Universiteit Leiden

CWI
Database Architectures

ADM 2022

Assignment 1

The TPC-H ad-hoc, decision support benchmark.

Universiteit Leiden

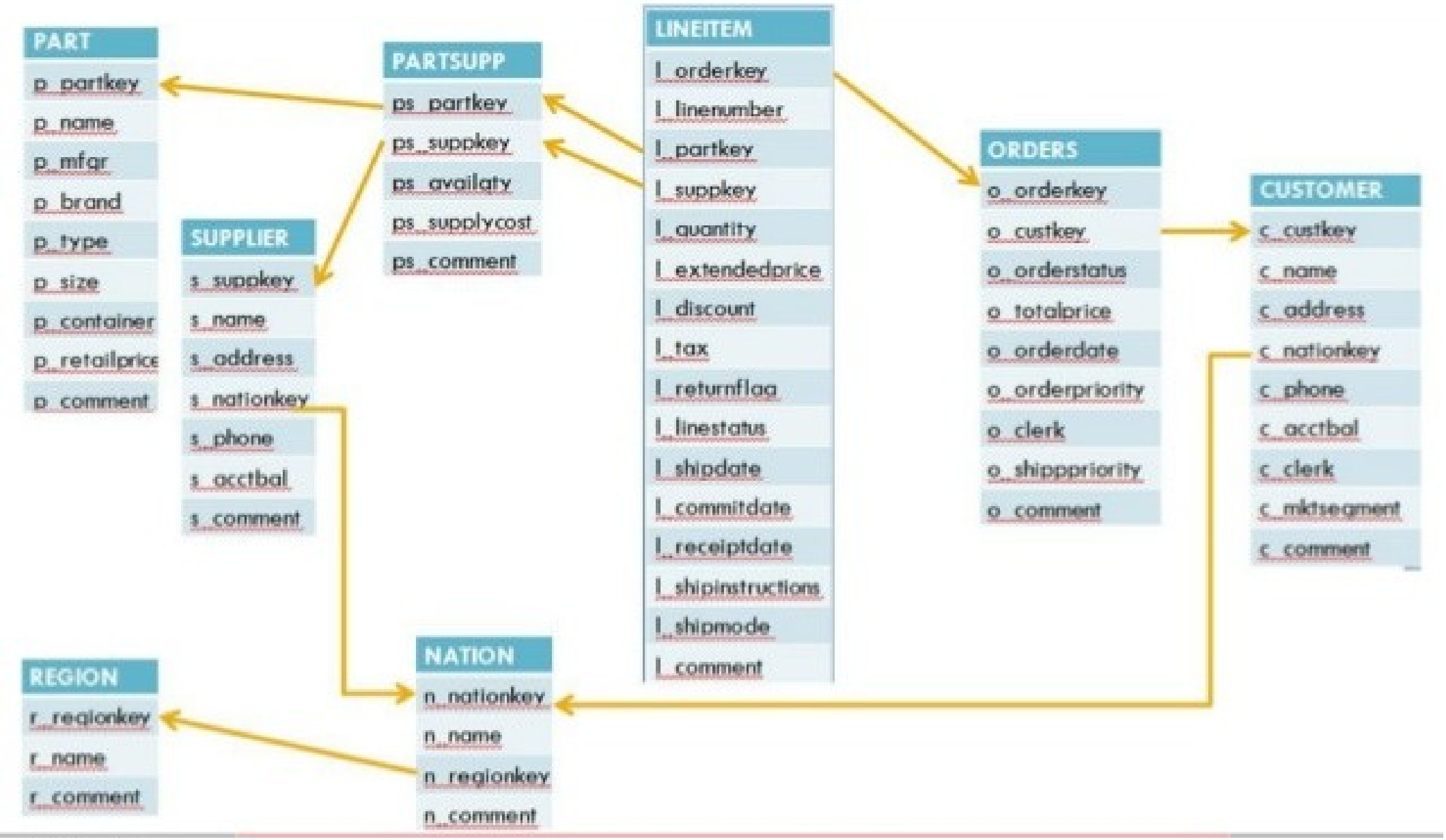
CWI
Database Architectures

TPC-H
ad-hoc, decision support benchmark

- (still) THE standard database OLAP benchmark
- By independent TPC organization
- All major DB vendors are members
- Official audited results (available online)

http://www.tpc.org/

TPC-H Database Schema



Universiteit Leiden

CWI
Database Architectures

TPC-H
ad-hoc, decision support benchmark

- Synthetic data
- Database generator “dbgen”
- Variable database size:
  - Scale factor “SF”: 1, 3, 10, 30, 100, 300, …
  - SF-1 ~= 1 GB

- 22 query templates
  - Query generator “qgen” to instantiate literals

http://www.tpc.org/

Universiteit Leiden

CWI
Database Architectures

TPC-H
ad-hoc, decision support benchmark

- “modes”
  - Single-client “power” (query time) test
  - Multi-client concurrent query throughput test

- Official runs also include *updates*
  - *Ignored / omitted here*

- Various metrics, also including price of system
  - Details online
  - ***Here: single-client query performance***

http://www.tpc.org/

Assignment 1

- **Provided:**
  - TPC-H sources: https://homepages.cwi.nl/~manegold/tpc-h_v3.0.1.zip
  - In there,
    - the data- & query-generator "dbgen" & "qgen" are in …/dbgen/
    - Data for SF-1 & SF-3 are in …/dbgen/SF-{1,3}/data.zip → *unzip!*
    - Sample query results are in …/dbgen/SF-1/results/
      - In two formats: computer-readable .cvs and human-readable .pretty
    - SQL schema creation and data loading scripts for MonetDB are in …/dbgen/MonetDB/
      - Might also work for other DBMSs, possibly requiring minor syntax changes
    - Queries for MonetDB and SF-1 are provided in …/dbgen/MonetDB/
      - If you want to run the queries on other scale factor(s) than SF-1, you need to edit query 11 ("q11.sql") as explained by the comment in "q11.sql"
      - Might also work for other DBMSs, possibly requiring minor syntax changes

Assignment 1

- **Optional:**
  - Build the TPC-H data- and query-generator “dbgen” & “qgen” yourself:
  - Sources are in `https://homepages.cwi.nl/~manegold/tpc-h_v3.0.1.zip`
    - Go to `…/dbgen/`
    - On Linux (and alike) build via `` `make -f Makefile.MonetDB` ``
      - Edit “`Makefile.MonetDB`” or “`makefile.suite`” accordingly for other systems
  - Generate the data:
    - In `…/dbgen/` call `` `./dbgen.sh 1` ``
      - This generates the data in `…/dbgen/SF-1/data/`
      - Change “`1`” to other number for other scale factors
  - Generate the queries:
    - In `…/dbgen/` call `` `./qgen.sh 1` ``
      - This generates the queries in `…/dbgen/SF-1/queries/`
      - Change “1” to other number for other scale factor
  - `SF-10 (~10 GB):` `https://homepages.cwi.nl/~manegold/SF-10.zip`
  - `SF-30 (~30 GB):` `https://homepages.cwi.nl/~manegold/SF-30.zip`

http://www.tpc.org/

Assignment 1

- **Tasks 1/2 (60%):**
  - Install MonetDB and one other DBMS of your choice
    - MonetDB: `https://www.monetdb.org/`
  - With both systems, for at least scale factors SF-1 & SF-3 (using the provided scripts and data):
    - Create TPC-H schema (`.../dbgen/MonetDB/0-create_tables.sql`)
    - Load TPC-H data (`.../dbgen/MonetDB/1-load_data.SF-*.sql`)
    - (create constraints: primary- & foreign-keys) (`.../dbgen/MonetDB/2-add_constraints.sql`)
    - Run TPC-H queries (`.../dbgen/MonetDB/q??.sql`)
  - **Verify (for SF-1 data & default query values) that results are correct**
  - Document in detail how and on what system you run:
    - hardware, OS, DBMS, version numbers, configuration parameters, tuning parameters, etc.
    - Make sure that your documentation is sufficient for a third person to repeat your experiments and yield the same results
  - Compare query execution times between multiple runs of the same DBMS *and* between DBMSs
    - Graphically visualize times and differences
  - Describe – in your own words – what performances differences, if any, you see (per query) between runs and between DBMSs

http://www.tpc.org/

Assignment 1

- **Tasks 2/2 (40%):**
  - Implement queries Q1 & Q6 in a programming-, scripting-, statistical-, data analysis language (or system) of your choice (C, C++, Java, Python, R, …)
    (hint: start with Q6, i.e., the simpler one of the two)
  - **Verify (for SF-1 data) the correctness of the results produced by your implementation!**
  - Compare execution times of your implementation (for scale factors SF-1 & SF-3) to those of the DBMSs
    - If your implementation is single-threaded, you might want to compare to the DBMSs running both single- and multi-threaded (where applicable)
  - Describe – in your own words – why the performance of your own implementation compares to those of the DBMS(s)

- *Bonus points will be awarded for*
  - *each scale factor you use larger than SF-3 (with DBMSs and/or your own implementation)*
  - *using more than one (significantly different) hardware platform or more than TWO DBMSs (and discussing their effect on the observed performance)*
  - *providing own implementations for Q1 & Q6 that "in fair comparison" are faster than MonetDB*

http://www.tpc.org/

Assignment 1

TPC-H Q6:

```
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;
```

http://www.tpc.org/

Assignment 1

TPC-H 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 (3)
group by  l_returnflag, l_linestatus
order by  l_returnflag, l_linestatus;
```

http://www.tpc.org/

Assignment 1

- **Produce:**
  - A report (in **PDF**) that describes:
    - How you run the benchmark (such that the reader could repeat your experiments)
    - How you implemented Q1 & Q6
    - How you verified that SF-1 results are correct (for DBMSs and own code)
    - The execution times you got (graphs and/or tables)
    - Your discussion of the performance results/differences
  - A compressed archive (e.g., zip) containing:
    - Your above report (named “report.pdf”)
    - The scripts / programs you created and used
    - Your own implementation of Q1 & Q6
    - Query results & execution times achieved (with SF-1 & SF-3)
- Work in **groups of (max.) 6 students** and name your submission file (archive)
  ADM2022-A1-<student-ID-1>-…-<student-ID-n>-archive.zip
  (with student IDs sorted in ascending order!)
- Submit via BrightSpace
- ***Deadline: Tuesday October 04, 2022, 09:00 CEST***