UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

Projecto 1 e Ficha/lab–TPC-H e benchmarking SGBDs

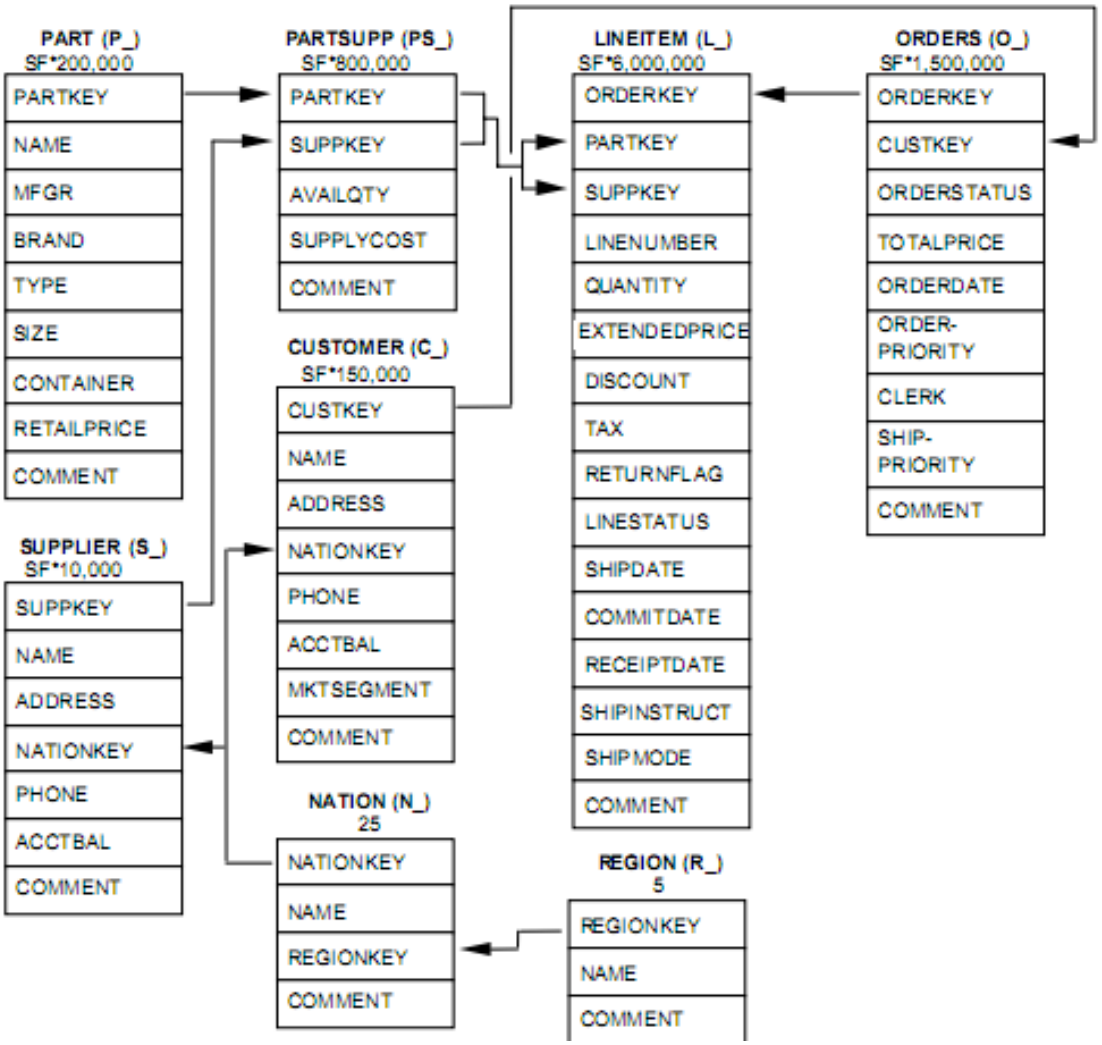
SGD

Ficha 1 – Advanced Analytics: TPC-H e benchmarking SGBDs

Objectivo: perceber como os motores de BD tradicionais lidam com grandes quantidades de dados e pesquisas complexas de forma eficiente. Preparar o projecto 1 que é feito depois autonomamente.

Objectivos da Avaliação do trabalho: impelir os grupos a fazerem a experiência, verificarem a eficiência e as dificuldades na execução, e a verem detalhes como os planos de execução. Será importante ter gráficos claros de comparação, análises e conclusões.

O TPC-H é um benchmark de bases de dados de apoio à decisão. Os benchmarks TPC são amplamente utilizados hoje na avaliação do desempenho dos sistemas de base de dados relacionais. Os resultados são publicados no site www.tpc.org.



22 queries complexas, exemplo Q1:

```
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 '1998-12-01' - interval '90' day
GROUP BY
    l_returnflag,
    l_linestatus
ORDER BY
    l_returnflag,
    l_linestatus;
```

Proj 1: (TPC-H perf bench e desenho do processamento)

A data final de submissão de entrega do projeto, com prazo pré-definido, está no inforestudante.

Gerar e carregar o benchmark TPC-H para **dois motores** de bases de dados (postgres e mysql) e comparar a velocidade a executar (avaliação de performance).
O TPC-H devera ser carregado com **40 GB de dados no Postgres e no MySQL**

Nota: o tamanho a usar deve depender na realidade do hardware que tem. Se o seu computador for moderno, por exemplo SSD 1TB com muito espaço livre, processador recente e 16 ou pelo menos 8 GB de memoria, 40 GB será adequado. Se tiver HDD ou de um modo geral se o seu computador for velho ou tiver pouca memoria, por exemplo, será mais adequado um tamanho de 25 GB ou até menos, mas reveja com o professor as limitações do seu sistema se quiser usar esses tamanhos mais pequenos. Será normalmente importante ter bastante espaço em disco extra, para alem do que necessita para a base de dados, porque se faltar memoria para processamento são criados ficheiros temporários em disco.

No Postgres deve correr também com 1 e com 5 threads simultâneas

Grupos de 2 alunos (ou 1), nao serão aceites mais elementos, ambos teem de saber tudo

Postgres + MySQL

Relatório com todos os resultados, ficheiros com resultados, ficheiros de código se usou código

Para cada motor e para comparação dos dois motores de BD:

1. gráfico de load time (carregar sem chaves)
2. gráfico de query time (pesquisar sem chaves), bem identificadas as queries e seus tempos
3. keys times (tempo de criação de cada chave PK e FK, bem identificado cada caso)
4. query times (pesquisas com chaves)
5. **Explain plan de uma query rapida e de uma lenta.**
6. **Explain plan mysql VS postgres para uma query lenta**

Proj 1 (English version): (TPC-H perf bench and processing design)

Groups have 2 elements.

In this part we intend to generate and load the TPC-H benchmark for two database engines (postgres and mysql) and compare the speed to be executed (performance evaluation). The TPC-H must be loaded with 40 GB of data.

For each engine and for comparison of the two BD engines:

1. load time graph (load without keys)
2. query time graph (search without keys), well identified queries and their times
3. keys times (creation time for each PK and FK key, well identified in each case)
4. query times (key searches)
5. Explain plan for a fast and a slow query.
6. Explain plan mysql VS postgres for a slow query

Lab – O que faremos

Nas aulas de TPC-H experimentamos instalar o TPC-H, mas com uma versão pequena de dataset. O projecto consta de, depois desta aula de experimentação, fazer avaliação mais completa, conforme indicado acima, de desempenho de SGBDs a correr carga de apoio a decisão.

UPLOAD: No final das aulas sobre TPC-H faz upload de todos os comandos e dos resultados que fizemos ou que completou na aula. No final das aulas sobre o assunto deverá ter instalado todas as tabelas com dados pequenos e corrido pelo menos as 5 primeiras pesquisas e obtido as respostas (se estiverem vazias pq tem muito poucos dados, mude um valor no where para experimentar ter valores).

Questões a ter em conta: tem de instalar o SGBD, haverá por vezes dificuldades com questões como indexação, chaves forasteiras e outros detalhes. Terá de experimentar.

Ficha 1 - Configurações do ambiente para postgres

Resumo:
Duas opções: se tiver Windows encontra um executável dentro de um zip que inclui as palavras TPCH e win ou algo do género; se tiver mac ou Linux, precisa de um zip que inclui as palavras TPCH e master ou algo do género (código fonte). Tem de configurar a makefile e depois correr o compilador de c para gerar o executável dbgen. Seguem-se essas instruções resumidas:

- To generate TPC-H compliant datasets, we must use the `dbgen` tool.
- Compile the `dbgen` tool by `make -f makefile.suite`.
    - Remember to modify `makefile.suite`.

```
Detalhes a modificar no makefile.suite:
CC       = gcc
DATABASE = INFORMIX
MACHINE  = LINUX
WORKLOAD = TPCH

Agora corra o make, por exemplo: ./make –f makefile.suite
```

For MacOS, the above `make` command will result in an error while compiling like below,

- `bm_utils.c:71:10: fatal error: 'malloc.h' file not found`
- `#include <malloc.h>`
- `^~~~~~~~~~`
- `1 error generated.`
  `make: *** [bm_utils.o] Error 1`
  To fix this, change the import statement `#include <malloc.h>` to `#include <sys/malloc.h>` in the files where error is reported (`bm_utils.c` and `varsub.c`) and then re-run the command `make`.

- Use the `dbgen` tool with the following options:
    - For example, you can use `./dbgen –s 1 –v`

1. Links para informação mais detalhada para configurar ambiente da experiência

Encontrei vários sítios com o código:

A. Pode fazer download de documentação e outros no site www.tpc.org. Pode fazer download do código de geração directamente do mesmo site, mas se o fizer terá de compilar de acordo com o que esta no Anexo 5. https://gist.github.com/yunpengn/6220ffc1b69cee5c861d93754e759d08:

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

B. Também disponibilizo no seguinte link uma versão já compilada para Windows e com as queries por defeito, para alem do código fonte:

https://drive.google.com/drive/folders/1fJPfm3udgfyIHKPMKGTOgmo3advGHda_?usp=drive_link

C. Outras fontes de código para dbgen (procurei em google por dbgen generate data):

experimentei esta e funcionou no meu mac, mas tive de alterar um detalhe como mostro no Anexo 4. Install and run from github:

https://docs.deistercloud.com/content/Databases/TPCH%20Benchmark/Data%20generation%20tool.xml

https://gist.github.com/yunpengn/6220ffc1b69cee5c861d93754e759d08

(Anexo 5. https://gist.github.com/yunpengn/6220ffc1b69cee5c861d93754e759d08)

Instruções pós compilação/download do dbgen:

Para gerar o esquema e os dados são necessários normalmente dois executáveis, chamados dbgen e qgen:

dbgen = database generator, gera os dados a carregar nas tabelas

qgen = query generator, gera as queries

Para simplificar, não contamos usar o qgen, porque em vez disso temos exemplos das pesquisas em anexo e no link dado que podemos usar. Porém, se quiser pode tentar gerar as pesquisas usando o qgen (que gera queries com valores aleatórios nos filtros), mas para tal teria de compilar o código correspondente.

O dbgen gera dados com o tamanho que quisermos, enquanto o qgen gera pesquisas (queries) com valores gerados aleatoriamente, para que cada vez que uma dada pesquisa corre não ser uma repetição da vez que correu anteriormente. Na prática, e para simplificar, usaremos o dbgen para gerar dados e um conjunto de 22 pesquisas fixas (isto é, com valores fixos) sem correr o qgen.

O primeiro objectivo será então obter o dbgen executável para o seu sistema operativo. Pode usar o código que partilhamos para obter esse executável ou, se tiver problemas com este código no seu sistema, pode procurar outra alternativa no google (e.g. tpc-h dbgen code) (e.g. tpc-h data generator).

Usando o link fornecido:

Se o seu sistema for Windows, tem um dbgen.exe que poderá funcionar logo. Se o seu sistema não for Windows, ou não funcionar, deverá compilar o código fonte. Neste segundo caso, deverá precisar de um compilador de código C, ler o makefile, perceber que "defines" alterar e correr make. Para tal comece por ler o ficheiro README no código fonte.

Uma vez que já tenha o código dbgen gerado, pode a seguir gerar os dados do tamanho que quiser, substituindo o parâmetro "**s"** por um valor no seguinte comando:

Uso: dbgen -s x

Neste comando o valor x é o número de GB a gerar. Acima está indicado quanto usar no projecto. Na aula prática faremos com apenas 10MB ou algo do género, para não demorar muito tempo.

O dbgen gerará 8 arquivos .tbl (dados de tabela). Cada arquivo é um ficheiro de texto cem que cada linha é uma linha de tabela, com valores separados por um delimitador.

De seguida teremos de criar as tabelas e preenchê-las com os dados que gerámos. Esta parte depende do motor de bases de dados que considerarmos.

Postgres – podemos usar o pgAdmin

(Nota: cuidado, verificar limitação de tamanho máximo por bug numa das versões do postgres para windows. Se tiver Windows e acontecer esse problema, terá de voltar instalar outra versão do postgres)

```
CREATE TABLE NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                       N_NAME       CHAR(25) NOT NULL,
                       N_REGIONKEY  INTEGER NOT NULL,
                       N_COMMENT    VARCHAR(152));

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

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 );

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);
```

```
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 );

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);

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);

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);
```

Load data

Nota: penso que havera uma dificuldade com o load de dados no postrges que é devido somente a um | no final de cada linha. Há quatro formas de resolver:

1. adicionar um indicador de line break no comando seguinte que inclui o | (não encontrei, penso que o postgres não tem tal opção, mas o mysql tem);
2. **adicionar uma coluna falsa vazia para fingir que o ultimo pedaço vazio depois do ultimo | é um campo (alter table customer add column lixo varchar(10);); No final após load, pode retirar essa coluna: alter table customer drop column lixo**
3. compilar de novo o código depois de mudar algo para não inclui o | no final da linha.
4. correr um script/programita perl/python ou java ou qq coisa da Shell/cmd line que retira automaticamente o | final de cada linha

Ainda haverá outra dificuldade que é o facto de, por questões de segurança, o postgres não ter permissões de acesso as directorias do computador excepto se soubermos atribuir essas permissões. Para resolver esse problema podemos alterar as permissões no sistema operativo da directoria onde terá os ficheiros com dados ou, se não souber faze-lo, e como solução "suja", mover os ficheiros com os dados para a directoria data dentro da directoria do postgres, já que essa tem permissões porque é onde o postgres tem os seus dados.
Nota: no mac ou Linux, o comando chmod permite-lhe mudar as permissões, e se fizer sudo consegue mesmo mudar qualquer permissão. Em particular, o seguinte comando deve ser o pesadelo de qualquer administrador de sistema preocupado com segurança, mas um sonho para os rstantes.

- **`chmod -R 777 folder`**

load:

```
#!/bin/bash
sed -i '' 's/.$//' *.tbl

COPY customer FROM 'DIR/customer.tbl' WITH DELIMITER AS '|';
COPY lineitem FROM 'DIR/lineitem.tbl' WITH DELIMITER AS '|';
COPY nation FROM 'DIR/nation.tbl' WITH DELIMITER AS '|';
COPY orders FROM 'DIR/orders.tbl' WITH DELIMITER AS '|';
COPY part FROM 'DIR/part.tbl' WITH DELIMITER AS '|';
COPY partsupp FROM 'DIR/partsupp.tbl' WITH DELIMITER AS '|';
COPY region FROM 'DIR/region.tbl' WITH DELIMITER AS '|';
COPY supplier FROM 'DIR/supplier.tbl' WITH DELIMITER AS '|';
```

Create keys

```
ALTER TABLE REGION ADD PRIMARY KEY (R_REGIONKEY);
ALTER TABLE NATION ADD PRIMARY KEY (N_NATIONKEY);
ALTER TABLE CUSTOMER ADD PRIMARY KEY (C_CUSTKEY);
ALTER TABLE SUPPLIER ADD PRIMARY KEY (S_SUPPKEY);
ALTER TABLE PART ADD PRIMARY KEY (P_PARTKEY);
ALTER TABLE PARTSUPP ADD PRIMARY KEY (PS_PARTKEY, PS_SUPPKEY);
ALTER TABLE ORDERS ADD PRIMARY KEY (O_ORDERKEY);
ALTER TABLE LINEITEM ADD PRIMARY KEY (L_ORDERKEY, L_LINENUMBER);
```

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

ALTER TABLE SUPPLIER ADD FOREIGN KEY (S_NATIONKEY) REFERENCES NATION(N_NATIONKEY);
```

```
ALTER TABLE CUSTOMER ADD FOREIGN KEY (C_NATIONKEY) REFERENCES NATION(N_NATIONKEY);

ALTER TABLE PARTSUPP ADD FOREIGN KEY (PS_SUPPKEY) REFERENCES SUPPLIER(S_SUPPKEY);
ALTER TABLE PARTSUPP ADD FOREIGN KEY (PS_PARTKEY) REFERENCES PART(P_PARTKEY);

ALTER TABLE ORDERS ADD FOREIGN KEY (O_CUSTKEY) REFERENCES CUSTOMER(C_CUSTKEY);

ALTER TABLE LINEITEM ADD FOREIGN KEY (L_ORDERKEY) REFERENCES ORDERS(O_ORDERKEY);
ALTER TABLE LINEITEM ADD FOREIGN KEY (L_PARTKEY,L_SUPPKEY) REFERENCES PARTSUPP(PS_PARTKEY,PS_SUPPKEY);
```

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

MySQL

Nota: se mysql pifar, no MAC:

cd /Applications/MAMP/db/mysql/

rm ib_logfile*

http://localhost/phpMyAdmin/

Configuração do MySQL

Se necessário, instale o MySQL. Depois, abra o cliente de acesso ao MySQL e crie uma base de dados para a simulação TCP-H. Agora, crie as tabelas. Para tal tem duas hipóteses: Ou corre os comandos abaixo, ou tira os comandos dos ficheiros dss.ddl (para criação de tabelas) e dss.ri (para criação de chaves).

```
CREATE TABLE NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                       N_NAME       CHAR(25) NOT NULL,
                       N_REGIONKEY  INTEGER NOT NULL,
                       N_COMMENT    VARCHAR(152));

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

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 );

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);

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 );


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);

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);

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);
```

Uma vez criadas as tabelas, pretende-se importar dados. Abra a consola do MySQL e execute os comandos:

```
load data local infile "Yourdirectory/part.tbl" into table part fields
terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/partsupp.tbl" into table partsupp
fields terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/customer.tbl" into table customer
fields terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/nation.tbl" into table nation fields
terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/orders.tbl" into table orders fields
terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/region.tbl" into table region fields
terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/supplier.tbl" into table supplier
fields terminated by "|" lines terminated by "\r\n";

load data local infile "Yourdirectory/lineitem.tbl" into table lineitem
fields terminated by "|" lines terminated by "\r\n
```

O passo seguinte é a criação de chaves, como indicado em dss.ri:

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

ALTER TABLE NATION
ADD PRIMARY KEY (N_NATIONKEY),

ALTER TABLE CUSTOMER
ADD PRIMARY KEY (C_CUSTKEY),

ALTER TABLE SUPPLIER
ADD PRIMARY KEY (S_SUPPKEY);

ALTER TABLE PART
ADD PRIMARY KEY (P_PARTKEY);

ALTER TABLE PARTSUPP
```

```
ADD PRIMARY KEY (PS_PARTKEY,PS_SUPPKEY);

ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);

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

Temos ainda a criação de chaves forasteiras ou estrangeiras. Caso alguma das chaves demore mais de 4 horas a ser criada, pode desistir. As chaves forasteiras não são absolutamente necessárias para o motor poder correr pesquisas.

```
ALTER TABLE NATION
ADD FOREIGN KEY NATION_FK1 (N_REGIONKEY) references REGION (R_REGIONKEY);

ALTER TABLE SUPPLIER
ADD FOREIGN KEY SUPPLIER_FK1 (S_NATIONKEY) references NATION (N_NATIONKEY);

ALTER TABLE CUSTOMER
ADD FOREIGN KEY CUSTOMER_FK1 (C_NATIONKEY) references NATION (N_NATIONKEY);

ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK1 (PS_SUPPKEY) references SUPPLIER (S_SUPPKEY),
ADD FOREIGN KEY PARTSUPP_FK2 (PS_PARTKEY) references PART (P_PARTKEY);

ALTER TABLE ORDERS
ADD FOREIGN KEY ORDERS_FK1 (O_CUSTKEY) references CUSTOMER (C_CUSTKEY);

ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK1 (L_ORDERKEY)  references ORDERS (O_ORDERKEY),
ADD  FOREIGN  KEY  LINEITEM_FK2  (L_PARTKEY,L_SUPPKEY)  references  PARTSUPP
(PS_PARTKEY,PS_SUPPKEY);
```

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

Configuração do MonetDB

Criação das tabelas: usar os create tables acima. Corrigir algum erro que dê.

Copia dos dados:

```
COPY RECORDS INTO region FROM "Yourdirectory/region.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO nation FROM "Yourdirectory/nation.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO part FROM "Yourdirectory/part.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO supplier FROM "Yourdirectory/supplier.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO customer FROM "Yourdirectory/customer.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO partsupp FROM "Yourdirectory/partsupp.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO orders FROM "Yourdirectory/orders.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n';

COPY RECORDS INTO lineitem FROM "Yourdirectory/lineitem.tbl" DELIMITERS
tuple_seperator '|' record_seperator '\r\n'
```

Chaves estrangeiras:

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

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

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

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

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

Project/EFIM/TPC-H default queries**:** Disponiveis no ficheiro TPCHdefaultQueries.txt.

Anexo 1. Mysql scripts

Create tables

```
DROP DATABASE IF EXISTS tpch;
CREATE DATABASE tpch;
use tpch;

CREATE TABLE NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                       N_NAME       CHAR(25) NOT NULL,
                       N_REGIONKEY  INTEGER NOT NULL,
                       N_COMMENT    VARCHAR(152));

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

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 );

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);

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 );

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);
```

```
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);

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);
```

Load data

```
load data local infile "part.tbl" into table part fields terminated by "|" lines terminated by "\n";
load data local infile "partsupp.tbl" into table partsupp fields terminated by "|" lines terminated by "\n";
load data local infile "customer.tbl" into table customer fields terminated by "|" lines terminated by "\n";
load data local infile "nation.tbl" into table nation fields terminated by "|" lines terminated by "\n";
load data local infile "orders.tbl" into table orders fields terminated by "|" lines terminated by "\n";
load data local infile "region.tbl" into table region fields terminated by "|" lines terminated by "\n";
load data local infile "supplier.tbl" into table supplier fields terminated by "|" lines terminated by "\n";
load data local infile "lineitem.tbl" into table lineitem fields terminated by "|" lines terminated by "\n";
```

Create keys

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

ALTER TABLE NATION
ADD PRIMARY KEY (N_NATIONKEY);

ALTER TABLE CUSTOMER
ADD PRIMARY KEY (C_CUSTKEY);

ALTER TABLE SUPPLIER
ADD PRIMARY KEY (S_SUPPKEY);

ALTER TABLE PART
ADD PRIMARY KEY (P_PARTKEY);
```

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

ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);

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


ALTER TABLE NATION
ADD FOREIGN KEY NATION_FK1 (N_REGIONKEY) references REGION (R_REGIONKEY);

ALTER TABLE SUPPLIER
ADD FOREIGN KEY SUPPLIER_FK1 (S_NATIONKEY) references NATION (N_NATIONKEY);

ALTER TABLE CUSTOMER
ADD FOREIGN KEY CUSTOMER_FK1 (C_NATIONKEY) references NATION (N_NATIONKEY);

ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK1 (PS_SUPPKEY) references SUPPLIER (S_SUPPKEY), ADD
FOREIGN KEY PARTSUPP_FK2 (PS_PARTKEY) references PART (P_PARTKEY);

ALTER TABLE ORDERS
ADD FOREIGN KEY ORDERS_FK1 (O_CUSTKEY) references CUSTOMER (C_CUSTKEY);

ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK1 (L_ORDERKEY) references ORDERS (O_ORDERKEY), ADD
FOREIGN KEY LINEITEM_FK2 (L_PARTKEY,L_SUPPKEY) references PARTSUPP
(PS_PARTKEY,PS_SUPPKEY);
```

Anexo 2. Pesquisas

P1.

```
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 '1998-12-01' - interval '108' day
group by
   l_returnflag,
   l_linestatus
order by
   l_returnflag,
   l_linestatus;
```

P2.

```
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 = 30
   and p_type like '%STEEL'
   and s_nationkey = n_nationkey
   and n_regionkey = r_regionkey
   and r_name = 'ASIA'
   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 = 'ASIA'
   )
order by
   s_acctbal desc,
   n_name,
   s_name,
   p_partkey
limit 100;
```

P3.

```
select
        l_orderkey,
        sum(l_extendedprice * (1 - l_discount)) as revenue,
        o_orderdate,
        o_shippriority
from
        customer,
        orders,
        lineitem
where
        c_mktsegment = 'AUTOMOBILE'
        and c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and o_orderdate < date '1995-03-13'
        and l_shipdate > date '1995-03-13'
group by
        l_orderkey,
        o_orderdate,
        o_shippriority
order by
        revenue desc,
        o_orderdate
limit 10;
```

P4.

```
select
        o_orderpriority,
        count(*) as order_count
from
        orders
where
        o_orderdate >= date '1995-01-01'
        and o_orderdate < date '1995-01-01' + 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;
```

P5.

```
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 = 'MIDDLE EAST'
        and o_orderdate >= date '1994-01-01'
        and o_orderdate < date '1994-01-01' + interval '1' year
group by
        n_name
order by
        revenue desc;
```

P6.

```
select
        sum(l_extendedprice * l_discount) as revenue
from
        lineitem
where
        l_shipdate >= date '1994-01-01'
        and l_shipdate < date '1994-01-01' + interval '1' year
        and l_discount between 0.06 - 0.01 and 0.06 + 0.01
        and l_quantity < 24;
```

P7.

```
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 = 'JAPAN' and n2.n_name = 'INDIA')
                                or (n1.n_name = 'INDIA' and n2.n_name = 'JAPAN')
                        )
                        and l_shipdate between date '1995-01-01' and date '1996-12-31'
        ) as shipping
group by
        supp_nation,
        cust_nation,
        l_year
order by
        supp_nation,
        cust_nation,
        l_year;
```

P8.

```
select
        o_year,
        sum(case
                when nation = 'INDIA' 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 = 'ASIA'
			and s_nationkey = n2.n_nationkey
			and o_orderdate between date '1995-01-01' and date '1996-12-31'
			and p_type = 'SMALL PLATED COPPER'
	) as all_nations
group by
	o_year
order by
	o_year;
```

P9.

```
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 '%dim%'
	) as profit
```

```
group by
        nation,
        o_year
order by
        nation,
        o_year desc;
```

P10.

```
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 '1993-08-01'
        and o_orderdate < date '1993-08-01' + 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 20;
```

P11:

```
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 = 'MOZAMBIQUE'
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 = 'MOZAMBIQUE'
                )
order by
        value desc;
```

P12.

```
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 ('RAIL', 'FOB')
        and l_commitdate < l_receiptdate
        and l_shipdate < l_commitdate
        and l_receiptdate >= date '1997-01-01'
        and l_receiptdate < date '1997-01-01' + interval '1' year
group by
        l_shipmode
order by
        l_shipmode;
```

P13.

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266


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

P14.

```
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 '1996-12-01'
        and l_shipdate < date '1996-12-01' + interval '1' month;
```

P15.

```
create view revenue0 (supplier_no, total_revenue) as
        select
                l_suppkey,
                sum(l_extendedprice * (1 - l_discount))
        from
                lineitem
        where
                l_shipdate >= date '1997-07-01'
```

```
                and l_shipdate < date '1997-07-01' + 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;
```

P16.

```
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#34'
        and p_type not like 'LARGE BRUSHED%'
        and p_size in (48, 19, 12, 4, 41, 7, 21, 39)
        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;
```

P17.

```
select
        sum(l_extendedprice) / 7.0 as avg_yearly
from
        lineitem,
        part
where
        p_partkey = l_partkey
        and p_brand = 'Brand#44'
        and p_container = 'WRAP PKG'
        and l_quantity < (
                select
                        0.2 * avg(l_quantity)
                from
                        lineitem
                where
                        l_partkey = p_partkey
        );
```

P18.

```
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) > 314
        )
        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;
```

p19.

```
select
	sum(l_extendedprice* (1 - l_discount)) as revenue
from
	lineitem,
	part
where
	(
		p_partkey = l_partkey
		and p_brand = 'Brand#52'
		and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
		and l_quantity >= 4 and l_quantity <= 4 + 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#11'
		and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
		and l_quantity >= 18 and l_quantity <= 18 + 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#51'
		and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
		and l_quantity >= 29 and l_quantity <= 29 + 10
		and p_size between 1 and 15
		and l_shipmode in ('AIR', 'AIR REG')
		and l_shipinstruct = 'DELIVER IN PERSON'
	);
```

P20.

```
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 'green%'
                        )
                        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 '1993-01-01'
                                        and l_shipdate < date '1993-01-01' + interval '1' year
                        )
        )
        and s_nationkey = n_nationkey
        and n_name = 'ALGERIA'
order by
        s_name;
```

P21.

```
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 = 'EGYPT'
group by
        s_name
order by
        numwait desc,
        s_name
limit 100;
```

P22.

```
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
                                ('20', '40', '22', '30', '39', '42', '21')
                        and c_acctbal > (
                                select
                                        avg(c_acctbal)
                                from
                                        customer
                                where
                                        c_acctbal > 0.00
                                        and substring(c_phone from 1 for 2) in
                                                ('20', '40', '22', '30', '39', '42', '21')
                        )
                        and not exists (
                                select
                                        *
                                from
                                        orders
```

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

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

Anexo 3: How to find tpc-h instructions for other DB engines

Example search phrases on google:

tpch mongodb

https://www.ifi.uzh.ch/dam/jcr:ffffffff-96c1-007c-0000-000010c732ce/VertiefungRutishauser.pdf

https://github.com/tllano11/dss-sql-vs-nosql-experiments

tpch nosql

https://github.com/aiquis/tpch-neo4j
https://neo4j.com/developer/guide-importing-data-and-etl/

Oracle database:
https://infohub.delltechnologies.com/l/design-guide-oracle-big-data-sql-on-dell-emc-powerflex-1/loading-tpch-data-into-the-oracle-database-3

https://github.com/glynnbird/couchimport

https://github.com/VoltDB/voltdb

Loading TPCH data into the Oracle database

•

We followed these steps to load the TPCH data into the Oracle database:
We copied orders and part table CSV files to this location on the Oracle database host:

```
[Oracle@oraclebds data]$ cp orders.tbl*, part.tbl* /tpc/data
```

We created a directory on the Oracle database:

```
SQL>Create OR replace directory tpch_dir AS '/tpch/data';
SQL>grant read on directory tpch_dir to tpch;
```

We created the TPCH tables "orders" and "part" that point to the csv file location using these commands:

```
SQL>create table tpch. orders
(
o_orderkey          NUMBER(10,0),
o_custkey            NUMBER(10,0),
o_orderstatus       CHAR(1),
o_totalprice          NUMBER
o_orderdate          CHAR(10),
o_orderpriority      CHAR(15)
o_clerk                  CHAR(15),
o_shippriority       INTEGER,
o_comment          VARCHAR2(7
            )
 ORGANIZATION EXTERNAL
 (TYPE oracle_loader
  DEFAULT DIRECTORY tpch_dir
```

```
  ACCESS PARAMETERS (
  FIELDS
  TERMINATED BY '|'
   MISSING FIELD VALUES ARE NULL
)
LOCATION('/tpch/data/orders*.tbl'));
```

Similarly, we created a TPCH table "part" that points to the location:

```
SQL> create table tpch.ext_part
(
P_partkey               NUMBER(10,0),
P_name                  VARCHAR2(55),
P_mfgr                    CHAR(25),
P_brand                  CHAR(10),
P_type                    VARCHAR2(25),
P_size                      INTEGER,
P_container           CHAR(10),
P_retailprice          NUMBER,
P_comment          VARCHAR2(23)
)
ORGANIZATION EXTERNAL
(TYPE  oracle_loader
DEFAULT DIRECTORY tpch_dir
ACCESS PARAMETERS (
FIELDS
TERMINATED BY '|'
MISSING FIELD VALUES ARE NULL
)
LOCATION('/tpch/data/part.tbl*));
```

Extra random collected info

Extra 1

To facilitate testing, I need some data that lends itself easily for partition, which led me to TPC-H's dbgen tool.
dbgen is pretty easy to build and compile on Linux. It also has a Windows version. I didn't bother trying because getting it built on Linux was such a quick and easy process. However, the documentation of dbgen, if it can be called as such, leaves a lot to be desired. But by experimenting and surfing around, I was able to get what I needed.

Instructions here are for CentOS. Building it on other flavors of Linux should also be easy.

1. yum install make and yum install gcc if you don't have that already;

2. wget source code;

3. tar xf tarball;

4. cd dbgen

5. vi makefile.suite and change 4 lines. See uncommented lines below:

```
################
## CHANGE NAME OF ANSI COMPILER HERE
################
CC      = gcc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata)
#                                  SQLSERVER, SYBASE, ORACLE
# Current values for MACHINE are:  ATT, DOS, HP, IBM, ICL, MVS,
#                                  SGI, SUN, U2200, VMS, LINUX, WIN32
```

```
# Current values for WORKLOAD are:  TPCH
DATABASE= SQLSERVER
MACHINE = LINUX
WORKLOAD = TPCH
```

Note: It does not have MySQL and others for DATABASE setting, but that really doesn't matter, because the file generated, which is delimited by a pipe, |, can be used for loading into any system.

6. make -f makefile.suite
This creates the dbgen binary that we can use in the next step.

7. ./dbgen -h gives you a brief description of switches that can be used. I am just interested in creating the lineitem table, because it has a good combination of integer, decimal, date, and character fields. The date inside the tables spans between 1992-01-01 and 1998-12-31, which is good for a partitioned table based on date. The DDL for this table is inside the dss.ddl file in the same directory. Please modify it as necessary. In fact, Lubor Kollar has it ready for SQL Server here, and Vadim Tkachenko has it for MySQL here.
Here is the command I used and its results:

```
 [root@centos dbgen]# ./dbgen -v -T L -s 2
TPC-H Population Generator (Version 2.14.0)
Copyright Transaction Processing Performance Council 1994 - 2010
Generating data for lineitem table/
Preloading text ... 100%
Do you want to overwrite ./lineitem.tbl ? [Y/N]: Y
done.
```

-v: verbose, -T L: lineitem only, -s 2: scale factor of 2. My understanding is that it roughly indicates that the end file will be close to 2 gig.

It turns out the end file is about 1.5 gig, with close to 12 million rows in it.

```
-rw-r--r-- 1 root root  1.5G Mar 30 21:33 lineitem.tbl
[root@centos dbgen]# wc -l lineitem.tbl
11997996 lineitem.tbl
```

You can play around with different key switches to get the size of file you want. For example, quoting from Lubor Kollar:
dbgen –T L -s 4 -C 3 -S 1

Using the –s option, we set the scale to 4 that generates a Lineitem table of 3GB. Using the –C option we split the table into 3 portions, and then using the –S option we chose only the first 1GB portion of the Lineitem table.
Running the command above, I got a file named lineitem.tbl.1 with close to 8 million rows, about 976 meg.

When trying to load it into SQL Server with SSIS bulk copying task, remember the delimiter is | and line separator is LF (line feed). For MySQL, something like this should do:

```
load data local infile '/root/dbgen/lineitem.tbl.1' into table lineitem
fields terminated by '|' lines terminated by '\n';
```

Extra 2

The data generated by **Load Data** directly load will report an error. The reason is that the data generated by dbgen has a separator at the end of each line, which causes the PG to be unable to correctly determine the number of fields, and import errors.

```
postgres=# copy part from '/home/zhuqingping/Study/BenchMark/TPCH/tpch_2_17_0/dbgen/part.tbl' with (delimiter '|' , null '' ); ERROR: extra data after last expected column CONTEXT: COPY part, line 1 : "1|goldenrod lavender spring chocolate lace|Manufacturer#1|Brand#13|PROMO BURNISHED COPPER|7|JUMBO PK..."
```

To solve this problem, the data needs to be modified to remove the separator at the end of each line. The C language implementation is as follows:

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>

int main( int argc, char *argv[])
{
   if (argc < 2 )
       printf ( "wrong use, %d\n" , argc);
   FILE *fp1 = fopen(argv[ 1 ], "r" );
   FILE *fp2 = fopen(argv[ 2 ], "w" );
    char str_data[ 1000 ];
    int len = 0 ;
    while ((fgets(str_data, 1000 , fp1) != NULL) && strlen (str_data)> 2 )
   {
      len = strlen (str_data);
      str_data[len- 2 ] = '\0' ;
       fprintf (fp2, "%s\n" , str_data);
   }
   fclose(fp1);
   fclose(fp2);
   return 0 ;
}
```

Extra 3

dbgen puts a SEPARATOR symbol "|" at the end of each line, but I want to get rid of it, keeping only "|" 's between two columns. In other words, the original data format is:
col1|col2|col3|col4|

What I want is:
col1|col2|col3|col4

This may simplify some of my code which uses a "split"-like function to extract data lines.
So, first step, modify "dss.h".

Locate the following line:

#define PR_END(fp) fprintf(fp, "\n")
change it to:

#define PR_END(fp) {fseek(fp, -1, SEEK_CUR);fprintf(fp, "\n");}
run "make".

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

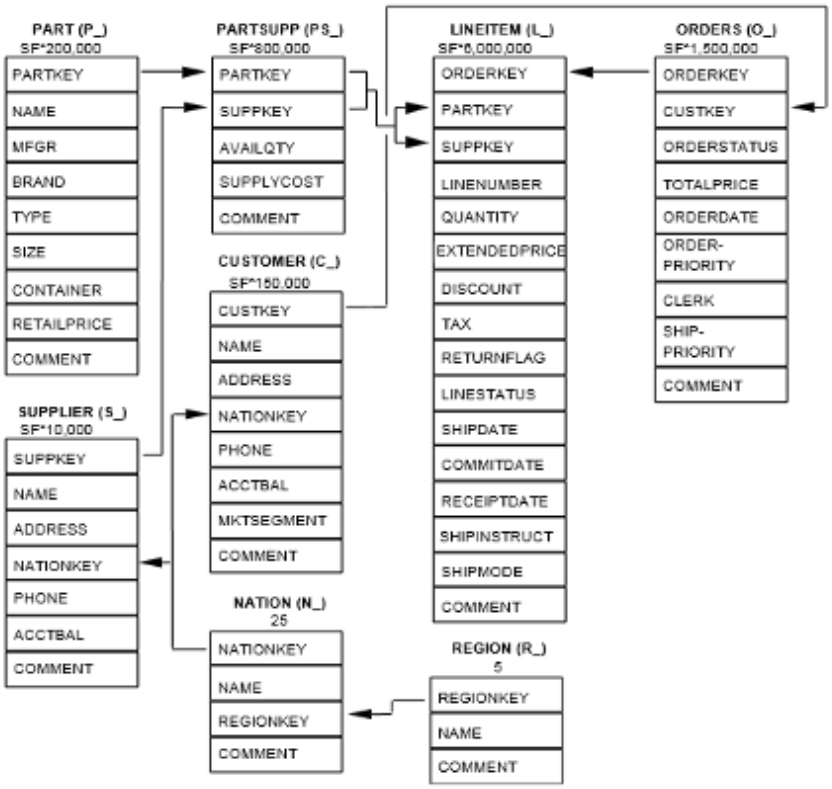
Anexo 4. Install and run from github

https://docs.deistercloud.com/content/Databases.30/TPCH%20Benchmark.90/Data%20generation%20tool.30.xml?embedded=true

TPCH database and dbgen data generation utility, courtesy of www.tpc.org, were developed to provide an approach to benchmarking and include:

- The `tpch` Database structure
- The `tpch dbgen` utility, a utility to populate the database with a specified amount of data (Scale Factor)
- The `tpch` benchmark queries, a set of pre-defined data warehouse queries to run against the database

We will show the details of the creation of the tpch database and it's population using the `dbgen` utility to generate data.



Legend:

- The parentheses following each table name contain the prefix of the column names for that table;
- The arrows point in the direction of the one-to-many relationships between tables;
- The number\formula below each table name represents the cardinality (number of rows) of the table. Some are factored by SF, the Scale Factor, to obtain the chosen database size. The cardinality for the LINEITEM table is approximate (see Clause 4.2.5).

In essence, the schema consists of 8 tables, 8 explicit unique indexes supporting 8 primary keys and 9 explicit indexes supporting 9 foreign keys.

1 Download dbgen

The `tpch dbgen` utility generates, by default, a set of flat files suitable for loading into the `tpch` schema with the size based on the "Scale Factor" argument. A scale factor of 1 produces a complete data set of approximately 1 GB, a scale factor of 10 produces a data set of approximately 10 GB etc.

Download the `dbgen` source code:

Copy

```
$ git clone https://github.com/electrum/tpch-dbgen.git

Cloning into 'tpch-dbgen'...

remote: Counting objects: 149, done.

remote: Total 149 (delta 0), reused 0 (delta 0), pack-reused 149

Receiving objects: 100% (149/149), 216.15 KiB | 202.00 KiB/s, done.

Resolving deltas: 100% (30/30), done.

Checking connectivity... done.
```

You need to have `git` and `gcc` compiler installed on your machine.

2 Compile dbgen

In the downloaded directory (tpch-dbgen), edit the file `makefile.suite` and set the following variables to the appropriate vaules:

Copy

```
CC=gcc

DATABASE=INFORMIX

MACHINE=LINUX

WORKLOAD=TPCH
```

The run the make utility:

Copy

```
$ make -f makefile.suite

gcc -g -DDBNAME=\"dss\" -DMAC -DINFORMIX -DTPCH -DRNG_TEST -D_FILE_OFFSET_BITS=64
-c -o build.o build.c

gcc -g -DDBNAME=\"dss\" -DMAC -DINFORMIX -DTPCH -DRNG_TEST -D_FILE_OFFSET_BITS=64
-c -o driver.o driver.c
```

```
...
```

```
Macs-MBP-4:tpc-h pedro$ git clone https://github.com/electrum/tpch-
dbgen.git
Cloning into 'tpch-dbgen'...
remote: Enumerating objects: 149, done.
remote: Total 149 (delta 0), reused 0 (delta 0), pack-reused 149
Receiving objects: 100% (149/149), 214.31 KiB | 881.00 KiB/s, done.
Resolving deltas: 100% (34/34), done.
Macs-MBP-4:tpc-h pedro$ ls
tpch-dbgen
Macs-MBP-4:tpc-h pedro$ cd tpch-dbgen
Macs-MBP-4:tpch-dbgen pedro$ ls
```

```
4    -c -o bm_utils.o bm_utils.c
bm_utils.c:71:10: fatal error: 'malloc.h' file not found
#include <malloc.h>
         ^~~~~~~~~~
1 error generated.
```

Found solution in https://github.com/pola-rs/tpch/blob/main/README.md:

Notes:

- For MacOS, the above `make` command will result in an error while compiling like below,
- `bm_utils.c:71:10: fatal error: 'malloc.h' file not found`
- `#include <malloc.h>`
- `         ^~~~~~~~~~`
- `1 error generated.`
  `make: *** [bm_utils.o] Error 1`
  To fix this, change the import statement `#include <malloc.h>` to `#include <sys/malloc.h>` in the files where error is reported (`bm_utils.c` and `varsub.c`) and then re-run the command `make`.

Agora já deu.

3 Test dbgen

Now you are ready to generate the `tpch` files

- Change to the appropriate directory where you want to generate `tpch` files. For example, create a subdirectory under the `tpch-dbgen` directory.

```
$ mkdir data

$ cd data
```

- Copy the `dbgen` executable file and `dists.dss` file there.

```
$ cp ../dbgen .

$ cp ../dists.dss .
```

- Run `dbgen` for the appropriate database size factor (1GB in the sample).

```
./dbgen -s 1
```

- Generation may take a while. When completed, you can see the resulting files.

```
$ ls -l

total 2150000

-rw-r--r--  1 deister  staff   24346144 13 may 12:05 customer.tbl

-rw-r--r--  1 deister  staff  759863287 13 may 12:05 lineitem.tbl

-rw-r--r--  1 deister  staff       2224 13 may 12:05 nation.tbl

-rw-r--r--  1 deister  staff  171952161 13 may 12:05 orders.tbl

-rw-r--r--  1 deister  staff   24135125 13 may 12:05 part.tbl

-rw-r--r--  1 deister  staff  118984616 13 may 12:05 partsupp.tbl

-rw-r--r--  1 deister  staff        389 13 may 12:05 region.tbl

-rw-r--r--  1 deister  staff    1409184 13 may 12:05 supplier.tbl
```

UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

Anexo 5.

https://gist.github.com/yunpengn/6220ffc1b69cee5c861d93754e759d08

How to generate dataset using TPC-H

This is a step-by-step guide on how to generate dataset using TPC-H.

- Download the database generation tool from here.
- To generate TPC-H compliant datasets, we must use the `dbgen` tool.
- Compile the `dbgen` tool by `make -f makefile.suite`.
    - Remember to modify `makefile.suite`.

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

- Use the `dbgen` tool with the following options:
    - For example, you can use `./dbgen -s 1 -v`

```
option  argument    default     action
------  --------    -------     ------
-h                              Display a usage summary
-f      none                    Force. Existing data files will be overwritten.
-F      none        yes         Flat file output.
-D       none                     Direct database load. ld_XXXX() routines must be defined in
load_stub.c
-s      <scale>     1          Scale of the database population. Scale 1.0 represents ~1 GB of
data
-T      <table>                 Generate the data for a particular table ONLY. Arguments:
                                p -- part/partuspp,
                                c -- customer, s -- supplier,
                                o -- orders/lineitem, n -- nation, r -- region,
                                l -- code (same as n and r),
                                O -- orders, L -- lineitem, P -- part,
                                S -- partsupp
-O      d                       Generate SQL for delete function instead of key ranges
-O      f                       Allow over-ride of default output file names
-O      h                        Generate headers in flat ascii files. hd_XXX routines must be
defined in load_stub.c
-O      m                       Flat files generate fixed length records
-O      r                       Generate key ranges for the UF2 update function
-O      v                       Verify data set without generating it.
-r       <percentage>     10      Scale each udpate file to the given percentage (expressed in
basis points) of the data set
-v      none                    Verbose. Progress messages are displayed as data is generated.
-n      <name>                  Use database <name> for in-line load
-C      <children>              Use <children> separate processes to generate data
-S      <n>                     Generate the <n>th part of a multi-part load or update set
-U       <updates>               Create a specified number of data sets in flat files for the
update/delete functions
-i      <n>                     Split the inserted rows in an refresh pair between <n> files
```

```
-d      <n>                     Split the deleted rows in an refresh pair between <n> files
```

- Now, you should see a few XXX`.tbl` files. However, there is a bug in `dbgen` which generates an extra | at the end of each line. To fix it, run the following command:

```
for i in `ls *.tbl`; do sed 's/|$//' $i > ${i/tbl/csv}; echo $i; done;
```

- Assume you have a PostgreSQL instance. Now load the data into it:

```
# Creates the schema.
psql -c "DROP DATABASE IF EXISTS tpc"
psql -c "CREATE DATABASE tpc"
psql -d tpc -f dss.ddl

# Adds primary keys & foreign keys.
psql -d tpc -f dss.ri

# Loads data.
psql -d tpc -c "COPY region FROM 'region.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY nation FROM 'nation.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY customer FROM 'customer.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY supplier FROM 'supplier.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY part FROM 'part.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY partsupp FROM 'partsupp.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY orders FROM 'orders.csv' WITH (FORMAT csv, DELIMITER '|')";
psql -d tpc -c "COPY lineitem FROM 'lineitem.csv' WITH (FORMAT csv, DELIMITER '|')";
```