《数据库系统原理》课程设计指导书

TPC Benchmark 电商数据管理系统

北京邮电大学计算机学院

2025 年 2 月

目 录

1. 引言

TPC（Transaction Processing Performance Council，事务处理性能委员会）专注于开发以数据为中心的基准标准，针对不同的系统和应用类型提出了不同的测试基准 TPC-H、TPC-C Benchmark。这些测试基准包括针对不同系统和应用所设计数据模型，以及与系统在实际应用中对应的数据量单位。通过将数据导入被测系统，根据应用需求进行模拟操作，根据对应的评价指标可以评价系统的性能。测试基准包括测试数据、测试方法/程序等。

TPC-H 是面向决策支持（DS）和联机/在线分析处理 OLAP 的测试标准，模拟了供应商和采购商之间的交易行为，包括：1）面向电子商务领域的 8 张关系表，2）生成测试基准数据集的数据生成程序，利用该程序可以为 8 张表生成测试数据，3）用于评估 OLAP 能力的基准程序，即针对 8 张表的 22 条分析型 SQL 查询语句，涉及分组、聚集等复杂多表查询处理和计算。

TPC-C 是面向联机/在线事务处理 OLTP 的测试标准，包括：1）面向电商领域的 10 张关系表，2）生成测试基准数据集的数据生成程序，利用该程序可以为 10 张表生成测试数据，3）用于评估 OLTP 能力的基准程序，即针对 10 张表的 5 个事务。每个事务以嵌入式 SQL 方式实现，编程语言为 C 和 Java，包括多个 SQL 查询语句，涉及对数据库表的 select、update、insert、delete 等操作。

在《数据库系统原理》课程实验中，同学按照要求，建立了 TPC-H 测试基准数据库，导入了测试数据，完成了数据查询、完整性约束、数据库接口、查询优化、事务管理等实验。

本次课程设计是对课程实验的扩展和延申。要求学生以小组为单位（每组 3-4 人），根据课堂教学所讲授的数据库系统的设计过程和开发方法，在《数据库系统原理》课程实验已经建立的 TPC-H Benchmark 电商数据库基础上，

（1）采用数据库系统平台（如 OceanBase, openGauss, GaussDB(MySQL)，或 PostgreSQL、MySQL、SQL Server 等，并根据需要适当增加 TPC-C 数据库表以支持执行 TPC-C 事务，导入 TPC-H、TPC-C 测试数据；

（2）采用 Java、Python、C、C++编程语言，开发客户端和服务器端业务处理逻辑和事务；

（3）【选做】可以采用 Web 和数据库开发框架和工具，如 Springboot + VUE、Springboot + freemarker 模板，MyBatis，或其它开发工具（如 Qt），设计开发具有（1）三层 B/S 结构、或（2）两层 C/S 架构的数据库应用系统——TPC Benchmark 电商数据管理系统（原型）。

该原型支持对 TPC-H、TPC-H Benchmark 数据库中各类电商数据的导入导出、简单查询、基于 TPC-H 的统计分析、基于 TPC-C 的联机事务处理。用户可以分析评估索引对查询、统计性能的影响，观察 SQL 语句的查询执行计划，观察分析数据库事务的并发执行过程。

2. TPC-H 测试基准

2.1 TPC-H 关系表

TPC-H 测试基准模拟了供应商和采购商之间的交易行为，包括：1）用于评估在线分析处理的基准程序，即针对 8 张表的 22 条分析型查询；2）生成测试基准数据集的数据生成程序，利用该程序可以为 8 张表生成测试数据，数据规模由外部参数控制。

TPC-H 是面向决策支持（DS）测试标准，采用雪花模型，涵盖了电子商务领域的 8 张关系表：订单表 ORDERS，区域表 REGION，国家表 NATION，供应商表 SUPPLIER，零部件表 PART，零部件供应表 PARTSUPP，客户表 CUSTOMER，订单明细表 LINEITEM。

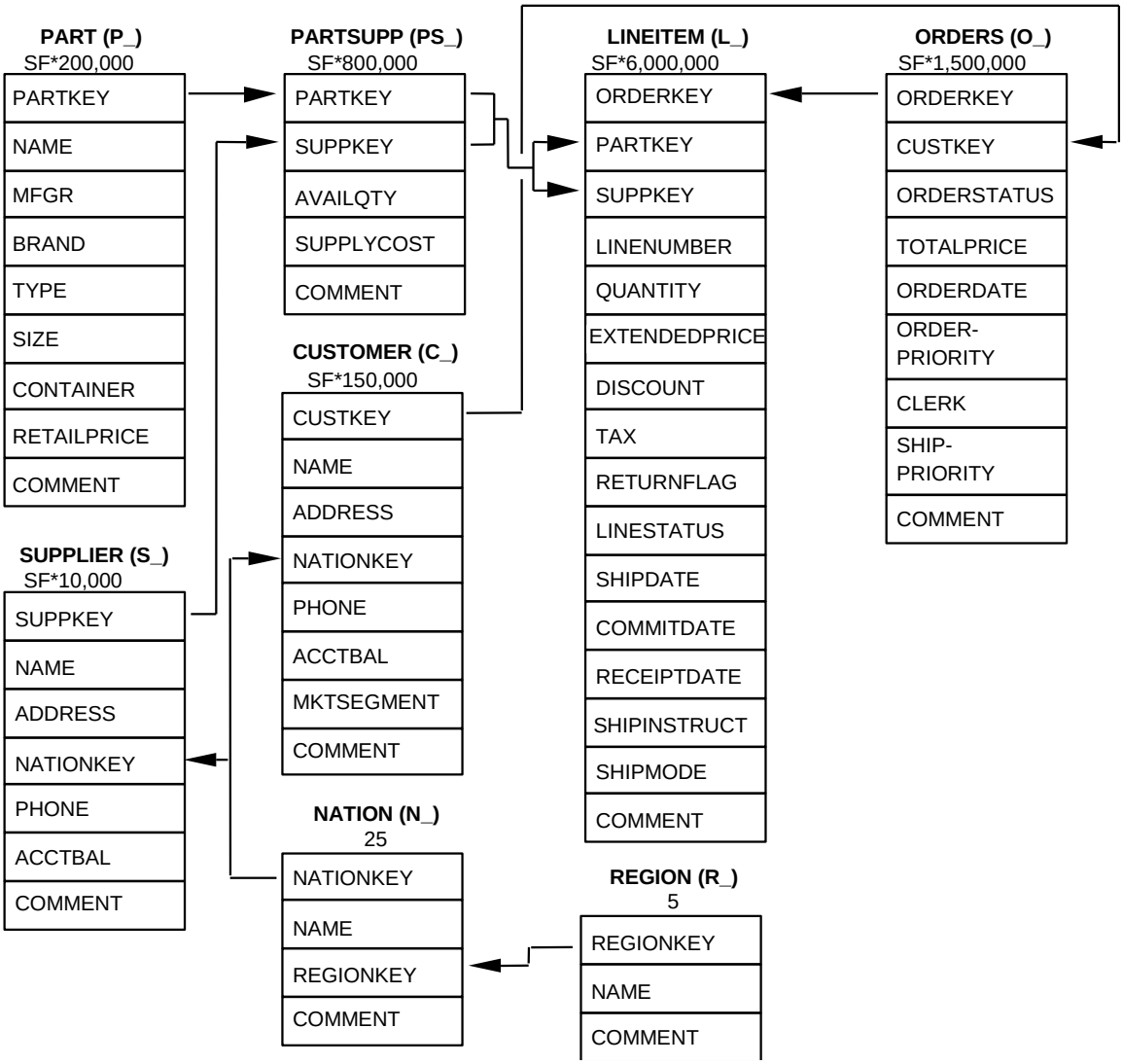

图 1 TPC-H 关系模式图

上述 8 张表代表了电商领域 8 个数据对象，这些数据对象间的关定量联关系如下。

订单明细表 lineitem 的 orderkey 与 orders 表的 o_orderkey 是一一对应关系，订单中的每种商品都出现在 lineitem 表中，每笔订单(orderkey)有 1-7 (linenumber) 种商品，两张表数据量是 1: (1-7)。

orders 表的 o_custkey 信息都在 customer 表中，但不是一一对应关系，即订单上的所有消费者信息都在 customer 表中，但不是所有消费者都购买了商品，大约 2/3 的消费者有订单。

part 和 partsupp 中的 partkey 是一一对应关系，每个零件有 4 个供应商，两张表数据量是 1:4。

supplier 和 partsupp 中的 suppkey 是一一对应关系，每个供应商供应 80 种零件，两张表数据量是 1:80。

lineitem 的 partkey 和 suppkey 均出现在 partsupp 中。

每个 region 有 5 个 nation，每个 nation 中，supplier 和 customer 的比例大约为 1:15。

2.2 TPC-H Benchmark 数据集生成

1. 基于脚本的数据生成

数据由脚本程序生成，通过 C 语言开发，可以通过 visual studio 生成或者配置好 gcc 环境后使用 make 命令生成，可以生成两个程序，dbgen 和 qgen，dbgen 用来生成数据库的数据，qgen 用来生成查询语句。dbgen 运行时需要将源代码目录中的 dists.dss 放到同一目录下。dbgen 通过-s 参数指定数据量为 1G 的倍率，-s 1 表示生成 1G 数据，-f 表示覆盖掉之前生成的数据文件。

以 dbgen -s 0.2 生成 200M 数据为例，生成的数据规模如下所示。

关系模式	数据文件名称	数据文件中数据行数
订单表 ORDERS	orders.txt	300000
供应商表 SUPPLIER	supplier.txt	2000
区域表 REGION	region.txt	5
国家表 NATION	nation.txt	25
零部件表 PART	part.txt	40000
零部件供应表 PARTSUPP	partsupp.txt	160000
客户表 CUSTOMER	customer.txt	30000
订单明细表 LINEITEM	lineitem.txt	1199969

2. 数据生成脚本示例

用于生成 TPC-H 的脚本程序可通过下述链接，从官网下载：
https://www.tpc.org/tpc_documents_current_versions/download_programs/tools-download-request5.asp?bm_type=TPC-H&bm_vers=3.0.1&mode=CURRENT-ONLY

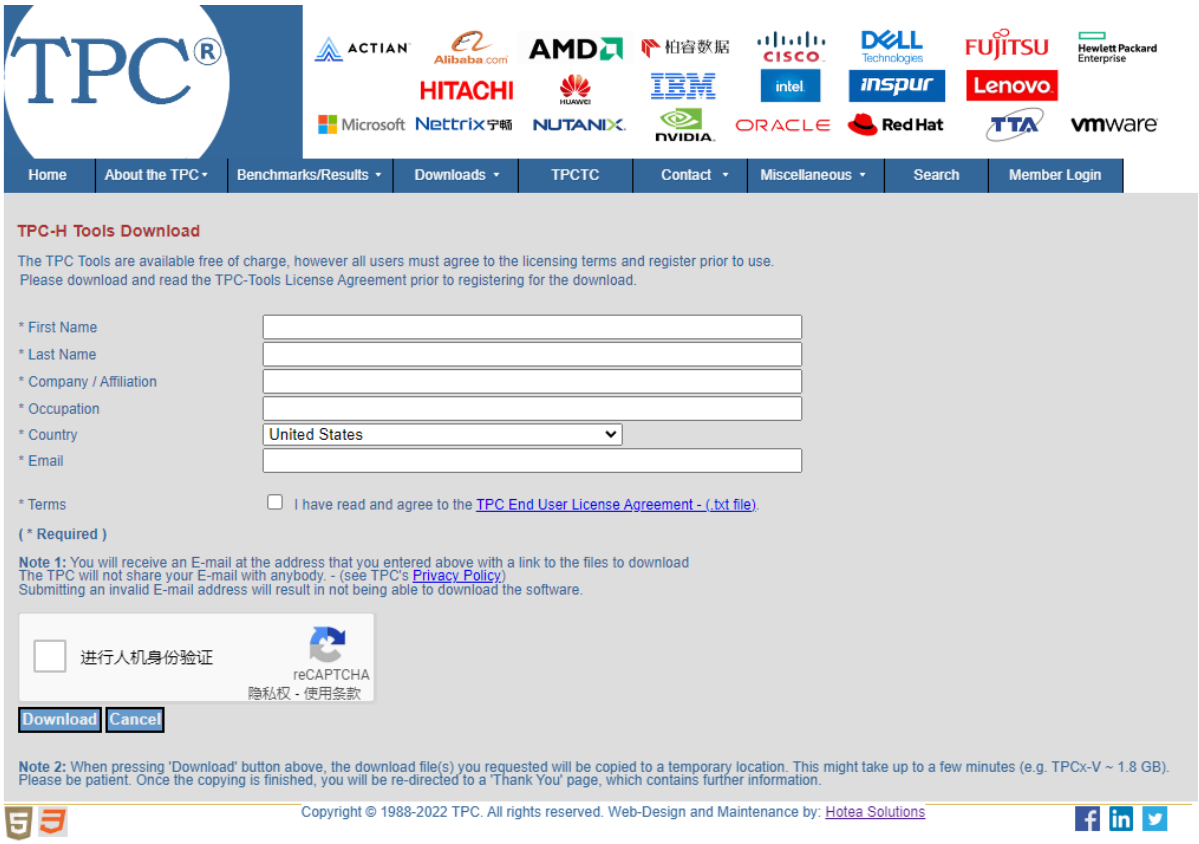

图 2-1 TPC-H 测试基准程序下载页面

部分 TPC-H 测试基准生成程序如下所示：

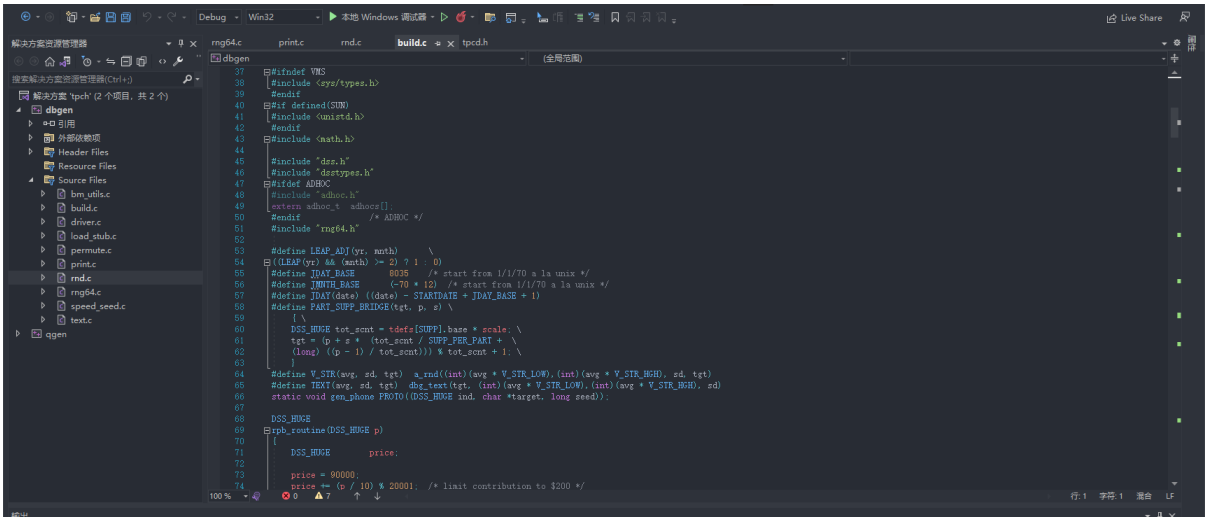

图 2-2 TPC-H 测试基准生成程序（部分）示例

3. 数据库应用系统生命周期与设计过程

本课程设计所开发的 TPC Benchmark 电商数据管理系统属于典型面向事务处理 OLTP 和联机分析处理 OLAP 的数据库应用系统。

面向某一具体应用领域的数据库应用系统（Data Base Application System, DBAS）包括:

- 以数据库文件形式存在的数据库 DB
- 关系数据库管理系统 DBMS， 其核心是数据库引擎. 如 OceanBase, openGauss、PostgreSQL、MySQL、Oracle 等。DBMS 提供了查询处理与优化、存储管理、事务管理等关系数据库系统的核心数据管理功能
- 数据库应用程序. 例如, 数据库系统对外接口程序、位于客户端、对服务器端的数据库 DB 进行各种数据查询与更新的数据库应用程序等。
- 数据库应用系统用户。可以是数据库系统的操作人员或数据库系统管理员 DBA, 也可以是其它应用程序。

数据库应用系统 DBAS 对外提供了**数据存储管理**和**数据访问与处理**功能。因此，DBAS 的设计与实现包括以下 2 个方面：

（1） 数据库 DB 所存储的面向具体应用领域**应用数据的存储设计与实现**。

（2） 对数据库数据进行访问和处理的各种**数据库事务和应用程序**的设计与实现。

注： 对比 程序 = 数据结构 + 算法

数据模型 = 数据定义 + 数据操作

数据库应用系统 DBAS 的设计过程和生命周期如图 3 所示，具体参见“附录 3-数据库应用系统生命周期模型”。DBAS 的设计需要经过概念设计、逻辑设计、物理设计等步骤，有关物理设计的一些内容参见“附录 4-存储技术与物理设计”。

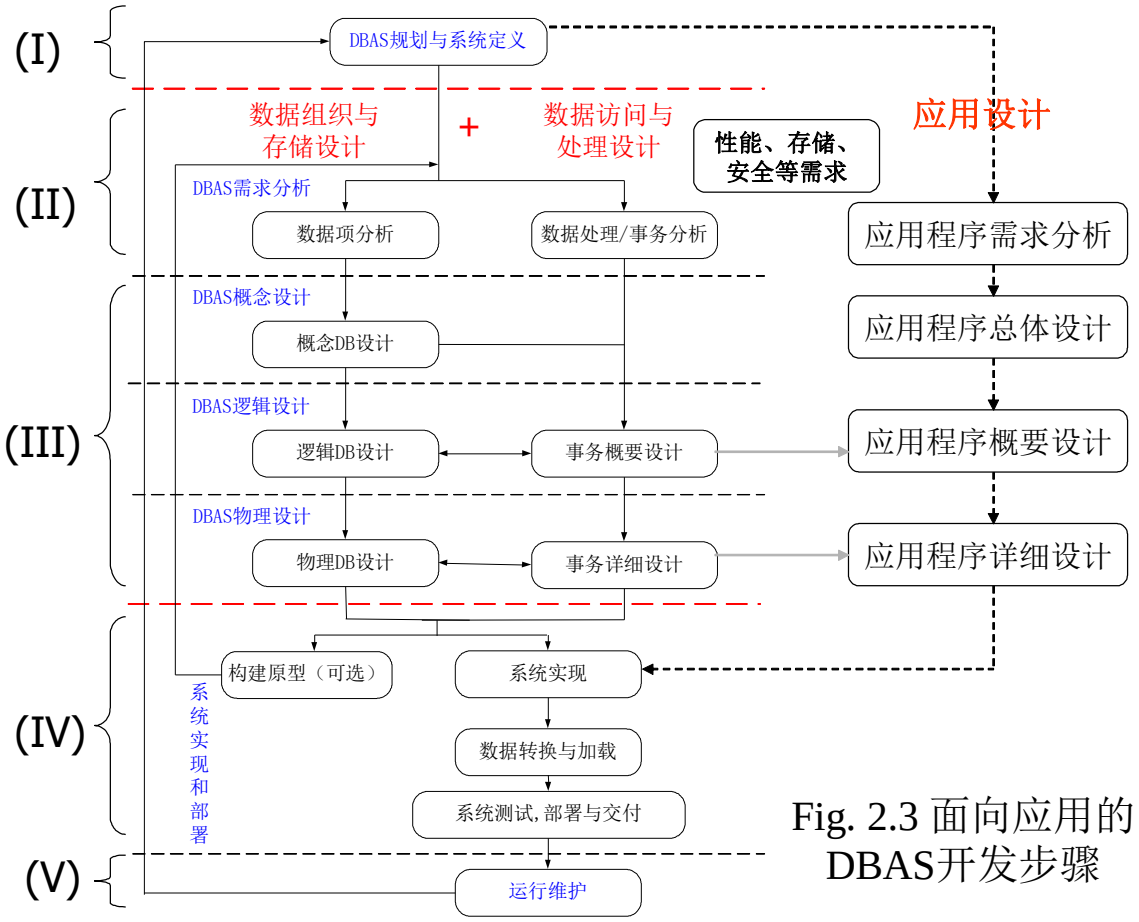

图 3 数据库应用系统 DBAS 的设计过程和生命周期

4. 系统架构

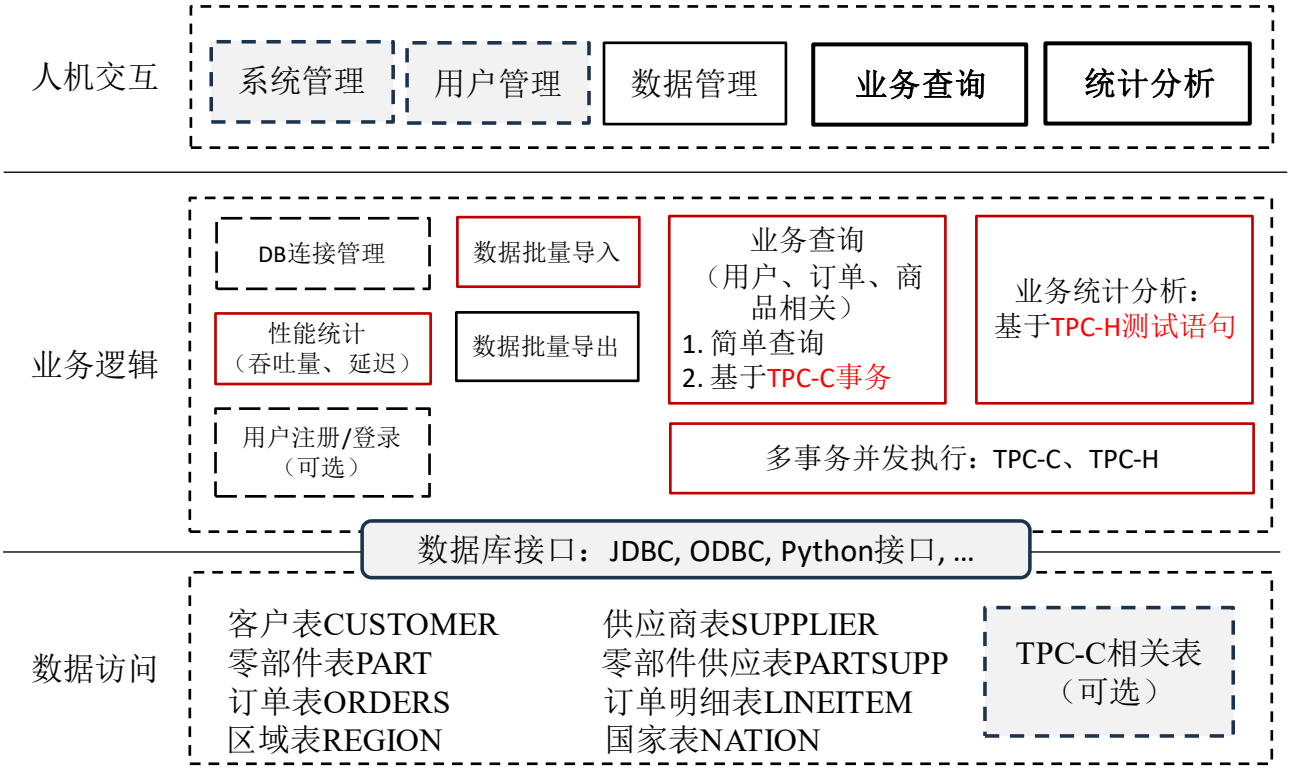

图 4 TPC-H Benchmark 电商数据管理系统层次结构

参照图4，系统采用三层B/S架构。其中数据访问层采用GaussDB数据库系统（如OceanBase, openGauss, GaussDB(MySQL)，或者 PostgreSQL、MySQL，存储存储在 8 张关系表中的电商业务数据等。

说明：根据业务查询、统计分析功能需要，可以增加TPC-C测试基准相关数据库表，如仓库表WAREHOUSE、库存表 STOCK，具体见“附录 4. TPC-C 数据库”。

业务逻辑层实现系统管理、用户管理、数据管理（数据导入/导出）、业务查询、统计分析等功能。用户通过客户端浏览器或其它形式的简单交互界面，访问服务器端的数据库系统，使用系统提供的各项功能，包括典型的 TPC-H 统计分析、TPC-C 数据查询处理。

开发语言可采用 Java、Python、C、C++、C#等，数据库访问接口采用 JDBC、ODBC、Python 接口等。

开发客户端的人机交互和服务器端的数据访问和业务处理功能可以采用两种方式：

1. 编程实现简单的人机交互界面，以及通过数据库接口访问服务器端数据库，并实现 TPC-H、TPC-C 典型业务处理功能。

2. 【选做】为提高系统开发效率，可以（但不强求）采用一些简单易学的 Web 开发框架和工具，如：

Springboot + VUE、Springboot + freemarker 模板，并借鉴 CSDN、知乎等技术网站/论坛上提供的开发案例和系统源码，示例：

- springboot+vue.js 搭建图书管理系统开源项目，
  https://blog.csdn.net/weixin_45132238/article/details/112390505
- 现在开发网站 web 应用一定要前后端分离吗-知乎- Zhihu
  https://www.zhihu.com/question/442535354
- 基于 Vue+Nodejs+MongoDB 小区社区综合治理管理系统毕业生源码案例设计【知乎】
  基于 Vue+Element+nodejs+Express+mongoDB 社区综合治理管理系统
  https://blog.csdn.net/saber04/article/details/107086249
- Github 上 10 个值得学习的 Springboot 开源项目 - 知乎 (zhihu.com)

https://zhuanlan.zhihu.com/p/83908719?utm_campaign=shareopn&utm_medium=social&utm_oi=679111968938397696&utm_psn=1612381112847839232&utm_source=wechat_session

- 基于 SpringBoot 实现操作 GaussDB(DWS)的项目实战

说明：

也可以或两层 C/S 架构和对应的开发环境/工具，如 Qt，不用框架。

5. 需求分析

所开发的 TPC-H Benchmark 电商数据管理系统，实现如下功能。

5.1系统界面

【简单实现即可】

（1）登录界面

登录界面要求有：（1）用户登录，（2）用户注册/撤销等功能，用户分为系统管理员和普通用户两类。

用户通过口令、密码登录系统，注意区分登录系统的口令和连接数据库的口令，普通用户不允许知道连接数据库的口令。

（2）界面布置

要求界面布置合理，简洁美观。

注意设计界面时合理布置一级、二级菜单。一级菜单对应系统管理、用户管理、数据管理、业务查询、业务分析五个主要功能。由一级菜单下的五个系统功能，分别点击进入二级菜单，二级菜单项对应上述五个功能下的具体细分业务功能，如“数据管理”下的“数据导入”。

选择数据库表时要求使用下拉菜单，不能将所有表名平铺在界面上。

5.2 用户管理

【简单实现即可】

用户分为系统管理员、普通用户两大类，两类用户通过账号名、密码登录系统。

管理员可以查看用户注册信息，添加删除普通用户；管理员可以查看数据库连接、后台数据库服务器及数据库的配置信息，可以设置、修改数据库连接时长、数据库缓冲区大小等参数。

普通用户可以自己注册，但需要管理员审批通过。或者：普通用户自己不注册，只能使用管理员分配的帐号，登录系统，完成各项业务查询和业务分析功能。

5.3 系统管理

【简单实现即可】

由系统管理员查看数据库连接、后台数据库服务器及数据库的配置信息，如数据库表所在的物理分区，修改数据库连接时长、数据库缓冲区大小等参数。

当后台服务器端执行 TPC-H、TPC-C 等统计和复杂查询分析功能时，能够统计系统吞吐量、执行延迟等性能指标。

5.4 数据管理

5.4.1 数据生成

利用 TPC-H 数据生成脚本程序，根据需要设定合适的数据规模参数，生成测试数据，存放在 txt 或 csv 文档中。

以 dbgen -s 0.2 生成 200M 数据为例，生成的数据规模如下所示：

关系模式	数据文件名称	数据文件中数据行数
订单表 ORDERS	orders.txt	300,000
供应商表 SUPPLIER	supplier.txt	2,000
区域表 REGION	region.txt	5
国家表 NATION	nation.txt	25
零部件表 PART	part.txt	40,000
零部件供应表 PARTSUPP	partsupp.txt	160,000
客户表 CUSTOMER	customer.txt	30,000
订单明细表 LINEITEM	lineitem.txt	1,199,969

要求：

为支持后续评估分析系统性能，生成数据总量应保证不低于 **600M**。

5.4.2 数据导入

通过数据导入程序，批量导入生成的部分关系表的测试数据，并对导入数据中的部分字段进行数据清洗。

要求：

1. 从生成的测试数据文件中导入至少 2 张关系表的数据，例如从以下关系表中选取：
   订单表 ORDERS，
   零部件供应表 PARTSUPP，
   订单明细表 LINEITEM

注：也可以选取其它规模较大的表作为数据导入对象

2. 导入的每张表中至少选取两个字段，进行数据清洗。
   清洗类型包括：
   - 字段数据类型检查，例如，订单表 ORDERS 的属性“订单总价”应为数值型；
   - 字段数据范围检查，例如，零部件供应表 PARTSUPP 的属性“零件供应数量”取值范围为>=0；
   - 字段空值检查，例如，订单明细表 LINEITEM 的属性“数量”不为空。

3. 导入和清洗过程应能够正确识别不符合要求的数据，并在导入日志中记录不符合要求的数据行的行号。导入过程不会因存在不符合要求的数据而终止，符合要求的规范数据必修全部导入数据库。

4. 其它关系表的数据可借助 DBSM 平台提供的导入机制，手动导入。

说明：
如果 TPC-H 测试数据生成程序生成的数据都符合规范，为验证数据清洗效果，可以人为手动修改表中的某些字段值，构造出非规范数据。

注意：
存放导入数据的数据文件在系统中的存放路径必须是可选的，不允许在导入程序中采用固定的导入文件路径。

5.4.3 数据导出

在用户界面上用下拉列表显示数据库中的各个关系表，指定某些表，如 ORDERS，将表中数据以 Excel 或 txt 格式的文件输出，要求输出文件中应当有表中各个属性名称，输出文件名为该关系表的名字。

至少从数据库中选择 3 张表，实现数据导出。

注意：
导出的数据文件在系统中的存放路径必须是可选的，不允许在导出程序中采用固定的导出文件存放路径。

5.5 业务查询

设计 SQL 查询语句（至少涉及两张表），或选择附录 2 给出的典型 TPC-H 测试语句，如“定价汇总报表查询”，设计客户端输入界面、结果输出界面，支持用户在客户端输入查询需求后，系统能够正确返回

查询结果，并以图表等方式呈现查询结果。

要求完成以下两类简单查询

1. 客户信息查询

要求：

1） 用户在查询界面的输入框中直接输入某客户的 name 或所在国家名称，系统以列表方式输出该小区全部信息；

2） 程序在查询界面的下拉列表中列出客户表中全部客户姓名或所在国家名称，用户通过下拉列表选择特定客户名称，以列表方式输出该客户全部信息；

2. 从附录 2 给出的典型 TPC-H 测试语句中，选择与订单 ORDER 或零部件 PART 相关的查询，对其进行适当修改完善，自行设计查询条件和查询结果呈现方式，要求查询结果能以图表的方式直观呈现。

注意：

查询结果要求可以导出到文件中，保存到可选文件存储路径下。

5.6 TPC-H 统计分析

从附录 2 给出的典型 TPC-H 测试语句中，选择 2 条具有比较复杂的统计分析功能、并且执行时间较长的 SQL 查询语句，如带有分组聚集运算功能的 SQL 查询，对其进行适当修改完善，在系统内实现。

用户可以在客户端通过数据库访问接口将这些 SQL 查询请求传送至服务器端数据库系统，启动查询，对 TPC-H 数据库进行统计分析。可以自行设计查询条件和查询结果呈现方式，如对特定区域/国家、特定时间段的数据进行统计分析。

要求：

1.记录每条 SQL 查询从提交查询请求到返回查询结果的时间。

2.使用者通过 DBMS 平台，提取和观察 SQL 查询的查询执行计划，分析其执行过程，以及对索引的使用情况。

5.7 TPC-C 事务查询

从附录 5 给出的典型 TPC-C 测试事务中，选择 2 个具有比较复杂的查询处理功能、并且执行时间较长的典型事务，如事务中带有 update、insert 等操作，对其进行适当修改完善（如改用 Python 重新编写），在

系统内实现这些事务。同时，根据这些事务访问的具体数据库表，在已有的 TPC-H 数据库 8 张表基础上，增加事务所访问的 TPC-C 数据库表。

用户可以在客户端启动、控制这些事务的执行，对数据库表进行增删改查。可以自行设计查询条件和查询结果呈现方式，如对特定区域/国家、特定时间段的数据进行统计分析。

要求：

1. 记录每个事务从提交查询请求到返回查询结果的时间，记录事务内部各条 SQL 查询语句的执行时间。
2. 使用者通过 DBMS 平台，提取和观察 SQL 查询的查询执行计划，分析其执行过程，以及对索引的使用情况。

5.8 TPC-H 并发测试

利用 TPC-H 测试程序，生成 TPC-H 测试数据和 SQL 测试模板。

采用 Python、Java、C 语言等，编写 TPC-H 测试程序。执行测试程序，从一个或多个客户端，通过数据库接口向数据库提交 SQL 查询，启动 TPC-H 测试，观察 SQL 事务的并发执行过程，统计系统吞吐量、延迟等系统性能指标。

5.9 TPC-C 并发事务测试

采用 TPC-C 测试程序，生成 TPC-C 测试数据和测试事务。

从多个客户端同时启动 TPC-C 测试，观察事务的并发执行过程，统计系统吞吐量、延迟等系统性能指标。

6. 应用程序设计与查询结果分析

6.1 数据批量导入

导入数据时，要求数据库中已有对应的表结构。

如前所述，向数据库表导入数据时，必须根据导入的数据在数据库中是否已经存在，采取 insert 或 update 动作。如果完全由导入程序采用 insert 语句逐条判断并导入，导入过程耗时费力，将非常慢。为此，采用 DBMS 提供的批量导入技术，向数据库中一次性地插入多条数据，实现对导入数据的批量清洗和快速导入。

不同的数据库平台、开发框架、编程语言提供了多种数据库批量导入方法，如 bulkinsert、bulkcopy 等批量导入语句。在本课程设计中，需要开发批量导入程序和定义在关系表上的导入触发器，两者者协同配合，完成数据清洗和数据批量导入工作。

以从数据文件客户表向订单表 ORDERS 中批量导入小区配置数据为例，其原理如下图所示。

第 1 步. 分批读取输入文件数据

导入程序分批读取导入数据文件，每次从数据文件 orders.txt 文件中批量读取多行（如 K=50 行）数据，存放在内存数据结构 InputBuffer 中，例如在 C#语言中的 datatable 数据结构。

在 C#语言中，采用 datatable 结构的 InputBuffer 具有数据库表订单表 ORDERS 完全一致的逻辑结构。

第 2 步. 数据清洗

在导入程序将 K 行数据逐条读入 InputBuffer 的过程中，对导入数据进行清洗，判断每条数据的各个属性值是否符合数据类型要求、取值是否在合法范围，剔除掉来自于数据文件的不符合要求的输入数据。并在导入日志（txt 文件）中记录被剔除的数据在输入文件的位置、编号。

第 3 步. 批量插入数据

对经过清洗后、存放在 InputBuffer 中的多条数据，采用数据库提供的批量插入语句，例如 BulkInsert、bulkcopy，配合定义在关系表 ORDERS 上的导入触发器，将这些数据全部插入数据库。

DBMS 收到由批量插入语句传来的需要导入 ORDERS 表的数据后，自动启动定义在订单表 ORDERS 上的导入触发器，由导入触发器对这批数据进行 insert 或 update 操作。具体如下（以 SQL Server 为例）：

（1）当用 bulkinsert 向 ORDERS 中插入数据时，在系统内部会自动生成一个临时关系表 tbInserted，该表存放被插入的数据，并且定义在该表上的导入触发器是可以直接访问 tbInserted 的。

（2）导入触发器依次扫描 tbInserted 中的每个元组，如果某个元组中的主键值<订单 key，客户 key>在数据库表 ORDERS 中已经出现，则将该元组放到临时关系表#UpdateTemp。按照此种方式，找出导入数据中需要 Update 操作的全部元组，并暂时存放在#UpdateTemp。

（3）tbInserted 中存放的是对应于 Insert 或 Update 操作的 2 类元组，因此将 tbInserted 与#UpdateTemp 做集

合差操作，得到的就是以前没有在 tbCell 中出现过、需要 Insert 的元组，这些元组缓存在#InsertTemp 中。

（4）触发器对临时关系表#InsertTemp、#UpdateTemp 中的元组分别执行 Insert、Update（也可以先 delete、后 insert）操作，将数据导入 tbCell 中。

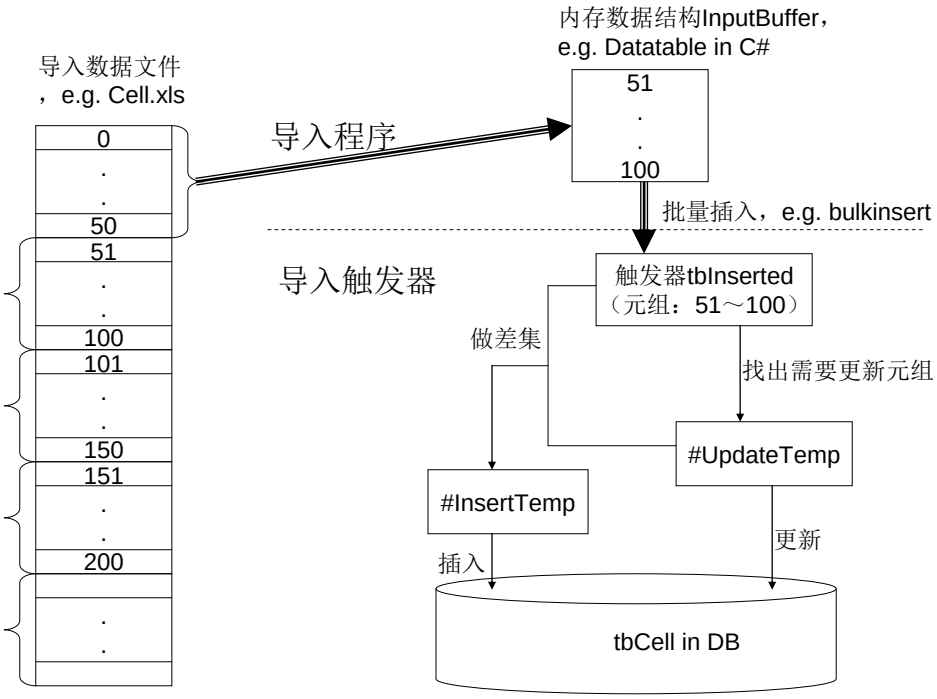

图 6 数据批量导入过程示意

注意：

1. 不允许将数据导入文件，如 orders.txt，的全部内容一次性地都读到内存中

原因：实际应用中，有可能导入数据文件非常大，如 1G 大小的数据文件。要求先对文件内容进行清洗，然后再导入数据库，为了清洗数据，必须将文件中数据先读入内存。数据文件过大，一次性全部读入内存会导致死机。

正确做法：

根据内存大小、需要导入的数据文件中一行数据的大小，计算允许一次性导入内存的数据量，如 400M。如果数据文件没有超过 400M，则一次性导入文件；如果文件大于 400M，分批导入。

简单做法：分批导入，每次导入固定行的数据，如 50 行。

2. 数据文件中多行数据读入内存、进行清洗后，不要再写回到磁盘文件中，应直接批量写入数据库，避免对 1 个大的数据文件（如 10G）多次读写磁盘。

C#/Java 中用 bulk insert、bulk copy 等操作可以将内存中的数据批量地插入数据；如果采用 Python 语言编写数据导入程序，可以采用 ExcuteMany 方法实现数据批量插入数据库。

6.2 函数/存储过程/触发器

数据导入功能需要使用触发器来实现。

针对前述业务查询、业务分析需求，设计函数、存储过程，或触发器，实现查询、分析计算中的某些步骤。

这些函数、存储过程经编译后，存储在数据库服务器上，应用程序的各功能模块直接调用这些存储过程，从而实现查询执行计划重用，提高系统开发和程序运行效率。

6.3 数据库物理文件/文件组设计

数据库在物理层面由多种数据库文件组成，如存放应用数据的数据文件、存放日志的日志文件。

数据库物理设计时需要结合数据库平台提供的机制，创建数据库，将数据库文件根据实际需要安排在不同的磁盘分区中。

例如，SQL Server 数据库的数据库文件分为主、辅文件、日志文件、文件组，数据库物理设计时，可以利用 create database 语句部署在不同物理分区。

查阅资料，根据课程设计采用的 OceanBase, openGauss、PostgreSQL 等数据库的数据文件类型、日志文件结构，以及数据库文件部署方式，将数据库的数据文件安放在不同的分区中，不要求只放在默认分区。

6.4 索引设计与查询执行计划

为提高系统的数据查询效率，需要根据用户访问情况，在 ORDERS 等关系表的相关属性上建立合适的索引，并利用 DBMS 平台提供的机制，观察 SQL 语句查询执行计划，分析索引对 select、update、insert、delete 等 DML 操作的影响，以及对 alter 等 DDL 操作的影响，判断分析索引有无起到预期作用。设计结果记录在数据库表设计文档中。

索引建立的原则参见课程讲义、“实验 8 数据查询分析与优化”。

6.5 TPC-H 并发测试

面向联机分析处理 OLAP 的 TPC-H 测试基准包括：（1）8 张表组成的测试数据库，（2）22 条 SQL 语句组成的测试模板，涉及采用分组和聚集运算对数据库表进行统计分析，如。（3）测试数据及测试脚本生成程序。

TPC-H 测试程序下载地址:
https://www.tpc.org/TPC_Documents_Current_Versions/download_programs/tools-download-request5.asp?bm_type=TPC-H&bm_vers=3.0.1&mode=CURRENT-ONLY

利用 TPC-HTPC-H 测试程序，进行测试的步骤如下：
Step1. 利用 TPC-HTPC-H 测试程序，生成测试数据；
Step2. 利用 TPC-HTPC-H 测试程序，生成 22 条 TPC-H SQL 测试语句对应的 SQL 测试模板；
Step3. 采用 Python、Java 等，编写测试程序，测试程序中包括多个 SQL 测试模板；将多个 SQL 测试模板通过 1 个或多个终端，提交给数据库系统，并发执行这些 SQL 测试模板。

Step1、Step2 可参见“附录 4 TPC-H 测试数据和 SQL 测试模板生成”。

Step3 可参见“附录 7 TPC-H 测试程序示例”

6.6 TPC-C 并发测试

面向联机事务处理 OLTP 的 TPC-C 测试事务包括 5 个基本测试事务。每个事务由多条 SQL 语句组成，涉及对 TPC-C 数据库的增、删、改、查，采用嵌入式 SQL 实现，编程语言为 C 或 JAVA。

包含详细代码及说明的 pdf 文件可从此处下载（TPC-C）：
https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

文档下载地址：https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf
五种 TPC-C 事务的定义和说明见文档第二章，五种事务的嵌入式 C 程序见附录。
BenchmarkSQL 是 TPC-C 的 java 实现

五个 TPC-C 基本事务为：

1. **New-Order**：客户输入一笔新的订货交易

事务内容：对于任意一个客户端，从固定的仓库随机选取 5-15 件商品，创建新订单。

2．Payment：更新客户账户余额以反应其支付状况

事务内容：对于任意一个客户端，从固定的仓库随机选取一个辖区及其内用户，采用随机的金额支付一笔订单，并作相应历史纪录。

3．Delivery：发货(批处理交易)

事务内容：对于任意一个客户端，随机选取一个发货包，更新被处理订单的用户余额，并把该订单从新订单中删除。

4．Order-Status：查询客户最近交易的状态

事务内容：对于任意一个客户端，从固定的仓库随机选取一个辖区及其内用户，读取其最后一条订单，显示订单内每件商品的状态。

5．Stock-Level：查询仓库库存状况，以便能够及时补货。

BenchmarkSQL 是采用 Java 实现的 TPC-C 测试基准，包括(1) TPC-C 测试数据库，由 10 张表组成；(2) 5 个由 Java 实现的基本测试事务。

测试过程分为三步：

Step 1. 建立 TPC-C 测试库，配置测试数据参数，生成 10 张表的测试数据，并导入数据库表；

Step 2. 配置客户端事务参数，生成测试事务；

Step 3. 启动终端测试事务，进行测试。

测试数据和测试事务的生成程序的下载地址如下：

https://sourceforge.net/projects/benchmarksql/files/benchmarksql-5.0.zip/download

采用 BenchmarkSQL 进行 TPC-C 测试的具体方法和步骤参见“附录 6 . TPC-C 并发事务测试”。

6.7 隔离级别与事务分析【选做】

参考教科书 **17.8 Transaction Isolation Levels/18.9 Weak Levels of Consistency** 相关内容，分析本课程设计采用的 DBMS 平台（MySQL, PostgreSQL, OceanBase, openGauss, SQL Server）所支持的事务隔离级别，参照课程实验中的事务管理相关实验，设计多用户并发访问数据库，观察在不同隔离级别下，查询带来的不一致性和查询结果的差异。

隔离级别	描述	不一致现象
Read uncommitted 未提交读、脏读	允许读取未提交的记录	读脏数据 不可重复读 幻象
Read committed	只允许读取已提交的记录，但不要求可重	不可重复读

提交读（SQL Server 默认的隔离级别）	复读（即不要求对同一数据项，每次读取的值是一样的）	幻象
Repeatable Read 可重复读	只允许读取已提交的记录，并要求在 1 个事务对数据的 2 次读取之间，其它事务不能对该数据进行更新（从而保证每次读取的值是一样的）	幻象
Serializable **可串行化**	调度的执行等价于串行调度	----------

7. 课程设计内容及注意事项

在上学期的课程实验基础上，根据前述系统需求，遵循图 4 数据库应用系统 DBAS 的设计过程和生命周期，设计开发 TPC-H Benchmark 电商数据管理系统。

系统设计包括数据存储的设计和数据处理设计两方面，即数据库表设计、软件模块结构设计（如系统模块化分、各模块输入/输出和功能描述）、索引设计等。

分组完成课程设计，每组 3-4 人。如果采用 B/S 架构和 Web 开发框架，且系统实现的功能比较多，设计和开发工作量较大，每组成员 4 人。

注意事项：

1. 选择合理的开发语言和环境，保证各项实验内容均能按照要求完成；
2. 先使用 create database 语句创建数据库，合理设计数据库主文件、辅文件，将数据文件、日志文件、索引文件安排在不同磁盘分区中；

   数据库创建完成后，再建立数据库表。可以采用基本课程实验的建表脚本；
3. 数据库表创建后，在后续的导入/导出、查询、分析过程中，不允许修改数据库中结构，如不允许 drop、create 关系表；但允许动态调整索引；

   注意区分：DB 设计者、DB 使用者的角色，权限不同；
4. 界面设计，详见 3.1，可以有自己个性化的设计，但必须涵盖所有功能，注意页面布局，尽量简洁、美观、直观，建议采用一级菜单-二级菜单项的界面排列方式；不要将全部功能安排在一个页面内；
5. 用户登录界面应区分开用户级口令、密码和数据库连接级口令密码，对用户级口令密码要有用户注册功能；
6. 导入时必须完成批量导入，不要一次性导入全表，而且数据库已存在表结构（为空表或旧数据）；

   导入表时要求通过路径选择任意的表；导出数据库中表和查询结果的图表都要求导出路径可选，不要直接在代码里固定表名和路径名；

   采用触发器，判断导入的数据是否主键与表中现有数据重复，采取 update 或 insert/delete 方式更新数据，建议：采用 update，但需要实验比较，哪个好一些？？

   数据分批时，要根据输入内容和可用内存估计，合理选择每次读入内存清洗的数据量。不能简单地输入读入 1 个货多个文件块。

   导入时：选择要导入的数据库表名/数据库中数据类型(如小区级工参)、要导入的输入文件名和路径；

   数据导出时，可以自定义导出文件名，选择文件存放路径

   防止数据导入时，分批读取导入文件中数据——不能全读到内存中，进行数据清洗（包括类型检

查、范围检查）；对清洗后的数据直接由内存，通过数据库批量导入操作，写入数据库。不允许：将清洗后的数据写入磁盘文件，再有文件导入数据库。

7. 导入时，增加导入进度条；

8. 查询时，要求输入框和下拉列表的方式都实现，2 种方式放在同一个框中；

输入查询条件时，对数值型属性，如 eid，需要进行类型检查，防止输入非数值型数据导致死机；

根据系统设计开发过程和结果，完成课程设计报告中，课程设计报告主要内容如下：

- （报告第 1 页）封面：北京邮电大学课程设计报告

  按照封面规定的格式，填写小区成员信息和人员分工、课程设计内容（简介）。
- 课程设计的目的、内容
- 系统需求分析

  从系统界面、数据管理、业务查询、业务处理等几方面，归纳系统应实现的功能，以及应到达的性能。
- 系统设计

  包括：系统架构/层次结构设计，软件模块设计，

  数据库设计（E-R 图设计、数据库逻辑设计/关系表设计、函数/触发器设计、索引及数据库文件设计、批量导入设计等），

  数据库文件物理设计，

  接口访问接口设计等
- 系统实现

  软硬件平台，开发环境，编程语言，采用的框架
- 系统运行示例

  系统提供的 5 大类业务功能的运行示例截图
- 总结

注意：

- 课程设计报告文档命名规则：**班级_姓名 1_姓名 2 数据库课程设计报告**
- 系统代码作为单独附件提交

8. 实验环境与工具

- DBMS:

  OceanBase, openGauss, PostgreSQL, MySQL, GaussDB(MySQL，SQL Server, SQL Server 等其它 DBMS 平台。

  为与课程实验内容相衔接，建议优先选用 OceanBase, openGauss，PostgreSQL。

- Programming languages:

  Python、C、C++、Java

- DB interface

  ODBC, JDBC，ADO, Python/connector 等

- 开发框架

  MyBatis;

  Springboot + VUE、Springboot + freemarker；

  SSH [JSP+HLML+JS，后端采用传统 JDBC 与 HQL]

注：

可以参照网上资料与案例，根据需要，自行选择多种开发框架和工具。

9. 实验报告与课程验收

1. 开发流程遵循图 4 所示的 DBAS 的设计过程和生命周期模型规定的步骤。每一步中应遵循数据库系统开发的基本原理和方法。
2. 课程设计文档应格式规范，内容完整。包括封面、目录、正文内容、案例图片、附录等
3. 课程设计完成后，各组提交设计文档、程序，并通过程序演示进行课程验收。
4. **重点注意事项：**

   1） 提交文档的名称统一采取如下形式：

   班级_姓名 1_姓名 2 数据库课程设计报告.doc

   2） 文档必修包括封面，封面格式参见：

   课程设计报告封面(分组版)

在封面栏目“课程设计内容”中，填写课程设计内容、组员分工。

3） 不要照抄指导书中内容

4） 将所有设计文档合并在一个 Word 文件中，图片插入文档中合适位置；所有程序代码以附录形式单独放在一个文件中。

5. 实验必须包括用户管理、数据批量导入、存储过程与触发器、索引设计、查询指标图形化显示这 4 方面内容，如有缺失，扣分。
6. 第 8 周左右中期检查
   - 线上或线下验收
   - 各组应完成的实验内容不少于全部内容的 50%，保证课程设计进度
7. 最终课程设计验收在第 15-16 周，提前完成的组可提前预约验收
8. 按时提交文档，并进行验收。验收前，文档发至：yewen@bupt.edu.cn

   验收时间、地点：待定

   如果提前做完，可以提前预约验收。

10. 参考资料

教材：

Abraham Silberschatz, Henry F.Korth, S. Sudarshan, **Database System Concepts** (7th Edition), Higher Education Press and McGraw-Hill Companies, Beijing, May, 2021

TPC 官网及 TPC-H 测试/数据生成程序：

用于生成 TPC-H 的脚本程序可通过下述链接，从官网下载：

https://www.tpc.org/tpc_documents_current_versions/download_programs/tools-download-request5.asp?bm_type=TPC-H&bm_vers=3.0.1&mode=CURRENT-ONLY

参考书目：

蒋崇礼，培养计算机类专业学生解决复杂工程问题解决的能力，清华大学出版社，北京，2018 年 7 月

北京邮电大学课程设计报告（封面）

<table>
<tr><td>课程设计名称</td><td colspan="2">数据库系统原理课程设计</td><td>学院</td><td>计算机学院</td><td>指导教师</td><td></td></tr>
<tr><td>班级</td><td>班内序号</td><td colspan="2">学号</td><td>学生姓名</td><td colspan="2">成绩</td></tr>
<tr><td>2018211xxx</td><td></td><td colspan="2"></td><td>xxx</td><td colspan="2"></td></tr>
<tr><td>2017211xxx</td><td></td><td colspan="2"></td><td>xxx</td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td><td colspan="2"></td></tr>
<tr><td>课程设计内容</td><td colspan="6">简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等
教学目的：

内容方法：

分工：</td></tr>
<tr><td>学生课程设计报告
(附页)</td><td colspan="6"></td></tr>
<tr><td>课程设计成绩评定</td><td colspan="6">遵照实践教学大纲并根据以下四方面综合评定成绩：
1、课程设计目的任务明确，选题符合教学要求，份量及难易程度
2、团队分工是否恰当与合理
3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果
4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范
评语：

成绩：

指导教师签名：
年 月 日</td></tr>
</table>

注：评语要体现每个学生的工作情况，可以加页

1. 附录 1. TPC-H Benchmark 数据库表

1.1 订单表 ORDERS

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
O_ORDERKEY	订单 key	INTEGER	NOT NULL 主键
O_CUSTKEY	客户 key	INTEGER	NOT NULL 外键
O_ORDERSTATUS	订单状态	CHAR(1)	NOT NULL
O_TOTALPRICE	订单总价	DECIMAL(15,2)	NOT NULL
O_ORDERDATE	下单日期	DATE	NOT NULL
O_ORDERPRIORITY	订单优先级	CHAR(15)	NOT NULL
O_CLERK	收银员	CHAR(15)	NOT NULL
O_SHIPPRIORITY	发货优先级	INTEGER	NOT NULL
O_COMMENT	备注	VARCHAR(79)	NOT NULL

1.2 供应商表 SUPPLIER

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
S_SUPPKEY	供应商 key	INTEGER	NOT NULL 主键
S_NAME	供应商姓名	CHAR(25)	NOT NULL
S_ADDRESS	供应商地址	VARCHAR(40)	NOT NULL
S_NATIONKEY	供应商国家 key	INTEGER	NOT NULL 外键
S_PHONE	供应商手机号	CHAR(15)	NOT NULL
S_ACCTBAL	供应商账户余额	DECIMAL(15, 2)	NOT NULL
S_COMMENT	备注	VARCHAR(101)	NOT NULL

1.3 地区表 REGION

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
R_REGIONKEY	地区 key	INTEGER	NOT NULL 主键
R_NAME	地区名	CHAR(25)	NOT NULL
R_COMMENT	备注	VARCHAR(152)	

1.4 国家表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
N_NATIONKEY	国家 key	INTEGER	NOT NULL 主键
N_NAME	国家名	CHAR(25)	NOT NULL
N_REGIONKEY	国家所在地区 key	INTEGER	NOT NULL 外键
N_COMMENT	备注	VARCHAR(152)	

1.5 零部件表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
P_PARTKEY	零件 key	INTEGER	NOT NULL 主键
P_NAME	零件名称	VARCHAR(55)	NOT NULL
P_MFGR	零件厂商	CHAR(25)	NOT NULL
P_BRAND	零件品牌	CHAR(10)	NOT NULL
P_TYPE	零件类型	VARCHAR(25)	NOT NULL
P_SIZE	零件大小	INTEGER	NOT NULL
P_CONTAINER	零件包装	CHAR(10)	NOT NULL
P_RETAILPRICE	零件零售价	DECIMAL(15,2)	NOT NULL
P_COMMENT	备注	VARCHAR(23)	NOT NULL

1.6 零部件供应表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
PS_PARTKEY	零件 key	INTEGER	NOT NULL 主键 外键
PS_SUPPKEY	零件供应行 key	INTEGER	NOT NULL 主键 外键
PS_AVAILQTY	零件供应数量	INTEGER	NOT NULL
PS_SUPPLYCOST	零件供应成本	DECIMAL(15,2)	NOT NULL
PS_COMMENT	备注	VARCHAR(199)	NOT NULL

1.7 客户表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
C_CUSTKEY	客户 key	INTEGER	NOT NULL 主键
C_NAME	客户姓名	VARCHAR(25)	NOT NULL 主键
C_ADDRESS	客户地址	VARCHAR(40)	NOT NULL
C_NATIONKEY	客户国家 key	INTEGER	NOT NULL 外键
C_PHONE	客户电话	CHAR(15)	NOT NULL
C_ACCTBAL	客户账户余额	DECIMAL(15, 2)	NOT NULL
C_MKTSEGMENT	客户市场领域	CHAR(10)	NOT NULL
C_COMMENT	备注	VARCHAR(117)	NOT NULL

1.8 订单明细表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
L_ORDERKEY	订单 key	INTEGER	NOT NULL 主键 外键
L_PARTKEY	零件 key	INTEGER	NOT NULL 外键
L_SUPPKEY	供应商 key	INTEGER	NOT NULL 外键
L_LINENUMBER	流水号	INTEGER	NOT NULL 主键
L_QUANTITY	数量	DECIMAL(15,2)	NOT NULL
L_EXTENDEDPRICE	价格	DECIMAL(15,2)	NOT NULL
L_DISCOUNT	折扣	DECIMAL(15,2)	NOT NULL
L_TAX	税	DECIMAL(15,2)	NOT NULL
L_RETURNFLAG	退货标志	CHAR(1)	NOT NULL

L_LINESTATUS	明细状态	CHAR(1)	NOT NULL
L_SHIPDATE	发货日期	DATE	NOT NULL
L_COMMITDATE	预计到达日期	DATE	NOT NULL
L_RECEIPTDATE	实际到达日期	DATE	NOT NULL
L_SHIPINSTRUCT	运单处理策略	CHAR(25)	NOT NULL
L_SHIPMODE	运送方式	CHAR(10	NOT NULL
L_COMMENT	备注	VARCHAR(44)	NOT NULL

2. 附录 2. 典型 TPC-H 测试语句

2.1定价汇总报表查询

查询已经开票，发货和退货三个类别订单的业务总量，根据给定日期，返回退货标志，订单状态，总价格，总折扣价格，总折扣价格加税，平均数量，平均价格，平均折扣，订单数量，结果按照退货标志和订单状态升序。

```
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '2021-12-01' - interval '90' day (3)
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

2.2最低成本供应商查询

这个查询用来确定每个地区该选择哪个供货商，在给定区域中，针对给定类型和大小的零件，找到能够以最低价格供应的供应商。 如果该地区的多个供应商以相同（最低）价格提供所需的零件类型和尺寸，则列出来具有 100 个最高帐户余额的供应商的零件。 返回供应商的账户余额、名称和国家，零件 key 和制造商，供应商的地址、电话号码和备注信息，结果按照账户余额降序，国家名，供应商名，地区 key 升序。

```
select
        s_acctbal,
        s_name,
        n_name,
        p_partkey,
        p_mfgr,
        s_address,
        s_phone,
        s_comment
from
        part,
        supplier,
        partsupp,
        nation,
        region
where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and p_size = 15
        and p_type like '%BRASS'
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'EUROPE'
        and ps_supplycost = (
                select
                        min(ps_supplycost)
                from
                        partsupp,
                        supplier,
                        nation,
                        region
                where
                        p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = 'EUROPE'
        )
order by
        s_acctbal desc,
```

```
    n_name,
    s_name,
    p_partkey;
LIMIT 100
```

2.3运输优先级查询

根据给定的日期，查找出 10 个未发货的价格最高的订单，返回订单 key，总潜在收入，下单日期，和运输优先级，按照潜在收入降序，下单日期升序，如果超过 10 个，只返回 10 个。

```
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'BUILDING'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '2018-03-15'
    and l_shipdate > date '2018-03-15'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
LIMIT 10
```

2.4订单优先级查询

这个查询用来确定订单优先级系统的工作是否良好，并给出客户满意度。根据下单日期，返回至少有一个订单明细在预计送达时间之后收到的订单优先级和订单数量，并根据订单优先级升序序。

```
select
```

```
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '2016-07-01'
    and o_orderdate < date '2016-07-01' + interval '3' month
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
```

2.5本地供应商收入量查询

列出给定地区每个国家的订单交易产生的收入量，返回国家名字和收入量，并根据收入量降序。

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
```

```
    and n_regionkey = r_regionkey
    and r_name = 'ASIA'
    and o_orderdate >= date '2017-01-01'
    and o_orderdate < date '2017-01-01' + interval '1' year
group by
    n_name
order by
    revenue desc;
```

2.6预测收入变化查询

根据给定发货时间、折扣，数量，返回在发货时间一年内，折扣增减 0.01，数量小于给定数量的订单明细的总折扣。

```
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '2017-01-01'
    and l_shipdate < date '2017-01-01' + interval '1' year
    and l_discount between .06 - 0.01 and .06 + 0.01
    and l_quantity < 24;
```

2.7批量出货查询

给定时间和供应商所在的国家和客户所在的国家，查找出在该段时间内两个国家出货的总收入，返回供应国家名，客户国家名，年份，和出货的收入，结果按照供应国家名，客户国家名，年份升序。

```
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
```

```
            n2.n_name as cust_nation,
            extract(year from l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
                or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
            )
            and l_shipdate between date '2018-01-01' and date '2019-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
```

2.8全国市场份额查询

给定国家，地区，周期和零件 key，查找周期内的每年该国在该地区收入所占的份额，结果返回年份和周周期比，按照年份升序。

```
select
    o_year,
    sum(case
        when nation = 'BRAZIL' then volume
        else 0
```

```
    end) / sum(volume) as mkt_share
from
    (
        select
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'AMERICA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between date '2018-01-01' and date '2019-12-31'
            and p_type = 'ECONOMY ANODIZED STEEL'
    ) as all_nations
group by
    o_year
order by
    o_year;
```

2.9产品类型利润度量查询

给出一个零件 key，查询该零件在不同国家，不同年份的利润，返回国家，年份和利润，按照国家升序，年份降序。

```
select
    nation,
    o_year,
```

```
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%green%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
```

2.10 退货报告查询

根据给定的零件 key，查找出收入损失最多的前 20 个客户，结果返回客户 key，客户名字，收入损失，客户账户余额，客户所在国家，客户地址，客户电话，备注，查询结果按照收入损失降序排序。

```
select
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
```

```
        c_address,
        c_phone,
        c_comment
from
        customer,
        orders,
        lineitem,
        nation
where
        c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and o_orderdate >= date '2016-10-01'
        and o_orderdate < date '2016-10-01' + interval '3' month
        and l_returnflag = 'R'
        and c_nationkey = n_nationkey
group by
        c_custkey,
        c_name,
        c_acctbal,
        c_phone,
        n_name,
        c_address,
        c_comment
order by
        revenue desc;
LIMIT 10
```

2.11 重要库存识别查询

给出国家和百分比，查找出供应商库存里超过这个国家总库存价值百分比的所有库存，结果返回零件key，库存价值，按照价值降序排序。

```
select
        ps_partkey,
        sum(ps_supplycost * ps_availqty) as value
from
        partsupp,
        supplier,
        nation
where
        ps_suppkey = s_suppkey
```

```
    and s_nationkey = n_nationkey
    and n_name = 'GERMANY'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) * 0.0001000000
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'GERMANY'
        )
order by
    value desc;
```

2.12 运送方式和订单优先级查询

这条查询语句用来确定选择便宜的货运模式是否会导致消费者更多在承诺日期之后收到货物，对订单优先级条目产生影响。给定日期和两种货运方式，在给定日期一年内收货并且属于两种货运模式的订单明细中，选出比承诺日期晚送达的，统计两种货运模式下，订单优先级为”1-URGENT”和”2-HIGH”的数量与不为这两种的数量，比较区别。结果返回货运模式，不同订单优先级数量。

```
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count
from
```

```
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('MAIL', 'SHIP')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '2017-01-01'
    and l_receiptdate < date '2017-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;
```

2.13 客户分布查询

根据客户订单数量确定客户分布，包括没有下过订单的客户，统计不同订单数量的客户数量，并且要在备注中检查订单不属于几种特殊类别的订单，结果返回订单数量和客户数量，按照客户数量和订单数量降序。

```
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%special%requests%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
```

2.14 促销效果查询

给定日期，查询这个日期后一个月内的收入有多大比例来自促销零件，结果返回促销零件收入所占的百分比。

```
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= date '2018-09-01'
    and l_shipdate < date '2018-09-01' + interval '1' month;
```

2.15 顶级供应商查询

查询给定时间内收入最多的供应商，结果返回供应商 key，名称，地址，电话，和总收入，按照供应商 key 升序。

```
create view revenue0 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= date '2019-01-01'
        and l_shipdate < date '2019-01-01' + interval '3' month
    group by
        l_suppkey;


select
    s_suppkey,
    s_name,
```

```
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue0
    )
order by
    s_suppkey;

drop view revenue0;
```

2.16 零部件/供应商关系查询

这个查询可以超出有多少供应商能够按照给定条件供应，且没有被客户投诉过，给定零件品牌，类型，大小，结果返回零件品牌，类型，大小和供应商数量，按照供应商数量降序，品牌，类型，大小升序。

```
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#45'
    and p_type not like 'MEDIUM POLISHED%'
    and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
    and ps_suppkey not in (
        select
            s_suppkey
        from
```

```
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
```

2.17 小额订单收入查询

给定零件品牌和包装查询比平均供货量百分之 20 还小的小额订单，结果返回这些订单收入在七年内的平均值。

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

2.18 大批量客户查询

给定数量，查询购买数量比指定数量大的客户信息，结果返回客户名字，客户 key，订单 key，下单日期，订单总价，购买数量，结果按照购买数量降序，下单日期升序，返回前 100 个。

```
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 300
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate;
LIMIT 100;
```

2.19 折扣收入查询

查询一些空运或者人工运输的订单的折扣收入，零件按照给定的组合进行选择，结果返回总折扣收入。

```
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#12'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 1 and l_quantity <= 1 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#23'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 10 and l_quantity <= 10 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#34'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
```

2.20 潜在零部件促销查询

给定国家和开始时间，查询在一年时间内，该国对给定条件的零件有超过百分之 50 供应量的供应商，结果返回供应商名字和地址，按照供应商名字升序。

```
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'forest%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= date '2017-01-01'
                    and l_shipdate < date '2017-01-01' + interval '1' year
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'CANADA'
order by
    s_name;
```

2.21 不能按时交货供应商查询

给定国家，查找出具有多个供货商的订单中唯一一个没有按时交货的供货商，返回供货商名字和没有交货的订单数量，按照订单数量降序，名字升序。

```
select
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'SAUDI ARABIA'
group by
```

```
    s_name
order by
    numwait desc,
    s_name;
```

2.22 全球销售机会查询

查询指定国家代码内，对应的客户没有下七年的订单，但是账户余额大于平均账户余额的客户，结果返回国家代码，客户数量，总账户余额，按照国家代码升序。

```
select
    cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone from 1 for 2) as cntrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone from 1 for 2) in
                ('13', '31', '23', '29', '30', '18', '17')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substring(c_phone from 1 for 2) in
                        ('13', '31', '23', '29', '30', '18', '17')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
```

```
                    o_custkey = c_custkey
                )
    ) as custsale
group by
    cntrycode
order by
    cntrycode;
```

3. 附录 3. TPC-H 数据库建库脚本

3.1建表及数据导入步骤

建议：

通过下述步骤在数据库中建立 TPC-H 中的 8 张关系表，并向表中导入数据，构建 TPC-H 数据库：

（1）使用 5.1-5.8 给出的 8 条 create table 建表语句，建立不带完整性约束的关系表

（2）使用数据库管理系统提供的数据导入机制，将 customer.txt 等数据文件中的数据导入这 8 张表。

具体数据导入方法参见“实验指导书-03-1 openGauss 数据库建表及数据导入-22-v1.0”。

（3）使用 5.9 中的 SQL 语句，在这 8 张表上添加约束

3.2订单表 ORDERS

```
CREATE TABLE ORDERS    ( O_ORDERKEY          INTEGER NOT NULL,
                         O_CUSTKEY           INTEGER NOT NULL,
                         O_ORDERSTATUS       CHAR(1) NOT NULL,
                         O_TOTALPRICE        DECIMAL(15,2) NOT NULL,
                         O_ORDERDATE         DATE NOT NULL,
                         O_ORDERPRIORITY     CHAR(15) NOT NULL,
                         O_CLERK             CHAR(15) NOT NULL,
                         O_SHIPPRIORITY      INTEGER NOT NULL,
                         O_COMMENT           VARCHAR(79) NOT NULL);
```

3.3区域表 REGION

```
CREATE TABLE REGION    ( R_REGIONKEY    INTEGER NOT NULL,
                         R_NAME         CHAR(25) NOT NULL,
                         R_COMMENT      VARCHAR(152));
```

3.4国家表 NATION

```
CREATE TABLE NATION    ( N_NATIONKEY    INTEGER NOT NULL,
```

```
    N_NAME          CHAR(25) NOT NULL,
    N_REGIONKEY     INTEGER NOT NULL,
    N_COMMENT       VARCHAR(152));
```

3.5供应商表 SUPPLIER

```
CREATE TABLE SUPPLIER ( S_SUPPKEY       INTEGER NOT NULL,
                        S_NAME          CHAR(25) NOT NULL,
                        S_ADDRESS       VARCHAR(40) NOT NULL,
                        S_NATIONKEY     INTEGER NOT NULL,
                        S_PHONE         CHAR(15) NOT NULL,
                        S_ACCTBAL       DECIMAL(15,2) NOT NULL,
                        S_COMMENT       VARCHAR(101) NOT NULL);
```

3.6零部件表 PART

```
CREATE TABLE PART    ( P_PARTKEY       INTEGER NOT NULL,
                       P_NAME          VARCHAR(55) NOT NULL,
                       P_MFGR          CHAR(25) NOT NULL,
                       P_BRAND         CHAR(10) NOT NULL,
                       P_TYPE          VARCHAR(25) NOT NULL,
                       P_SIZE          INTEGER NOT NULL,
                       P_CONTAINER     CHAR(10) NOT NULL,
                       P_RETAILPRICE DECIMAL(15,2) NOT NULL,
                       P_COMMENT       VARCHAR(23) NOT NULL );
```

3.7零部件供应表 PARTSUPP

```
CREATE TABLE PARTSUPP ( PS_PARTKEY      INTEGER NOT NULL,
                        PS_SUPPKEY      INTEGER NOT NULL,
                        PS_AVAILQTY     INTEGER NOT NULL,
                        PS_SUPPLYCOST   DECIMAL(15,2)   NOT NULL,
                        PS_COMMENT      VARCHAR(199) NOT NULL );
```

3.8 客户表 CUSTOMER

```
CREATE TABLE CUSTOMER ( C_CUSTKEY        INTEGER NOT NULL,
                        C_NAME           VARCHAR(25) NOT NULL,
                        C_ADDRESS        VARCHAR(40) NOT NULL,
                        C_NATIONKEY      INTEGER NOT NULL,
                        C_PHONE          CHAR(15) NOT NULL,
                        C_ACCTBAL        DECIMAL(15,2)     NOT NULL,
                        C_MKTSEGMENT     CHAR(10) NOT NULL,
                        C_COMMENT        VARCHAR(117) NOT NULL);
```

3.9订单明细表 LINEITEM

```
CREATE TABLE LINEITEM ( L_ORDERKEY       INTEGER NOT NULL,
                        L_PARTKEY        INTEGER NOT NULL,
                        L_SUPPKEY        INTEGER NOT NULL,
                        L_LINENUMBER     INTEGER NOT NULL,
                        L_QUANTITY       DECIMAL(15,2) NOT NULL,
                        L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
                        L_DISCOUNT       DECIMAL(15,2) NOT NULL,
                        L_TAX            DECIMAL(15,2) NOT NULL,
                        L_RETURNFLAG     CHAR(1) NOT NULL,
                        L_LINESTATUS     CHAR(1) NOT NULL,
                        L_SHIPDATE       DATE NOT NULL,
                        L_COMMITDATE     DATE NOT NULL,
                        L_RECEIPTDATE DATE NOT NULL,
                        L_SHIPINSTRUCT CHAR(25) NOT NULL,
                        L_SHIPMODE       CHAR(10) NOT NULL,
                        L_COMMENT        VARCHAR(44) NOT NULL);
```

3.10 完整性约束

```
ALTER TABLE REGION
ADD PRIMARY KEY (R_REGIONKEY);

ALTER TABLE NATION
ADD PRIMARY KEY (N_NATIONKEY);
```

```
ALTER TABLE NATION
ADD FOREIGN KEY (N_REGIONKEY) references REGION;

ALTER TABLE PART
ADD PRIMARY KEY (P_PARTKEY);

ALTER TABLE SUPPLIER
ADD PRIMARY KEY (S_SUPPKEY);

ALTER TABLE SUPPLIER
ADD FOREIGN KEY (S_NATIONKEY) references NATION;

ALTER TABLE PARTSUPP
ADD PRIMARY KEY (PS_PARTKEY,PS_SUPPKEY);

ALTER TABLE CUSTOMER
ADD PRIMARY KEY (C_CUSTKEY);

ALTER TABLE CUSTOMER
ADD FOREIGN KEY (C_NATIONKEY) references NATION;

ALTER TABLE LINEITEM
ADD PRIMARY KEY (L_ORDERKEY,L_LINENUMBER);

ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);

ALTER TABLE PARTSUPP
ADD FOREIGN KEY (PS_SUPPKEY) references SUPPLIER;

ALTER TABLE PARTSUPP
ADD FOREIGN KEY (PS_PARTKEY) references PART;

ALTER TABLE ORDERS
ADD FOREIGN KEY (O_CUSTKEY) references CUSTOMER;

ALTER TABLE LINEITEM
ADD FOREIGN KEY (L_ORDERKEY)    references ORDERS;

ALTER TABLE LINEITEM
ADD FOREIGN KEY (L_PARTKEY,L_SUPPKEY) references PARTSUPP;
```

4. 附录 4 TPC-H 测试数据和 SQL 测试模板生成

利用 TPC-HTPC-H 测试程序，进行测试的步骤如下：

Step1. 利用 TPC-HTPC-H 测试程序，生成测试数据；

Step2. 利用 TPC-HTPC-H 测试程序，生成 22 条 TPC-H SQL 测试语句对应的 SQL 测试模板；

Step3. 采用 Python、Java 等，编写测试程序，测试程序中包括多个 SQL 测试模板；将多个 SQL 测试模板通过 1 个或多个终端，提交给数据库系统，并发执行这些 SQL 测试模板。

TPC-H 测试程序下载地址:
https://www.tpc.org/TPC_Documents_Current_Versions/download_programs/tools-download-request5.asp?bm_type=TPC-H&bm_vers=3.0.1&mode=CURRENT-ONLY

4.1 编译

打开 dbgen 目录，首先在 tpcd.h 中创建 openGauss 数据库的格式

```
#ifdef OPENGAUSS
#define GEN_QUERY_PLAN    "EXPLAIN"
#define START_TRAN          "BEGIN TRANSACTION"
#define END_TRAN             "COMMIT;"
#define SET_OUTPUT          ""
#define SET_ROWCOUNT       "LIMIT %d\n"
#define SET_DBASE           ""
#endif /* VECTORWISE */
```

然后修改 makefile.suite 文件中的参数：

```
CC          = gcc
DATABASE = POSTGRESQL
MACHINE = LINUX
WORKLOAD = TPCH
```

在 dbgen 目录下打开终端，执行命令

```
make -f makefile.suite
```

编译成功会在 dbgen 目录下生成 dbgen 和 qgen 可执行文件

4.2 生成测试数据

在 dbgen 目录下打开终端，执行
./dbgen -vf -s 1
-s 参数后面的数字表示生成数据规模，1 表示生成 1GB 数据，想要生成更大规模的数据修改参数即可。
然后在 opengauss 数据库中创建对应的数据库和数据表，将数据导入数据表

4.3 生成 SQL 查询语句及测试模板

复制 qgen 和 dists.dss 文件复制到到 dbgen 下的 queries 文件夹，在 queries 文件夹中执行：
./qgen -d 1 >d1.sql
其中-d 表示默认参数，如果想随机生成查询中的数字可以删掉-d 参数，1 > query1.sql 表示按照查询模板 1 生成 sql 查询到 query1.sql 文件，queries 目录中一共有 22 个模板，更换模板修改数字即可，query1.sql 为目标文件名。

5. 附录 5 . TPC-C 数据库

5.1关系模式图

TPC-C 数据库的组成部分定义为由九个单独的表和索引表组成。

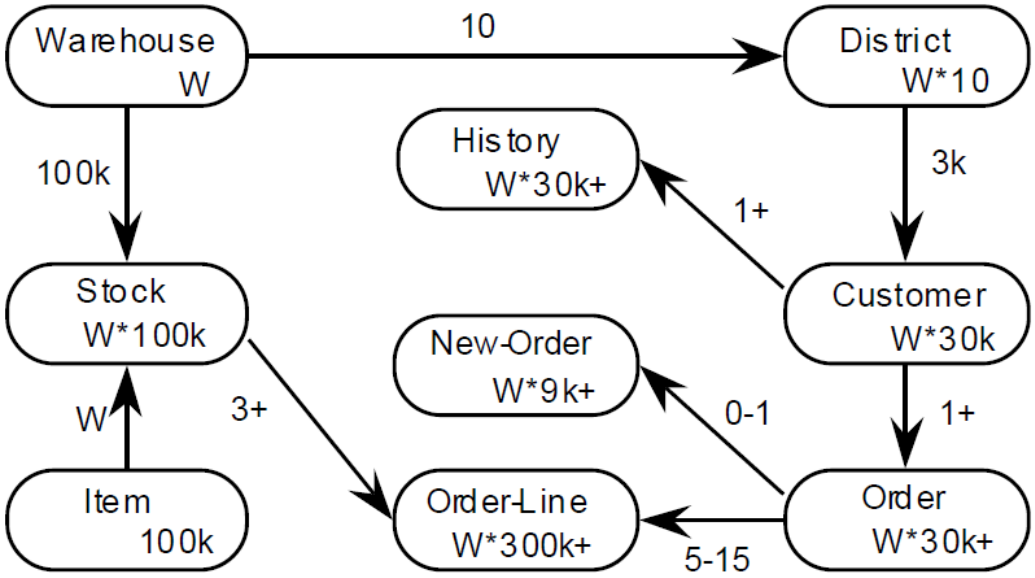

图 7　TPC-C 关系图

说明：

- 显示的所有数字都说明了数据库填充要求
- 实体块中的数字代表表的基数（行数）。 这些数字由仓库数 W 分解，以说明数据库扩展。
- 关系箭头旁边的数字代表关系的基数。
- 加号（+）用于关系或表的基数之后，以说明在添加或删除行时，该数字在测量间隔内的初始数据库总体中会发生微小变化。

5.2仓库表 WAREHOUSE

属性名称	字段定义	注释， 完整性/约束说明
W_ID	2*W unique IDs	W Warehouses are populated 主键
W_NAME	variable text, size 10	
W_STREET_1	variable text, size 20	
W_STREET_2	variable text, size 20	
W_CITY	variable text, size 20	

W_STATE	fixed text, size 2	
W_ZIP	fixed text, size 9	
W_TAX	signed numeric(4,4)	Sales tax
W_YID	signed numeric(12,2)	Year to date balance

5.3区域表 DISTRICT

属性名称	字段定义	注释， 完整性/约束说明
D_ID	20 unique IDs	10 are populated per warehouse 主键
D_W_ID	2*W unique IDs	主键/外键，参考 W_ID
D_NAME	variable text, size 10	
D_STREET_1	variable text, size 20	
D_STREET_2	variable text, size 20	
D_CITY	variable text, size 20	
D_STATE	fixed text, size 2	
D_ZIP	fixed text, size 9	
D_TAX	signed numeric(4,4)	Sales tax
D_YTD	signed numeric(12,2)	Year to date balance
D_NEXT_O_ID	10,000,000 unique IDS	Next available Order number

5.4客户表 CUSTOMER

属性名称	字段定义	注释， 完整性/约束说明
C_ID	96,000 unique IDs	3,000 are populated per district 主键
C_D_ID	20 unique IDs	主键/外键，参考 D_ID
C_W_ID	2*W unique IDs	主键/外键。参考 D_W_ID
C_FIRST	variable text, size 16	
C_MIDDLE	fixed text, size 2	
C_LAST	variable text, size 16	
C_STREET_1	variable text, size 20	
C_STREET_2	variable text, size 20	
C_CITY	variable text, size 20	Sales tax
C_STATE	fixed text, size 2	Year to date balance
C_ZIP	fixed text, size 9	Next available Order number
C_PHONE	fixed text, size 16	
C_SINCE	date and time	
C_CREDIT	fixed text, size 2	"GC"=good, "BC"=bad
C_CREDIT_LIM	signed numeric(12, 2)	
C_DISCOUNT	signed numeric(4, 4)	
C_BALANCE	signed numeric(12, 2)	
C_YTD_PAYMENT	signed numeric(12, 2)	
C_PAYMENT_CNT	numeric(4)	
C_DELIVERY_CNT	numeric(4)	

C_DATA	variable text, size 500	Miscellaneous information

5.5历史记录表 HISTORY

属性名称	字段定义	注释， 完整性/约束说明
H_C_ID	96,000 unique IDs	外键，参考 C_ID
H_C_D_ID	20 unique IDs	外键，参考 C_D_ID
H_C_W_ID	2*W unique IDs	外键，参考 C_W_ID
H_D_ID	20 unique IDs	外键，参考 D_ID
H_W_ID	2*W unique IDs	外键，参考 D_W_ID
H_DATE	date and time	
H_AMOUNT	signed numeric(6, 2)	
H_DATA	variable text, size 24	Miscellaneous information

5.6新订单表 NEW-ORDER

属性名称	字段定义	注释， 完整性/约束说明
NO_O_ID	10,000,000 unique IDs	主键/外键，参考 O_ID
NO_C_ID	20 unique IDs	主键/外键，参考 O_D_ID
NO_W_ID	2*W unique IDs	主键/外键，参考 O_W_ID

5.7订单表 ORDER

属性名称	字段定义	注释， 完整性/约束说明
O_ID	10,000,000 unique IDs	主键
O_D_ID	20 unique IDs	主键/外键，参考 C_D_ID
O_W_ID	2*W unique IDs	主键/外键，参考 C_W_ID
O_C_ID	96,000 unique IDs	外键，参考 C_ID

O_ENTRY_D	date and time	
O_CARRIER_ID	10 unique IDs, or null	
O_OL_CNT	numeric(2)	Count of Order-Lines
O_ALL_LOCAL	numeric(1)	

5.8订单行表 ORDER-LINE

属性名称	字段定义	注释， 完整性/约束说明
OL_O_ID	10,000,000 unique IDs	主键/外键，参考 O_ID
OL_D_ID	20 unique IDs	主键/外键，参考 O_D_ID
OL_W_ID	2*W unique IDs	主键/外键，参考 O_W_ID
OL_NUMBER	15 unique IDs	主键
OL_I_ID	200,000 unique IDs	外键，参考 S_I_ID
OL_SUPPLY_W_ID	2*W unique IDs	外键，参考 S_W_ID
OL_DELIVERY_D	date and time, or null	
OL_QUANTITY	numeric(2)	
OL_AMOUNT	signed numeric(6, 2)	
OL_DIST_INFO	fixed text, size 24	

5.9条目表 ITEM

属性名称	字段定义	注释， 完整性/约束说明
I_ID	200,000 unique IDs	100,000 items are populated 主键
I_M_ID	200,000 unique IDs	Image ID associated to Item
I_NAME	variable text, size 24	
I_PRICE	numeric(5, 2)	

I_DATA	variable text, size 50	Brand information

5.10 库存表 STOCK

属性名称	字段定义	注释， 完整性/约束说明
S_I_ID	200,000 unique IDs	100,000 populated per warehouse 主键/外键，参考 I_ID
S_W_ID	2*W unique IDs	主键/外键，参考 W_ID
S_QUANTITY	signed numeric(4)	
S_DIST_01	fixed text, size 24	
S_DIST_02	fixed text, size 24	
S_DIST_03	fixed text, size 24	
S_DIST_04	fixed text, size 24	
S_DIST_05	fixed text, size 24	
S_DIST_06	fixed text, size 24	
S_DIST_07	fixed text, size 24	
S_DIST_08	fixed text, size 24	
S_DIST_09	fixed text, size 24	
S_DIST_10	fixed text, size 24	
S_YTD	numeric(8)	
S_ORDER_CNT	numeric(4)	
S_REMOTE_CNT	numeric(4)	
S_DATA	variable text, size 50	Make information

6. 附录 6 . TPC-C 并发事务测试

面向联机事务处理 OLTP 的 TPC-C 测试事务由多条 SQL 语句组成，采用嵌入式 SQL 实现，编程语言为 C 或 JAVA。

包含详细代码及说明的 pdf 文件可从此处下载（TPC-C）：

https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

文档下载地址：https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf

五种 TPC-C 事务的定义和说明见文档第二章，五种事务的嵌入式 C 程序见附录。

BenchmarkSQL 是 TPC-C 的 java 实现

6.1五种 TPC-C 基本事务

1．New-Order：客户输入一笔新的订货交易
事务内容：对于任意一个客户端，从固定的仓库随机选取 5-15 件商品，创建新订单。
2．Payment：更新客户账户余额以反应其支付状况
事务内容：对于任意一个客户端，从固定的仓库随机选取一个辖区及其内用户，采用随机的金额支付一笔订单，并作相应历史纪录。
3．Delivery：发货(批处理交易)
事务内容：对于任意一个客户端，随机选取一个发货包，更新被处理订单的用户余额，并把该订单从新订单中删除。
4．Order-Status：查询客户最近交易的状态
事务内容：对于任意一个客户端，从固定的仓库随机选取一个辖区及其内用户，读取其最后一条订单，显示订单内每件商品的状态。
5．Stock-Level：查询仓库库存状况，以便能够及时补货。

6.2基于 BenchmarkSQL 的 TPC-C 测试过程

BenchmarkSQL 是采用 Java 实现的 TPC-C 测试基准，包括(1) TPC-C 测试数据库，由 10 张表组成；(2) 5 个由 Java 实现的基本测试事务。

测试过程分为三步：

Step 1. 建立 TPC-C 测试库，配置测试数据参数，生成 10 张表的测试数据，并导入数据库表；
Step 2. 配置客户端事务参数，生成测试事务；
Step 3. 启动终端测试事务，进行测试。

测试数据和测试事务的生成程序的下载地址如下：
https://sourceforge.net/projects/benchmarksql/files/benchmarksql-5.0.zip/download

下面以 benchmarkSQL5.0 为例，说明 TPC-C 的测试数据、测试事务的配置过程，以及测试过程。

1. 配置 JAVA 环境

BenchmarkSQL 需要 JDK7 及以上的 JAVA 环境支持。

2. 修改配置文件，规定测试数据的规模和 10 张数据库表中的数据总量及其比例

run 目录下的 propg.pg 文件为 BenchmarkSQL 设置数据及数据库的配置文件。其中数据库相关的设置为：

```
db=postgres
driver=org.postgresql.Driver
```

conn=jdbc:postgresql://ip:port/databasename
user=gaussdb
password=openGauss@123

连接 opengauss 数据库 driver 使用 org.postgresql.Driver，user 和 password 为用户名和密码，db 设置为 postgres 即可，conn 设置为 conn=jdbc:postgresql://ip:port/databasename，修改对应的 ip，端口号和数据库名字。

propg.pg 文件中关于数据和测试的设置如下，按照本机配置合理设置：

//仓库数量
warehouses=4
说明：
（1）本配置定义了仓库数量，即规定了仓库表 WAREHOUSE 中的数据总量。
（2）TPC-C 数据库各张表的数据总量维持一定比例关系，当定义了仓库表 WAREHOUSE 的规模后，其它 9 张表的数据总量也就随之确定，TPC-C 测试数据总量也确定下来了。

TPC-C 测试数据生成程序将按此配置生成 10 张表中的测试数据。

//初始化数据的加载进程数量，默认为 4，每个仓库大小约为 100MB，实际使用可以结合 os 性能做配置
loadWorkers=4

3. 配置客户端/终端数量

//终端数，即并发客户端数量，通常设置为 CPU 线程总数的 2～6 倍，
terminals=4

用户通过客户端访问服务器端数据库。该配置定义了可并发访问数据库端服务器的客户端数量，每个客户端上可串行运行多个 TPC-C 事务。

4. 配置终端可运行的串行 TPC-C 事务总数

//每个客户端/终端 terminal 可运行的固定事务数量
runTxnsPerTerminal=10

当该值为 10 时，每个 terminal 运行 10 个事务。如果有 32 个终端，即 terminals=32，那整体运行 320 个事务后，TPC-C 测试结束。

当参数 runTxnsPerTerminal 配置为非 0 值时，下面的 runMins 参数必须设置为 0。

runTxnsPerTerminal=0 //Number of total transactions per minute

参数 runMins 表示要压测的时间长度，单位为分钟。

参数 runMins 的值为非 0 值时，runTxnsPerTerminal 参数必须设置为 0。这两个参数不能同时设置为正整数，如果设置其中一个，另一个必须为 0，例如：

runMins=5

limitTxnsPerMin=0

主要区别是 runMins 定义时间长度来控制测试时间，runTxnsPerTerminal 定义事务总数来控制时间。

/

/终端和仓库的绑定模式，设置为 true 时可以运行 4.x 兼容模式，意思为每个终端都有一个固定的仓库。设置为 false 时可以均匀的使用数据库整体配置。TPCC 规定每个终端都必须有一个绑定的仓库，所以一般使用默认值 true。

terminalWarehouseFixed=true

5. 配置终端可运行的串行事务的类型和比例

在一次 TPC-C 测试过程中，一个终端可顺序执行一组串行事务，所执行的串行事务总数由参数 runTxnsPerTerminal 来确定。这些串行事务由 5 种 TPC-C 基本事务组成，即 **New-Order、Payment、Delivery、Order-Status、Stock-Level。**

在一个终端执行的一组串行事务中，5 个 TPC-C 基本事务的数量比例由以下配置事务权重 weight 来定义：

newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4

说明：

（1）以上五个 weight 值的总和必须等于 100，默认值为：45, 43, 4, 4，4，与 TPC-C 测试定义的比例一致。实际操作过程中，可以调整比重来适应各种场景。

（2）采用上述配置确定的一组串行事务中，5 各个基本事务的总数量和每个事务数量的比例关系是确定的，但 5 个事务出现的先后顺序由测试程序随机生成。例如，对于以下配置：

runTxnsPerTerminal=5
newOrderWeight=20
paymentWeight=20
orderStatusWeight=20
deliveryWeight=20
stockLevelWeight=20

，生成的串行事务可以是

New-Order; Payment; Delivery; Order-Status; Stock-Level

，也可以是

Payment; New-Order; Order-Status; Delivery; Stock-Level

6. 生成测试数据和启动 TPC-C 测试

在 run 目录下打开终端执行下面两条命令，第一条是创建数据表并生成导入数据，第二条是执行测试

./runDatabaseBuild.sh props.pg

./runBenchmark.sh props.pg

测试完成后可以执行下面命令删除数据

./runDatabaseDestroy.sh props.pg

测试完成后会输出以下指标：

Running Average tpmTOTAL：每分钟平均执行事务数（所有事务）

Memory Usage：客户端内存使用情况

Measured tpmC (NewOrders) ：每分钟执行的事务数（只统计 NewOrders 事务）

Transaction Count：执行的交易总数量

6.3基本事务 A1： The New-Order Transaction

新订单事务包括通过单个数据库事务输入完整的订单。它代表了一种中等等级的读写事务，具有高执行频率和严格的响应时间要求，以满足在线用户的需求。此事务是工作负载的支柱。它旨在对系统施加可变负载，以反映生产环境中常见的在线数据库活动。

新订单事务过程说明：

对于给定的仓库编号（W_ID）、地区编号（D_W_ID、D_ID）、客户编号（C_W_ID、C_D_ID、C_ID）、物品计数（ol_cnt）以及给定的一组物品（OL_I_ID ）、供应仓库 (OL_SUPPLY_W_ID) 和数量 (OL_QUANTITY)：

1) 选择 WAREHOUSE 表中与 W_ID 匹配的行，并检索仓库税率 W_TAX。同时选择 CUSTOMER 表中匹配 C_W_ID、C_D_ID 和 C_ID 的行，并检索客户贴现率 C_DISCOUNT、客户姓氏 C_LAST 和客户信用状态 C_CREDIT。
2) 选择 DISTRICT 表中 D_W_ID 和 D_ID 匹配的行，检索地区税率 D_TAX，检索该地区的下一个可用订单号 D_NEXT_O_ID 并加一。
3) 在 NEW-ORDER 表和 ORDER 表中都插入了一个新行，以反映新订单的创建。O_CARRIER_ID 设置为空值。 如果订单仅包括主订单行，则 O_ALL_LOCAL 设置为 1，否则 O_ALL_LOCAL 设置为 0。计算项目数 O_OL_CNT 以匹配 ol_cnt。
4) 对于订单上的每个 O_OL_CNT 项目：

   - 选择 ITEM 表中具有匹配 I_ID（等于 OL_I_ID）的行，并检索项目价格 I_PRICE、项目名称 I_NAME 和 I_DATA。如果 I_ID 有一个未使用的值，则发出“未找到”条件，导致数据库事务回滚。

   - 选择 STOCK 表中匹配 S_I_ID（等于 OL_I_ID）和 S_W_ID（等于 OL_SUPPLY_W_ID）的行。检索库存数量 S_QUANTITY，S_DIST_xx（其中 xx 代表区号），S_DATA。如果 S_QUANTITY 的检索值超过 OL_QUANTITY 10 或更多，则 S_QUANTITY 减少 OL_QUANTITY；否则 S_QUANTITY 更新为 (S_QUANTITY - OL_QUANTITY)+91。S_YTD 增加 OL_QUANTITY 并且 S_ORDER_CNT 增加 1。如果订单行是远程的，则 S_REMOTE_CNT 增加 1。

- 订单中商品的金额 (OL_AMOUNT) 计算为：
OL_QUANTITY * I_PRICE

- 检查 I_DATA 和 S_DATA 中的字符串。如果它们都包含字符串“ORIGINAL”，则该项目的品牌通用字段设置为“B”，否则，品牌通用字段设置为“G”。

- 将新行插入到 ORDER-LINE 表中以反映订单上的项目。 OL_DELIVERY_D 设置为空值，OL_NUMBER 设置为在所有具有相同 OL_O_ID 值的 ORDER-LINE 行中的唯一值，OL_DIST_INFO 设置为 S_DIST_xx 的内容，其中 xx 代表区号（OL_D_ID）

5) 完整订单的总金额计算如下：
总和（OL_AMOUNT）*（1 - C_DISCOUNT）*（1 + W_TAX + D_TAX）
- 数据库事务被提交，除非它由于最后一个项目编号的未使用值而被回滚。
- 输出数据被传送到终端。

代码：

```
int neword()
{
EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

gettimestamp(datetime);

EXEC SQL SELECT c_discount, c_last, c_credit, w_tax
      INTO :c_discount, :c_last, :c_credit, :w_tax
      FROM customer, warehouse
      WHERE w_id = :w_id AND c_w_id = w_id AND
            c_d_id = :d_id AND c_id = :c_id;

EXEC SQL SELECT d_next_o_id, d_tax INTO :d_next_o_id, :d_tax
      FROM district
      WHERE d_id = :d_id AND d_w_id = :w_id;

EXEC SQL UPDATE district SET d_next_o_id = :d_next_o_id + 1
      WHERE d_id = :d_id AND d_w_id = :w_id;

o_id=d_next_o_id;

EXEC SQL INSERT INTO ORDERS (o_id, o_d_id, o_w_id, o_c_id,
                        o_entry_d, o_ol_cnt, o_all_local)
      VALUES (:o_id, :d_id, :w_id, :c_id,
            :datetime, :o_ol_cnt, :o_all_local);
```

```
EXEC SQL INSERT INTO NEW_ORDER (no_o_id, no_d_id, no_w_id)
    VALUES (:o_id, :d_id, :w_id);

for (ol_number=1; ol_number<=o_ol_cnt; ol_number++)
{
    ol_supply_w_id=atol(supware[ol_number-1]);
    if (ol_supply_w_id != w_id) o_all_local=0;
    ol_i_id=atol(itemid[ol_number-1]);
    ol_quantity=atol(qty[ol_number-1]);

EXEC SQL WHENEVER NOT FOUND GOTO invaliditem;

EXEC SQL SELECT i_price, i_name , i_data
    INTO :i_price, :i_name, :i_data
    FROM item
    WHERE i_id = :ol_i_id;

price[ol_number-1] = i_price;
strncpy(iname[ol_number-1],i_name,24);

EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;

EXEC SQL SELECT s_quantity, s_data,
    s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05
    s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
    INTO :s_quantity, :s_data,
    :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05
    :s_dist_06, :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10
    FROM stock
    WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id;

pick_dist_info(ol_dist_info, ol_w_id); // pick correct s_dist_xx
stock[ol_number-1] = s_quantity;

if ( (strstr(i_data,"original") != NULL) &&
    (strstr(s_data,"original") != NULL) )
    bg[ol_number-1] = 'B';
else
    bg[ol_number-1] = 'G';

if (s_quantity > ol_quantity)
```

```
   s_quantity = s_quantity - ol_quantity;
else
   s_quantity = s_quantity - ol_quantity + 91;

EXEC SQL UPDATE stock SET s_quantity = :s_quantity
      WHERE s_i_id = :ol_i_id
      AND s_w_id = :ol_supply_w_id;

ol_amount = ol_quantity * i_price * (1+w_tax+d_tax) * (1-c_discount);
amt[ol_number-1]=ol_amount;
total += ol_amount;

EXEC SQL INSERT
      INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number,
                  ol_i_id, ol_supply_w_id,
                  ol_quantity, ol_amount, ol_dist_info)
      VALUES (:o_id, :d_id, :w_id, :ol_number,
            :ol_i_id, :ol_supply_w_id,
            :ol_quantity, :ol_amount, :ol_dist_info);
} /*End Order Lines*/

EXEC SQL COMMIT WORK;
return(0);

invaliditem:
   EXEC SQL ROLLBACK WORK;
   printf("Item number is not valid");
   return(0);

sqlerr:
   error();
}
```

6.4基本事务 A2： The Payment Transaction

支付事务更新客户的余额，并将支付反映在地区和仓库销售统计上。 它代表了一种轻量级的读写事务，具有高执行频率和严格的响应时间要求，以满足在线用户的需求。 此外，此事务包括对 CUSTOMER 表的非主键访问。

支付事务过程说明：

对于给定的仓库编号（W_ID）、地区编号（D_W_ID、D_ID）、客户编号（C_W_ID、C_D_ID、C_ID）或客户姓氏（C_W_ID、C_D_ID、C_LAST）和付款金额（H_AMOUNT）：

1) WAREHOUSE 表中具有匹配 W_ID 的行被选中。检索 W_NAME、W_STREET_1、W_STREET_2、W_CITY、W_STATE 和 W_ZIP，并将仓库年初至今余额 W_YTD 增加 H_AMOUNT。
2) 选择 DISTRICT 表中 D_W_ID 和 D_ID 匹配的行。检索 D_NAME、D_STREET_1、D_STREET_2、D_CITY、D_STATE 和 D_ZIP，并且 D_YTD（该地区的年初至今余额）增加 H_AMOUNT。
3) 根据客户姓氏选择客户：
   CUSTOMER 表中所有匹配 C_W_ID、C_D_ID 和 C_LAST 的行都按照 C_FIRST 升序选择。 设 n 为选定的行数。 C_ID、C_FIRST、C_MIDDLE、C_STREET_1、C_STREET_2、C_CITY、C_STATE、C_ZIP、C_PHONE、C_SINCE、C_CREDIT、C_CREDIT_LIM、C_DISCOUNT 和 C_BALANCE 从排序集中位置（n/2 向上舍入到下一个整数）处的行检索从 CUSTOMER 表中选择的行数。C_BALANCE 减少 H_AMOUNT。C_YTD_PAYMENT 增加了 H_AMOUNT。C_PAYMENT_CNT 增加 1。
4) 如果 C_CREDIT 的值等于“BC”，则还从所选客户检索 C_DATA，并在 C_DATA 字段的左侧插入以下历史信息：C_ID、C_D_ID、C_W_ID、D_ID、W_ID 和 H_AMOUNT。将 C_DATA 的现有内容向右移动相等数量的字节，并丢弃移出 C_DATA 字段右侧的字节。C_DATA 字段的内容不超过 500 个字符。使用新的 C_DATA 字段更新选定的客户。如果 C_DATA 实现为两个字段，则必须将它们视为一个字段并对其进行操作。
5) H_DATA 是通过连接由 4 个空格分隔的 W_NAME 和 D_NAME 构建的。新行插入到 HISTORY 表中，H_C_ID = C_ID, H_C_D_ID = C_D_ID, H_C_W_ID =C_W_ID、H_D_ID = D_ID 和 H_W_ID = W_ID。
   - 数据库事务已提交。
   - 输出数据被传送到终端。

代码：

```
int payment()
{
    EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
    EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

    gettimestamp(datetime);
    EXEC SQL UPDATE warehouse SET w_ytd = w_ytd + :h_amount
        WHERE w_id=:w_id;

    EXEC SQL SELECT w_street_1, w_street_2, w_city, w_state, w_zip, w_name
        INTO :w_street_1, :w_street_2, :w_city, :w_state, :w_zip, :w_name
        FROM warehouse
        WHERE w_id=:w_id;

    EXEC SQL UPDATE district SET d_ytd = d_ytd + :h_amount
        WHERE d_w_id=:w_id AND d_id=:d_id;
```

```
        EXEC SQL SELECT d_street_1, d_street_2, d_city, d_state, d_zip, d_name
                INTO :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :d_name
                FROM district
                WHERE d_w_id=:w_id AND d_id=:d_id;

if (byname)
{
        EXEC SQL SELECT count(c_id) INTO :namecnt
                FROM customer
                WHERE c_last=:c_last AND c_d_id=:c_d_id AND c_w_id=:c_w_id;

        EXEC SQL DECLARE c_byname CURSOR FOR
                SELECT c_first, c_middle, c_id,
                    c_street_1, c_street_2, c_city, c_state, c_zip,
                    c_phone, c_credit, c_credit_lim,
                    c_discount, c_balance, c_since
                FROM customer
                WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_last=:c_last
                ORDER BY c_first;

EXEC SQL OPEN c_byname;

if (namecnt%2) namecnt++; // Locate midpoint customer;
for (n=0; n<namecnt/2; n++)
{
   EXEC SQL FETCH c_byname
        INTO :c_first, :c_middle, :c_id,
          :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
          :c_phone, :c_credit, :c_credit_lim,
          :c_discount, :c_balance, :c_since;
}
EXEC SQL CLOSE c_byname;
}
else
{
   EXEC SQL SELECT c_first, c_middle, c_last,
          c_street_1, c_street_2, c_city, c_state, c_zip,
          c_phone, c_credit, c_credit_lim,
          c_discount, c_balance, c_since
   INTO :c_first, :c_middle, :c_last,
          :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
          :c_phone, :c_credit, :c_credit_lim,
```

```
        :c_discount, :c_balance, :c_since
   FROM customer
   WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;
}
c_balance += h_amount;
c_credit[2]='\0';
if (strstr(c_credit, "BC") )
{
   EXEC SQL SELECT c_data INTO :c_data
       FROM customer
      WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;

sprintf(c_new_data,"| %4d %2d %4d %2d %4d $%7.2f %12c %24c",
                c_id,c_d_id,c_w_id,d_id,w_id,h_amount
                h_date, h_data);
strncat(c_new_data,c_data,500-strlen(c_new_data));

EXEC SQL UPDATE customer
       SET c_balance = :c_balance, c_data = :c_new_data
       WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND
             c_id = :c_id;
}
else
{
   EXEC SQL UPDATE customer SET c_balance = :c_balance
       WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND
             c_id = :c_id;
}
strncpy(h_data,w_name,10);
h_data[10]='\0';
strncat(h_data,d_name,10);
h_data[20]=' ';
h_data[21]=' ';
h_data[22]=' ';
h_data[23]=' ';

EXEC SQL INSERT INTO history (h_c_d_id, h_c_w_id, h_c_id, h_d_id,
                              h_w_id, h_date, h_amount, h_data)
       VALUES (:c_d_id, :c_w_id, :c_id, :d_id,
             :w_id, :datetime, :h_amount, :h_data);
EXEC SQL COMMIT WORK;
return(0);
```

```
sqlerr:
    error();
}
```

7. 附录 7 TPC-H 测试程序示例

下面给出 Python 程序示例，该程序连接数据库，通过数据库访问接口，提交 TPC-H SQL 测试模板，启动 TPC-H 测试。

7.1环境配置

Python 环境中需要安装数据库连接池的包 DBUtils，链接 opengauss 的包 psycopg2

7.2程序示例

```
import time
import psycopg2
import threading
from DBUtils.PooledDB import PooledDB, SharedDBConnection
POOL = PooledDB(
    creator= psycopg2,  # 使用链接数据库的模块
    maxconnections=6,  # 连接池允许的最大连接数，0 和 None 表示不限制连接数
    mincached=0,  # 初始化时，链接池中至少创建的空闲的链接，0 表示不创建
    maxcached=0,  # 链接池中最多闲置的链接，0 和 None 不限制
    maxshared=0,  # 链接池中最多共享的链接数量，0 和 None 表示全部共享
    blocking=True,  # 连接池中如果没有可用连接后，是否阻塞等待。True，等待；False，不等待然后报错
    maxusage=None,  # 一个链接最多被重复使用的次数，None 表示无限制
    setsession=[],  # 开始会话前执行的命令列表。
    ping=0,
    host='127.0.0.1',
    port=3306,
    user='root',
    password='123',
    database='databasename
)
```

```
def func():
    conn = POOL.connection()#连接数据库
    cursor = conn.cursor()#创建游标
    cursor.execute('select * from tb1') #执行 SQL
    //说明：此处根据需要，可填入具体的 TPC-H SQL 测试模板；
          可以向服务器传入多个 SQL 测试模板，执行并发 TPC-H 测试
    result = cursor.fetchall()#获取执行结果
    conn.close()#关闭连接
```

其中，host，port，user, database 为数据库信息，根据环境设置，maxconnections 根据 cpu 核数合理设置。