TEST Pearl 010 "Databases", September 19th 2019, 8:45-9:45

Question:	SQL	Databases	Database design	Total
Points:	20	10	10	40
Score:				

You may use 1 A4 document with your own notes for this exam and a *simple* calculator. Scientific or graphical calculators, laptops, mobile phones, books etc. are not allowed. ***Put those in your bag now!***

Question 1 (SQL) — Points: 20

In honour of the 50th anniversary of the first man on the moon, we focus on a simplified database that holds data on space missions. A description of the table structure can be found in Figure 1.

The database obviously has a table with astronauts and a table with missions. It has a table "MisAst" that stores which astronauts went on which missions. Unmanned missions do not have a row in this table. Moreover the database stores special dates for each mission such as the launch and return to earth, optionally also date of landing or impact on the moon. To provide some data on 'the space race', a final table holds achievements of missions and astronauts and the country who obtained this achievement.

Figure 2 contains some example data. Notice that there are three unmanned missions and that Svetlana Y. Savitskaya (astronaut 6) has two achievements on her name.

Note: In the requested SQL queries and statements, only use values given in the question, i.e., do not directly use identifier numbers from the example data unless they are explicitly given in the question.

Tip: Figure 4 contains an informal description of the syntax of SQL.

(a) Write an SQL query that produces the description of all Russian achievements.

Solution:
In the evaluation of these questions, special attention was given to correct usage of
- value conditions (C),
- join conditions (J),
- aggregation (A), i.e., group by and count,
- inserts, deletes, and updates (U), and
- SELECT-clause and other SQL features (S) such as DISTINCT and ORDER BY.

This question requires a value condition (C)

```
SELECT description
FROM Achievement
WHERE country = 'Russia'
```

(b) Write an SQL query that gives the name of the astronaut who was the "1st woman in space" and the name of her mission.

Solution:
This question requires two joins (2J) and a condition (C).

```
SELECT as.name, m.name
FROM Astronaut as, Achievement ach, Mission m
WHERE ach.description = "1st␣woman␣in␣space"
AND ach.asid = as.asid
AND ach.mid = m.mid
```

(c) Write an SQL query that gives per ***manned*** mission its name and the number of achievements the mission obtained.

Solution:
This question requires two joins (2J), and an aggregate (A).

```
SELECT m.name, COUNT(*)
FROM Mission m, Achievement a, MisAst ma
WHERE m.mid = ma.mid
AND m.mid = a.mid
GROUP BY m.name
```

In the above, the join with table MisAst functions as the selection for ‘manned’: if an astronaut is associated with the mission, then it is a “manned mission”.
Some students tried to solve the question with a join with Achievement and a condition “`a.asid IS NOT NULL`”. This thinking is not entirely correct: it is possible that there are missions without achievements, but at the same time, the solution above also does not produce a ‘0 achievements’ for such missions. One would need an *outer join* to achieve this, but that is not part of the learning goals of this pearl. Therefore we have also accepted the solution below (and correct variants of it)

```
SELECT m.name, COUNT(*)
FROM Mission m, Achievement a
WHERE m.mid = a.mid
AND a.asid IS NOT NULL
GROUP BY m.name
```

Some students interpreted “manned mission” as requiring only men on the mission, no women. Although this is not what in English the meaning is of “manned mission”, we are not teaching English in this pearl, so we also accept correct solutions for this interpretation. Under this interpretation, a correct solution could look like this (correct variations of it are also acceptable).

```
SELECT m.name, COUNT(*)
FROM Mission m, Achievement a
WHERE m.mid = a.mid
AND description LIKE '%man%'
AND NOT(description LIKE '%woman%')
GROUP BY m.name
```

(d) Write an SQL statement that removes the achievement "1st US lunar impact".

Solution:
This question requires a DELETE statement (U) with a condition (C).

```
DELETE FROM Achievement
WHERE description = "1st␣US␣lunar␣impact"
```

(e) Give an SQL query that gives certain information about the "1st man on the moon": the name of the astronaut, the name of the mission, the country, and the "landing" date.

Solution:
This question requires 2 conditions (C) and three joins (3J).

```
SELECT as.name, m.name, ach.country, d.dmr
FROM Astronaut as, Mission m, Achievement ach, Dates d
WHERE ach.description = "1st␣man␣on␣the␣moon"
AND d.type = "landing"
AND as.asid = ach.asid
AND m.mid = ach.mid
AND d.mid = ach.mid
```

(f) Write an SQL query that gives all names of astronauts with how many missions they have done (the example data has exactly one mission per astronaut, but the query should also give the right numbers if the tables would have data on more astronauts and missions).

Solution:
This question requires 1 aggregate (A) and 1 join (J).

```
SELECT as.name, COUNT(*)
FROM Astronaut as, MisAst ma
WHERE as.asid = ma.asid
GROUP BY as.name
```

(g) Write an SQL query that gives the launch, landing, and return date of the "Apollo 11" mission. The form of the result should be one row with three attributes 'launch', 'landing' and 'return'.

Solution:
This question requires 3 joins (3J), and 4 conditions (4C).

```
SELECT d1.dmr AS "launch", d2.dmr AS "landing", d3.dmr AS "return"
FROM Mission m, Dates d1, Dates d2, Dates d3
WHERE d1.type = "launch" AND d1.mid = m.mid
AND d2.type = "landing" AND d2.mid = m.mid
AND d3.type = "return" AND d3.mid = m.mid
AND m.name = "Apollo␣11"
```

Question 2 (Databases) **Points: 10**

(a) [1 points] A “Database Management System” (DBMS) is
- ◯ A computer
- ✓ **A piece of software**
- ◯ A collection of data

(b) [1 points] Which role is involved in the operational use of the database?
- ◯ Application programmer
- ✓ **Database administrator**
- ◯ Database designer
- ◯ Information analyst
- ◯ Information manager

(c) [1 points] Which statement about the ANSI/SPARC architecture is incorrect?
- ◯ A main purpose of the ANSI/SPARC architecture is *data independence*
- ◯ The notion of *External schema* is defined in the ANSI/SPARC architecture
- ◯ How data is exactly stored on disk in terms of bytes is of concern of the ANSI/SPARC architecture
- ✓ **The role of *Information analyst* is defined in the ANSI/SPARC architecture**

(d) [1 points] Which is *not* a requirement for a DBMS?
- ◯ High availability
- ◯ High reliability
- ◯ High speed
- ◯ High durability
- ◯ High security
- ✓ **High usability**

(e) [1 points] Which statement about OLAP is correct?
- ◯ OLAP uses relatively many queries and transactions.
- ✓ **Each query or transaction affects relatively large amounts of data.**
- ◯ Typical applications are operational administrative processes.
- ◯ OLAP is an acronym and stands for “On-Line Application Procession”.

(f) [1 points] SQL (Structured Query Language) is
- ◯ only a Query Language (QL)
- ◯ a QL and Data Manipulation Language (DML)
- ◯ a QL and Data Definition Language (DDL)
- ◯ a QL and Storage Definition Language (SDL)
- ◯ a QL, DML, and DDL, but not an SDL
- ✓ **a QL, DML, DDL, and SDL (i.e., all of them)**

(g) [1 points] A *relation instance* is
- ◯ A specification of the structure of a table.

- ◯ A pair of records of two different tables where one record has a foreign key referring to the primary key of the other.
- ◯ A pair of tables where one table has a foreign key referring to the primary key of the other.
- √ **The contents of all attributes and all rows of a table.**

(h) [1 points] Which statement about keys is *correct*?

- ◯ A composite key can never be a primary key
- ◯ A surrogate key can never be a primary key
- ◯ A foreign key can never be a primary key
- ◯ A foreign key can never be a composite key
- ◯ A foreign key can never be a surrogate key
- √ **A surrogate key can never be a composite key**

(i) [1 points] Which is *not* a benefit of *views*

- ◯ they distribute refined access rights
- ◯ they simplify query design
- ◯ they make queries more efficient
- √ **they allow more users to access the database**

(j) [1 points] The data in a database is often queried or changed by many users and applications at the same time. This is indicated in database terminology with the term

- √ **Concurrency**
- ◯ Parallellism
- ◯ Isolation
- ◯ Atomicity

Question 3 (Database design) **Points: 10**

Figure 3 contains part of the data model belonging to a database of a simplified book information system. It keeps track of books with some information about title, abstract, genre(s), the writer(s) of these books, and the publishers of these books.

(a) [10 points] Given this ER-model, design a table structure for this model. Give your answer as a list of tables with for each table:

1. the name of the table,
2. the names of the attributes,
3. which attribute(s) form the primary key, and
4. which attribute(s) are foreign keys and what they refer to.

FORMATTING INSTRUCTIONS: The answer box has rich text formatting capabilities. In the second row in the middle, there is an icon that looks like a 3-by-3 table. Use it to create a table with 4 columns (one column for each of the above) and as many rows you need (one header row with column headings and below it one row per table).

Solution:

Table name	**Attributes**	**Primary key**	**Foreign key(s)**
Writer	wid, name, birthdate	wid	*none*
Genre	gid, name	gid	*none*
Publisher	pid, name, country	pid	*none*
Book	isbn, title, abstract, pid	isbn	pid refers to Publisher(pid)
WriterBook	wid, isbn	wid, isbn	wid refers to Writer(wid); isbn refers to Book(isbn)
GenreBook	gid, isbn	gid, isbn	gid refers to Genre(gid); isbn refers to Book(isbn)

Some students have not provided separate tables for the many-to-many relationships, i.e., there is no counterpart for the “WriterBook” and “GenreBook” tables of the reference solution above. This is wrong in general and will have many points subtracted for missing tables, missing primary keys and incorrect foreign keys.

Some students have argued, however, that one can replicate the book information over multiple rows as many as needed and use as primary key the attributes isbn,wid,gid. A book with 2 writers and 3 genres will then have have 5 or 6 rows. A few more points have been awarded for this approach with appropriate and correct explanations about it.

Table	**Attributes / Description**
Astronaut	**asid**:int, name:text
	Table that holds the astronauts. 'asid' is a unique number for each astronaut. 'name' is the name of the astronaut.
Mission	**mid**:int, name:text
	Table that holds the missions. 'mid' is a unique number for each mission. 'name' is the name of the mission.
MisAst	***mid***:int, ***asid***:int
	Table for storing which astronauts went on which missions. 'mid' refers to the mission. 'asid' refers to the astronaut. "Unmanned" missions do not have a row in this table.
Achievement	**achid**:int, description:text, country:text, *asid*:int, *mid*:int
	Table for storing achievements. 'achid' is a unique number for each achievement. 'description' is a textual description of the achievement. 'country' is the country that obtained the achievement. 'asid' refers to the astronaut associated with the achievement. 'mid' refers to the mission associated with the achievement. Achievements for unmanned missions have a *NULL* value for 'asid'.
Dates	**did**:int, type:text, dmr:date, *mid*:int
	Table that stores important dates for missions. 'did' is a unique number for each date. 'type' describes what the date pertains to. 'dmr' is the date itself (it stands for day/month/year). 'mid' refers to the mission the important date belongs to.

Figure 1: In honour of the 50th anniversary of the first man on the moon, we focus on a table structure of a simplified database that holds date on space missions (Primary keys in **bold**; Foreign keys in *italics*).

Student number: .... Name: ....

Astronaut

asid	name
1	Neil A. Armstrong
2	Edwin E. Aldrin Jr.
3	Michael Collins
4	Yuri A. Gagarin
5	Valentina Tereshkova
6	Svetlana Y. Savitskaya
7	Sally Ride

Mission

mid	name
10	Luna 2
11	Luna 16
12	Ranger 4
13	Apollo 11
14	Vostok 6
15	Soyuz T-7
16	STS-7
17	Vostok 1

MisAst

mid	asid
13	1
13	2
13	3
14	5
15	6
16	7
17	4

Dates

did	type	dmr	mid
20	launch	12/9/1959	10
21	impact	13/9/1959	10
22	launch	12/9/1970	11
23	landing	20/9/1970	11
24	return	24/9/1970	11
25	launch	23/4/1962	12
26	impact	26/4/1962	12
27	launch	16/7/1969	13
28	landing	20/7/1969	13
29	return	24/7/1969	13
30	launch	16/6/1963	14
31	return	19/6/1963	14
32	launch	19/8/1982	15
33	return	10/12/1982	15
34	launch	18/6/1983	16
35	return	24/6/1983	16
36	launch	12/4/1961	17
37	return	12/4/1961	17

Achievement

achid	description	country	asid	mid
50	1st lunar impact	Russia	*NULL*	10
51	1st lunar soil return	Russia	*NULL*	11
52	1st US lunar impact	US	*NULL*	12
53	1st man on the moon	US	1	13
54	2nd man on the moon	US	2	13
55	1st woman in space	Russia	5	14
56	2nd woman in space	Russia	6	15
57	1st woman space walk	Russia	6	15
58	1st US woman in space	US	7	16
59	1st man in space	Russia	4	17

Figure 2: Example data for the "space mission" database (source: several pages of Wikipedia)

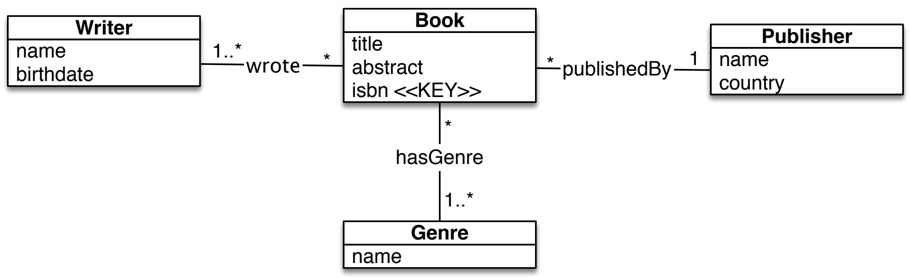

Figure 3: ER model of a book database (for Question 3).

In the informal syntax, we use the following notations

- $A|B$ to indicate a choice between A and B
- $[A]$ to indicate that A is optional
- $A*$ to indicate that A appears 0 or more times
- $A+$ to indicate that A appears 1 or more times
- 'A' to indicate that the symbol A is literally that symbol

We are not precise in punctuation in the syntax, but this is irrelevant in this exam anyway.

```
createtable : CREATE TABLE tablename '(' columndef+  constraint*  ')'
query : SELECT ( column [ AS colname ] ) +
        FROM ( tablename [ AS colname ] )+
        WHERE condition
        [ GROUP BY column+  ] [ ORDER BY column+  ]
statement : delete | update | insert

columndef : colname type [NOT NULL] [UNIQUE] [PRIMARY KEY]
            [REFERENCES tablename (colname+)]
constraint : PRIMARY KEY (colname, ... ) | CHECK ( condition )
             | FOREIGN KEY(colname, ... ) REFERENCES tablename(colname, ... )
column : [ tablename '.' ] colname | ' * ' | aggregate
aggregate : ( COUNT | SUM | AVG | MIN | MAX ) '(' column | ' * ' ')'
Examples of condition: column = value [ (OR | AND) [NOT] column <> value ]
                       | column IS [NOT] NULL
                       | column [NOT] IN (value, ... ) ...
delete : DELETE FROM tablename WHERE condition
update : UPDATE tablename SET ( column = value) *
         FROM ( tablename [ AS colname ] )+
         WHERE condition
insert : INSERT INTO tablename [ '(' colname+  ')' ]
         query | VALUES '(' value+  ')'
```

Figure 4: Informal syntax of SQL (for Question 1)