PREDICT BY point_kmeans (FEATURES position) as pos FROM
(select * from kmeans_2d limit 10);
 id | pos
----+-----
  1 |   2
  2 |   2
  3 |   1
  4 |   3
  5 |   2
  6 |   2
  7 |   2
  8 |   1
  9 |   3
 10 |   1
(10 rows)
```

针对相同的推断任务，同一个模型的结果是稳定的。且基于相同的超参数和训练集训练的模型也具有稳定性，同时 AI 模型训练存在随机成分（每个 batch 的数据分布、随机梯度下降），所以不同的模型间的计算表现、结果允许存在小的差别。

5. 查看执行计划。

使用 explain 语句可对“CREATE MODEL”和“PREDICT BY”的模型训练或预测过程中的执行计划进行分析。Explain 关键字后可直接拼接 CREATE MODEL/ PREDICT BY 语句（子句），也可接可选的参数，支持的参数如下：

表 4 EXPLAIN 支持的参数

参数名	描述
ANALYZE	布尔型变量，追加运行时间、循环次数等描述信息
VERBOSE	布尔型变量，控制训练的运行信息是否输出到客户端

参数名	描述
COSTS	布尔型变量
CPU	布尔型变量
DETAIL	布尔型变量，不可用
NODES	布尔型变量，不可用
NUM_NODES	布尔型变量，不可用
BUFFERS	布尔型变量
TIMING	布尔型变量
PLAN	布尔型变量
FORMAT	可选格式类型：TEXT / XML / JSON / YAML

示例：

```
postgres=# Explain CREATE MODEL patient_logisitic_regression USING
logistic_regression FEATURES second_attack, treatment TARGET trait_anxiety > 50
FROM patients WITH batch_size=10, learning_rate = 0.05;
NOTICE:  Hyperparameter batch_size takes value 10
NOTICE:  Hyperparameter decay takes value DEFAULT (0.950000)
NOTICE:  Hyperparameter learning_rate takes value 0.050000
NOTICE:  Hyperparameter max_iterations takes value DEFAULT (100)
NOTICE:  Hyperparameter max_seconds takes value DEFAULT (0)
NOTICE:  Hyperparameter optimizer takes value DEFAULT (gd)
NOTICE:  Hyperparameter tolerance takes value DEFAULT (0.000500)
```

```
NOTICE:  Hyperparameter seed takes value DEFAULT (0)
NOTICE:  Hyperparameter verbose takes value DEFAULT (FALSE)
NOTICE:  GD shuffle cache size 212369
                              QUERY PLAN
-----------------------------------------------------------------------
 Gradient Descent  (cost=0.00..0.00 rows=0 width=0)
   ->  Seq Scan on patients  (cost=0.00..32.20 rows=1776 width=12)
(2 rows)
```

6. 异常场景。

- 训练阶段。

1. 场景一：当超参数的设置超出取值范围，模型训练失败，返回 ERROR，并提示错误，例如：

```
postgres=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES second_attack,treatment TARGET trait_anxiety FROM patients WITH optimizer='aa';
NOTICE:  Hyperparameter batch_size takes value DEFAULT (1000)
NOTICE:  Hyperparameter decay takes value DEFAULT (0.950000)
NOTICE:  Hyperparameter learning_rate takes value DEFAULT (0.800000)
NOTICE:  Hyperparameter max_iterations takes value DEFAULT (100)
NOTICE:  Hyperparameter max_seconds takes value DEFAULT (0)
NOTICE:  Hyperparameter optimizer takes value aa
ERROR:  Invalid hyperparameter value for optimizer. Valid values are: gd, ngd. (default is gd)
```

2. 场景二：当模型名称已存在，模型保存失败，返回 ERROR，并提示错误原因：

```
postgres=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES second_attack,treatment TARGET trait_anxiety FROM patients;
NOTICE:  Hyperparameter batch_size takes value DEFAULT (1000)
NOTICE:  Hyperparameter decay takes value DEFAULT (0.950000)
NOTICE:  Hyperparameter learning_rate takes value DEFAULT (0.800000)
NOTICE:  Hyperparameter max_iterations takes value DEFAULT (100)
NOTICE:  Hyperparameter max_seconds takes value DEFAULT (0)
NOTICE:  Hyperparameter optimizer takes value DEFAULT (gd)
NOTICE:  Hyperparameter tolerance takes value DEFAULT (0.000500)
NOTICE:  Hyperparameter seed takes value DEFAULT (0)
NOTICE:  Hyperparameter verbose takes value DEFAULT (FALSE)
```

```
NOTICE:  GD shuffle cache size 5502
ERROR:  The model name "patient_linear_regression" already exists in gs_model_warehouse.
```

3. 场景三：FEATURE 或者 TARGETS 列是*，返回 ERROR，并提示错误原因：

```
postgres=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES *  TARGET trait_anxiety  FROM patients;
ERROR:  FEATURES clause cannot be *
----------------------------------------------------------------------------------------------------------------------------、
postgres=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES second_attack,treatment TARGET *  FROM patients;
ERROR:  TARGET clause cannot be *
```

4. 场景四：对于无监督学习方法使用 TARGET 关键字，或者在监督学习方法中不适用 TARGET 关键字，均会返回 ERROR，并提示错误原因：

```
postgres=# CREATE MODEL patient_linear_regression USING linear_regression FEATURES second_attack,treatment FROM patients;
ERROR:  Supervised ML algorithms require TARGET clause
----------------------------------------------------------------------------------------------------------------------------
CREATE MODEL patient_linear_regression USING linear_regression TARGET trait_anxiety  FROM patients;  ERROR:  Supervised ML algorithms require FEATURES clause
```

5. 场景五：当 GUC 参数 statement_timeout 设置了时长，训练超时执行的语句将被终止：执行 CREATE MODEL 语句。训练集的大小、训练轮数（iteration）、提前终止条件（tolerance、max_seconds）、并行线程数（nthread）等参数都会影响训练时长。当时长超过数据库限制，语句被终止模型训练失败。

- 推断阶段。

1. 场景六：当模型名在系统表中查找不到，数据库会报 ERROR：

```
postgres=# select id, PREDICT BY patient_logistic_regression (FEATURES second_attack,treatment) FROM patients;
ERROR:  There is no model called "patient_logistic_regression".
```

2. 场景七：当做推断任务 FEATURES 的数据维度和数据类型与训练集存在不一致，将报 ERROR，并提示错误原因，例如：

```
postgres=# select id, PREDICT BY patient_linear_regression (FEATURES second_attack) FROM patients;
ERROR:  Invalid number of features for prediction, provided 1, expected 2
CONTEXT:  referenced column: patient_linear_regression_pred
-------------------------------------------------------------------------------------------------------------------------------
postgres=# select id, PREDICT BY patient_linear_regression (FEATURES 1,second_attack,treatment) FROM patients;
ERROR:  Invalid number of features for prediction, provided 3, expected 2
CONTEXT:  referenced column: patient_linear_regression_pre
```

GBASE®

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微信二维码

技术支持热线：400-013-9696