Homework 8

Problem 8.1

a)

Fragmentation is needed if data becomes too large to fit on a single node. The data is then split up into several parts. This also allows a query to run in parallel on different nodes at a time, which potentially speeds up the processing. Replication is used to achieve load balancing and fault tolerance. Having the same fragment on multiple nodes can help to reduce „hotspots“. However, it is very hard to predict how the fragments should be distributed across the landscape.

b)

When using horizontal fragmentation, one will get a set of tuples (full records) in each fragment, when the data is split vertical, one will retrieve a set of attributes („columns“).

In the case of a horizontal split the data can be reconstructed using the UNION operator, for vertical fragments this is achieved by joining the individual fragments using the row identifiers.

Example horizontal: A company has three invoicing departments, each department takes care of invoices for a different type of customers. Dep1 deals with big corporations, which range from CID 0 – 9999 , dep2 has small business from CID 10000 – 19999 and dep3 private customers for CIDs 20000 – 29999. With horizontal Fragmentation the data can be seperated in three partitions and each department only needs to access a single node.

Example vertical: Two other departments, demand managment and shipping, work with order data. However, the demand management needs access to information like price, etc. and shipping to shipping location, etc. Vertical Fragmentation helps to seperate these attributes onto different nodes, now each deparment only needs to access „their“ nodes.

c)

The criteria are Completeness, Reconstruction and Disjointness.

Possible Problems for (R $\widehat{=}$ some example Relation)

- Completeness:
  - Fragment 1: $\sigma_{Year>2001}(R)$
  - Fragment 2: $\sigma_{Year<1995}(R)$
- Reconstruction:
  - Fragment 1: $\sigma_{Year>2001}(R)$
  - Fragment 2: $\sigma_{Year<1995}(R)$
  - If tuples are missing, the relation cannot be reconstructed.
- Disjointness:
  - Fragment 1: $\sigma_{Year>2005}(R)$
  - Fragment 2: $\sigma_{Year<2006}(R)$

d)

- **Round-Robin**
  - *Adv:* Data is distributed evenly across the system.
  - *Disadv:* Pruning is not possible.
- **Range-Partitioning**
  - *Adv:* Pruning of even range queries is possible.
  - *Disadv:* Data may not be distributed evenly $\longrightarrow$ data skew
- **Hash-Partitioning**
  - *Adv:* Depending on the hash function, the data is distributed evenly. Pruning of single values is possible.
  - *Disadv:* Pruning of range queries is not possible.

e)

To support co-partitioning tables have to be partitioned on the same key (the join key). For the example we have to use customer.custkey in table customer and calls.custkey in table calls. This way ensures that all calls for one customer are located in one partition, this enables pruning. For two fragments the hash function $h(k) = k \ mod \ 2$ on the custkeys could be used. Query plan:

$$\boldsymbol{\pi}_{call.amount,call.month,cust.firstname,cust.lastname}(\\ \boldsymbol{\sigma}_{cust.custkey=1 \ \boldsymbol{AND} \ call.month=12/2017}(cust_1 \bowtie calls_1))$$

We assume just one possible plan needs to be given, not the perfect execution plan (The DMBS optimizes the query.).

f)

Allocation is the process of assigning (replicated) fragments to different nodes (How shall the fragments be distributed across the system?). This is achieved by an allocation algorithm, which tries to minimize the runtime and maximizes throughput With an optimal distribution, which is hard to achieve, because it is generally not known to the system how the data will be queried in future ($\longrightarrow$ heuristics).

Problem 8.2

a)

Naming Convention: Relation `Order` is `O`, `shipping` is `ship` and `invoice` is `inv`

- Fragments for Darmstadt:
  - Shipping department
    $F_1 := \pi_{O.order\_number, O.article, O.customer\_name, O.quantity, O.ship\_zip, O.ship\_city, O.ship\_street}$
    $(\sigma_{O.ship\_zip \geq 64283\ AND\ O.ship\_zip \leq 64297}(O))$
  - Invoicing department
    $F_2 := \pi_{O.order\_number, O.total\_price, O.inv\_zip, O.inv\_city, O.inv\_street}(O)$
- Fragments for Weiterstadt:
  $F_3 := \pi_{O.order\_number, O.article, O.customer\_name, O.quantity, O.ship\_zip, O.ship\_city, O.ship\_street}($
  $\sigma_{O.ship\_zip < 64283\ OR\ O.ship\_zip > 64297}(O))$

b) Reconstruction:

Assuming we can use a natural Join:

$(F_1 \cup F_3) \bowtie F_2$
SELECT * FROM (F_1 UNION ALL F_3) x NATURAL JOIN F_2;

else:

$(F_1 \cup F_3) \bowtie_{(F_1 \cup F_3).order\_number = F_2.order\_number} (F_2)$
SELECT * FROM (F_1 UNION ALL F_3) x INNER JOIN F_2
ON x.order_number = F_2.order _number;

c) First of all, the zip code has to be updated within the tuple. Further, the wrong record might have been saved in a different fragment and then has to be transferred to the correct one. Alternatively, the zip code was in the range of 64283 – 64297, in this cas the update step suffices.

Problem 8.3

We assume the fragemented table is Employee not Employer (which is is not known).

a) SELECT * FROM Employee;
b) SELECT * FROM EmployeeOther UNION EmployeeWE;
c) SELECT * FROM EmployeeOther AT N_1 UNION EmployeeWE AT N_2;