Group **A**

Please fill in your name and registration number (Matrikelnr.) **immediately**.

Exam 3 ON	**SOLUTION KEY**	09.10.2019
	Advanced Database Systems (**184.780**)	**GROUP A**
Matrikelnr.	Last Name	First Name

Duration: 75 minutes. Provide the solutions on the designated pages. **Good Luck!**

Question 1: (4)

For each of the statements below, decide if it is true or false and tick the corresponding circle. You get +1 credit for each correct answer, −1 credit for each wrong answer and 0 credit if you leave the answer open. In total, you always get ≥ 0 credits on the entire exam question 1.

1. In a distributed DBMS, "execution transparency" means that the distributed DBMS executes the queries issued by the users on a first-come-first-served basis (by this we mean that there are no priorities among users; the queries of all users have the same chance to be executed next.) true ◯ false ⊗
2. Suppose that, in a distributed DBMS using the 2PC-protocol, some subordinate receives a commit message (for some transaction T) from the coordinator, writes the commit log record to disk and then crashes. In this case, it is guaranteed that during recovery of this node, the transaction T will be completed. true ◯ false ⊗
3. Suppose that a cluster applies scalar clocks and consider two events of the same process p: event e_q is receiving some message M_q from another process q and event e_r is sending some message M_r to another process r with $q \neq r$. Now suppose that $C(e_q) < C(e_r)$ holds, where $C(\cdot)$ denotes the global clock of events. Then it is guaranteed that M_q was sent before M_r and that M_q is received before M_r. true ⊗ false ◯
4. In a column store, it may happen (depending on the concrete data) that bit vector encoding requires more space than dictionary encoding and the other way around. true ⊗ false ◯

Question 2: (4)

The university wants to support students by establishing a mentoring system, i.e.: each student is assigned a professor as her/his mentor. Assume that students are identified by their Matrikelnummer (matrnr) and professors are identified by their personnel id (pid). The information on students and professors has been stored in a document store in the following way:

```
db.students.insert([
{ _id: "S1",
matrnr: "123456",
fname: "Alice",
lname: "Williams",
studies: "937",
mentor: "1234" },

{ _id: "S2",
matrnr: "135678",
fname: "Bob",
lname: "Higgins",
studies: "645",
mentor: "5678" },
```

```
% db.students.insert continued
{ _id: "S3",
matrnr: "144567",
fname: "Carol",
lname: "Hawkins",
mentor: "5678" },

{ _id: "S4",
matrnr: "153456",
fname: "Dave",
lname: "Stevens",
mentor: "1234" }
]);
```

```
db.professors.insert([
{ _id: "P1",
pid: "1234",
fname: "Franz",
lname: "Maier" },

{ _id: "P2",
pid: "5678",
fname: "Helga",
lname: "Huber" },

{ _id: "P3",
pid: "3579",
lname: "Mueller",
area: "logic",
institute: 192 }
]);
```

(a) Suppose that the most time-critical kind of queries asks for information on the professors including the list of students that each professor mentors. How would you create the professors collection in this case? [1 credit]

```
db.professors.insert([
{ _id:  "P1",
pid:  "1234",
fname:  "Franz",
lname:  "Maier",
students:  ["123456", "153456"] },

{ _id:  "P2",
pid:  "5678",
fname:  "Helga",
lname:  "Huber",
students:  [ "135678", "144567" ] },

{ _id:  "P3",
pid:  "3579",
lname:  "Mueller",
area:  "logic",
institute:  192 }
]);
```

(b) Give a representation of the entire document store as a relational database: provide both the schema and the actual tables. Make sure that the schema is in 3NF to avoid redundancies. Show the schema in the form Table (attr1, attr2, attr3, ...), where "Table" refers to a relation schema with attributes "attr1", "attr2", "attr3", ..., such that "attr1" is the primary key. [3 credits]

Schema:
Students (matrnr, fname, lname, studies, mentor) and
Professors (pid, fname, lname, area, institute).

The tables have the following content:

Students				
matrnr	fname	lname	studies	mentor
"123456"	"Alice"	"Williams"	"937"	"1234"
"135678"	"Bob"	"Higgins"	"645"	"5678"
"144567"	"Carol"	"Hawkins"	NULL	"5678"
"153456"	"Dave"	"Stevens"	NULL	"1234"

Professors				
pid	fname	lname	area	institute
"1234"	"Franz"	"Maier"	NULL	NULL
"5678"	"Helga"	"Huber"	NULL	NULL
"3579"	NULL	"Mueller"	"logic"	192

Question 3: (4)

In this question you are asked to compute the communication cost (number of bytes transferred) for the given query and three given join strategies. **Project as early as possible in all cases** to achieve the minimal communication cost. `Room.bid` is a foreign key to `Building`, i.e., **the join has a selectivity of** $\frac{1}{30}$.

$$\pi_{capacity,addr}(Building \bowtie_{id=bid} \pi_{bid,capacity,name}(Room)) \tag{1}$$

Assume a distributed database with 3 sites and the relations `Flight` and `Aircraft`. See the following tables for details of the scenario. You can assume that the relations are *non-fragmented* and all records (and attributes) are *fixed-size*.

Relation	Site	# Records	Record Size (byte)
Room	1	4 500	100
Building	2	30	190

Relation	Attribute	Size
Room	bid	15
Room	capacity	5
Room	name	20
Room	info	60
Building	id	15
Building	name	80
Building	addr	95

(a) Compute the communication cost if the join is computed at site 3. [1 credits]

`We use MB/kB as shorthand for` 10^6 `and` 10^3 `bytes respectively.`

$$Size_{Room} = 4\,500 \cdot 20byte = 90kB$$
$$Size_{Building} = 30 \cdot 110byte = 3,3kB$$
$$Total = Size_{Room} + Size_{Building} = 93,3kB$$

(b) Compute the communication cost if the join is computed at site 2 and the result then transferred to site 3. [1 credits]

$$Size_{Join} = 4\,500 \cdot 100byte = 450kB$$
$$Total = Size_{Join} + Size_{Room} = 540kB$$

(c) Compute the communication cost if the join is computed by first computing $Room \ltimes Building$ at site 1, then using the result to compute the join at site 2 and finally transferring the result to site 3. **For the selectivity of the semijoin, recall that Room.bid is a foreign key to Building.** (Don't forget about projecting away unnecessary fields.)

[2 credits]

$$Size_{Building.id} = 30 \cdot 15byte = 450byte$$
$$Size_{Semijoin} = 4\,500 \cdot 20byte = 90kB$$
$$Total = Size_{Building.id} + Size_{Semijoin} + Size_{Join} = 540,45kB$$

Note that because Room.bid is a foreign key to Building, the semijoin will not remove any rows.

Question 4: (4)

You are leading a large-scale data analytics project for a grocery shop. You have one table, containing a integer product id and a 16 character string containing a card number with which the product was paid. Whenever a customer pays with card, one row for every product they bought will be appended to the table. Your most important query is to compute the number of products bought in total with each card number.

Discuss how compression in column stores could be used to improve performance on this query. What types of compression make sense in this scenario and how do they affect the evaluation of the aggregation required by this query.

Dictionary encoding and run-length encoding on the card number column both are likely to provide significant benefit.

- For run-length encoding, observe that it is safe to assume that customers buy multiple products at once. Therefore the same card number would be repeated multiple times (once for every bought product). We can then aggregate without decompressing by just summing up the occurance counts for each card number.
- For dictionary encoding note that a 16 character string will require a lot of memory. With dictionary encoding the column would instead contain integers. Even by conservative estimates (8 byte keys to the dictionary) this would halve the size of the column and thus heavily speed up IO (note that we never need to read the product id column for this query).

In fact, it would even be possible to combine the two approaches.

Question 5: (4)

(a) Evaluate the following *Cypher* query on the Database given on the last sheet of the exam:

```
MATCH (start {name:"Virgil"})-[:Fan]-(i1)-[:Fan]->(i2)-[:Fan]-(start)
RETURN start.name, i1.name, i2.name
```

Note: Don't forget about the directed edge in the query.

start.name	i1.name	i2.name
Virgil	Horace	Ovid
Virgil	Cattalus	Ovid

(b) Evaluate the following *Cypher* query on the Database given on the last sheet of the exam:

```
MATCH p = shortestPath((a {name: "Metamorphoses"})-[*]-(b {name: "Aeneid"}))
RETURN p
```

Metamorphoses -> Apuleius -> Ovid -> Virgil -> Aenid

(c) *Assume the data model described on the last sheet of the exam.*

Write a *Cypher* query that returns all the works written by fans of the poet named *Ovid* that have *at least* 100 `verses`.

```
MATCH (:Poet {name:  'Ovid'})<-[:Fan]-()-[:by]-(w:Work)
WHERE w.verses >= 100 RETURN w
```

(d) *Assume the data model described on the last sheet of the exam.*

Write a *Cypher* query that outputs the 3 poets with the highest number of fans. The output should be in descending order in the number of fans.

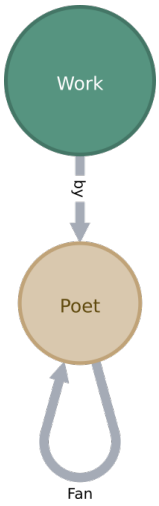
```
MATCH (n:Poet)<-[:Fan]-(a)
RETURN n, COUNT(a) AS fans
ORDER BY fans DESC
LIMIT 3
Note that counting the edges would also be fine for this question as it is
reasonable to assume at most one fan edge between two poets.
```

Overall: 20 points

Sheet for Question 5. You can tear off this sheet if you like.

Graph DB Data Model

The data model is visualized on the right. The graph contains nodes for poets and their works. They are identified with the labels `Poets` and `Work`, respectively. They all have at least the field `name`. All `Work` nodes also have a field `verses`, containing the number of verses in the work as an integer.
Poets can be fans of other poets, works are written by a poet. These relationships are modeled by edge labels `Fan` and `by`, respectively.



Graph Database

The nodes in the image contain the content of their ***`name`*** *field.*
See the schema visualizaton above for which color nodes have which label.

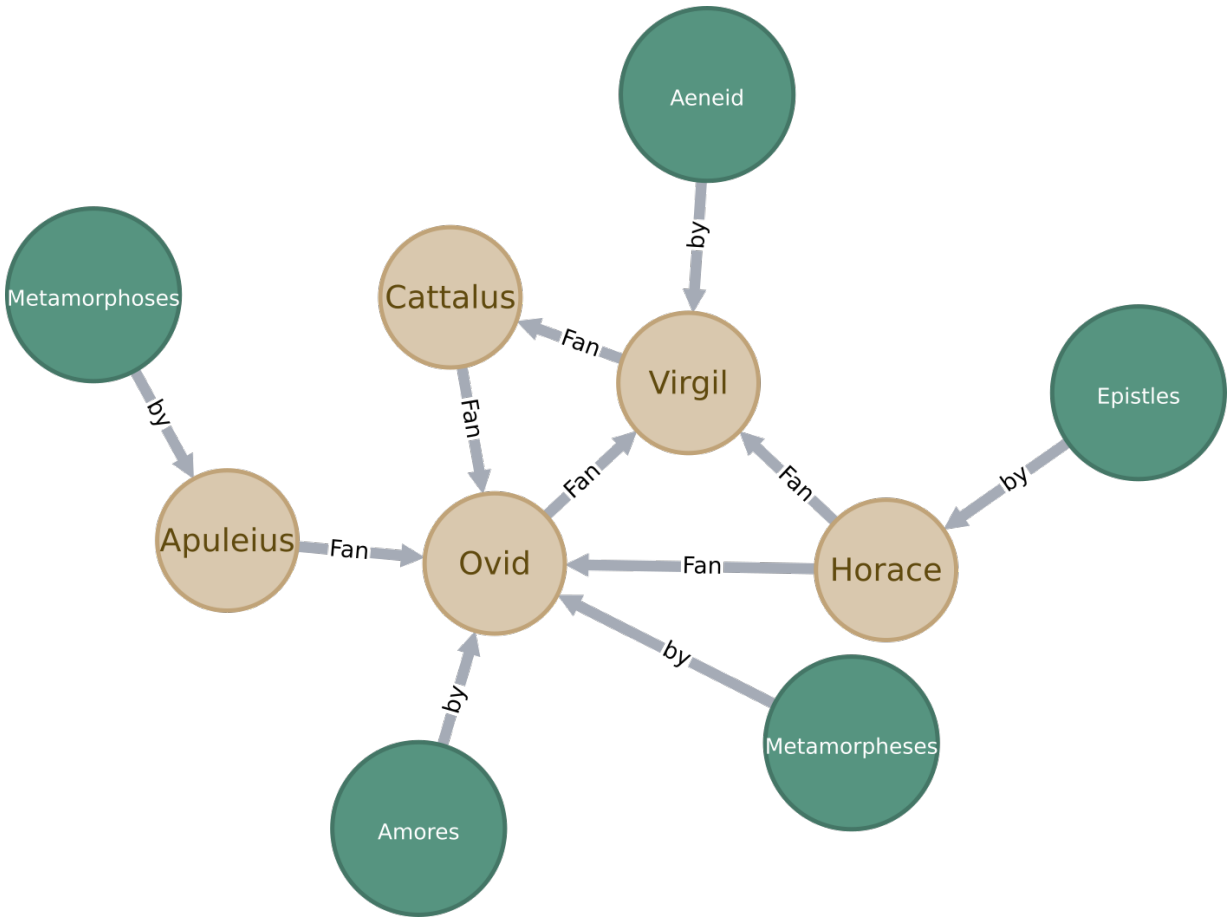


Figure 1: Database for Question 5