CMPUT 291
Midterm Examination – Feb 15, 2018

Section: B1/EB1

Instructor: Joerg Sander

Name _____

Signature _____

Instructions:

1. Turn off your cell phone.
2. Sign your name on the signature line.
3. Place your student id number in the designated spot on the top of the **next** page.
4. The time for this test is 70 minutes.
5. The exam is "closed-book", i.e., no references, notes, books, calculators, or smart phones are allowed.
6. Place all answers in this booklet and do not hand in any other work.

Student Id Number: _____

Marks obtained:

Part 1	Part 2	Part 3	Total
/15	/22	/30	/67

Part 1 [15 marks]

1. [3 marks] In the provided space, briefly characterize the different schemas in a database system. Use a short phrase or single sentence.

1. **External Schema:**

   What application programs and users see (Views)

2. **Conceptual Schema:**

   Description of the logical structure of the data (Table Schemata)

3. **Physical Schema:**

   File structures and indexes used to implement the tables

2. [2 marks] What does "logical data independence" refer to, in the context of DBMSs? Select only one of the following statements.

☐ Changes in the logical schema of several tables can be made independently of each other.

☐ The physical schema can be changed without affecting the conceptual schema.

☐ The external schema can be changed without affecting the logical schema.

☒ The conceptual schema can be changed without affecting the applications.

3. [3 marks] What does a *Relation* in the relational data model consists of?

1. A schema with (2 marks)
   - **i. Name of the relation**
   - **ii. Name and type of each column**
   - **iii. Possible integrity constraints**

2. An instance, which is a set of rows (an actual table) (1 mark)

4. [2 marks] Assume the following SQL Create Table command, which may or may not be correct, and assume that Table_Q4_2 already exists:

```
CREATE TABLE Table_Q4_1 (
    A1  CHAR(34),
    A2  CHAR(21),
    A3  CHAR(2),
    A4  CHAR(10),
    PRIMARY KEY (A1, A2),
    FOREIGN KEY (A1) REFERENCES Table_Q4_2)
```

Select the *one* (and only one!) correct statement among the following statements.

☐ (A1, A2) cannot be the primary key since it its not minimal.

☒ The attribute A1 uniquely determines a row in table Table_Q4_2.

☐ The attribute A1 uniquely determines A2, and A2 uniquely determines A1.

☐ A1 cannot be in the PRIMARY KEY clause and at the same time in the FOREIGN KEY clause.

5. [2 marks] Assume the following SQL Create Table command (and assume that Table_Q5_2 already exists):

```
CREATE TABLE Table_Q5_1 (
    A1  CHAR(34),
    A2  CHAR(21),
    A3  CHAR(2),
    A4  CHAR(10),
    PRIMARY KEY (A1),
    FOREIGN KEY (A3) REFERENCES Table_Q5_2
                        ON DELETE CASCADE)
```

Select the *one* (and only one!) correct statement among the following statements.

- ⊗ When a tuple *t* in Table_Q5_2 is deleted, also all tuples in Table_Q5_1 with the same value for A3 as *t* will be deleted.
- ☐ When a tuple *t* in Table_Q5_1 is deleted, also all tuples in Table_Q5_2 with the same value for A3 as *t* will be deleted.
- ☐ When a tuple *t* in Table_Q5_1 is deleted, also all other tuples in Table_Q5_1 with the same value for A3 as *t* will be deleted.
- ☐ When a tuple *t* in Table_Q5_2 is deleted, also all other tuples in Table_Q5_2 with the same value for A3 as *t* will be deleted.

1. [3 marks] Determine if the following statement is correct or incorrect in the context of the *Relational Model*, and briefly justify your answer (with an argument that refers to the central definitions relevant to this question).

"***Every relation has a key***."

	YES	*NO*
***TRUE*?**	⊗	

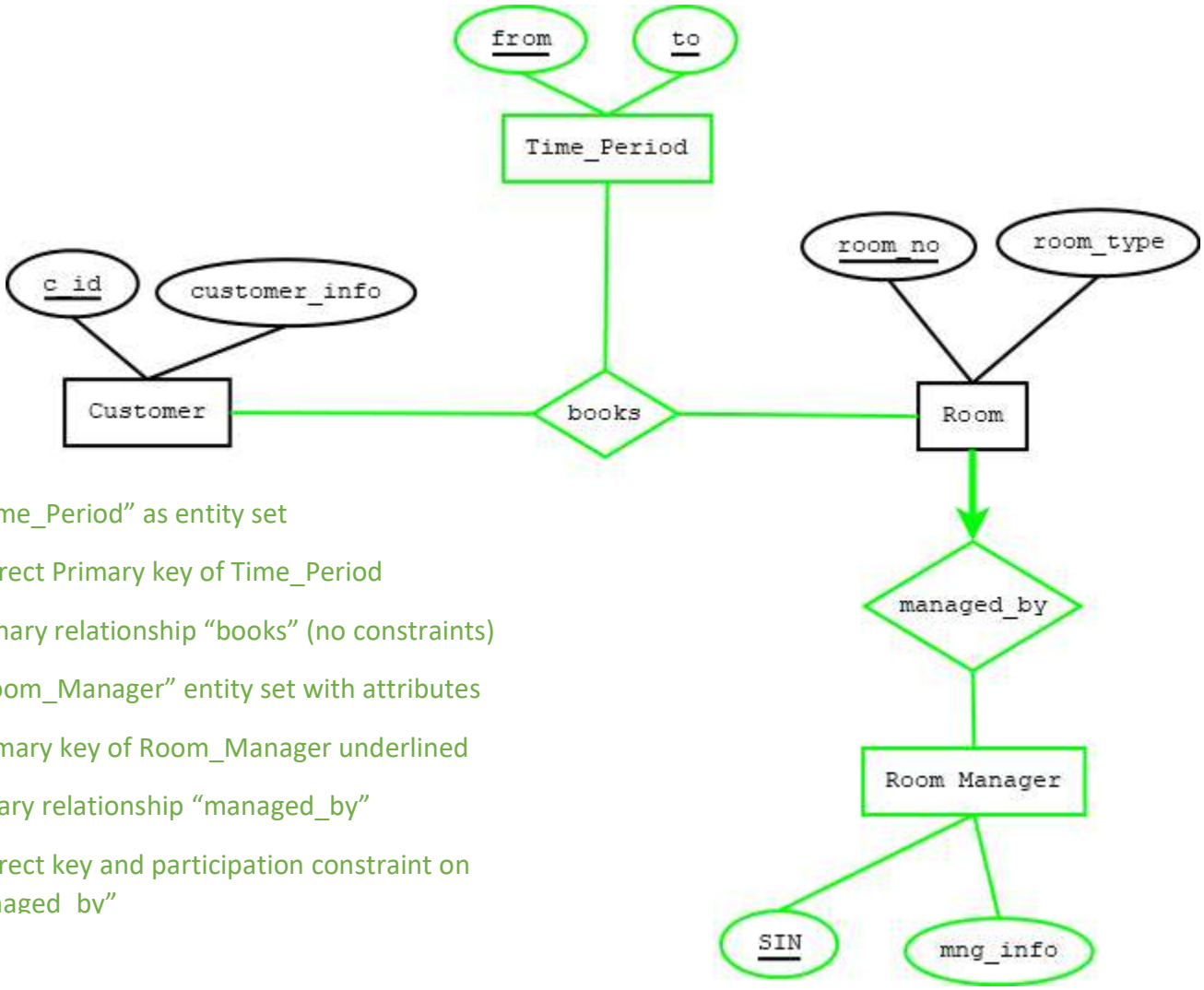
Justification:

A relation according to the relational model is a SET of rows, i.e., the combination of all attributes uniquely determines a row => the combination of all attributes is a *superkey*, i.e., by definition of superkey, *contains* a key.

Part 2 [22 marks]

1. [9 marks] Complete the ER diagram below according to the following specifications. Be sure to underline the primary keys in your diagram, and indicate any participation constraints, key roles, etc. The ER diagram represents a simplified model for part of a hotel management system:

A Customer can book rooms for a certain time period (from a start date to an end date); it is possible that a customer books several rooms for the same time period, or that a customer books the same room for multiple, different time periods.
Each room is managed by a single room manager, and room managers are described by a unique social insurance number (SIN) and some manager information.



1 "Time_Period" as entity set

1 correct Primary key of Time_Period

2 ternary relationship "books" (no constraints)

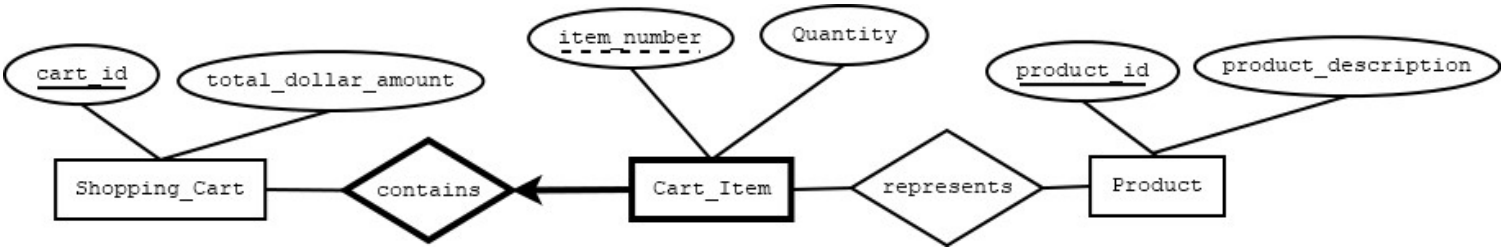
1 "Room_Manager" entity set with attributes

1 Primary key of Room_Manager underlined

1 binary relationship "managed_by"

2 correct key and participation constraint on "managed_by"

2. [13 marks] Translate the following ER diagram into relational tables using CREATE TABLE commands, so that as much of the information in the diagram as possible is represented (and nothing else that is not expressed in the diagram). Use suitable datatypes for the columns.




```
CREATE TABLE Shopping_Cart (
   cart_id TEXT,
   total_dollar_amount REAL,
   PRIMARY KEY (cart_id));
```

1 for attributes
1 for primary key

```
CREATE TABLE Cart_Items (
   item_number INT,
   quantity INT,
   cart_id TEXT,
   PRIMARY KEY (item_number, cart_id),
   FOREIGN KEY (cart_id) REFERENCES Shopping_Cart;
```

Merging Cart Items with contains (4 marks):

1 for all attributes
2 for correct primary key
1 for foreign key

```
CREATE TABLE Product (
   product_id TEXT,
   product_description TEXT,
   PRIMARY KEY (product_id),
```

1 for attributes
1 for primary key

```
CREATE TABLE represents (
   item_number INT,
   cart_id TEXT,
   product_id TEXT,
   PRIMARY KEY (item_number, cart_id, product_id),
   FOREIGN KEY (cart_id, item_number) REFERENCES Cart_Items,
   FOREIGN KEY (product_id) REFERENCES Product;
```

1 for attributes
2 for primary key
2 for 2 foreign keys

Part 3 [30 marks]

For the queries in this part, consider the following database that supports a "Brain Exercise" phone App in which users can play different types of games on their phone, and a central database keeps track of user information, game information, and player scores:

```
CREATE TABLE Users (
   email        TEXT,
   password     TEXT,
   username     TEXT,
   age_group    TEXT,
   PRIMARY KEY (username));

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')
```

```
CREATE TABLE Games (
    name        TEXT,
    category    TEXT,
    PRIMARY KEY (name));

Example tuple: ('number racer', 'math')
```

```
CREATE TABLE Scores (
    username                TEXT,
    game_name               TEXT,
    timestamp_played        DATE,
    score                   INTEGER,
    PRIMARY KEY (username, game_name, timestamp_played),
    FOREIGN KEY (username) REFERENCES Users,
    FOREIGN KEY (game_name) REFERENCES Games(name));

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)
```

1. [5 marks] Complete the following ***SQL*** query so that it lists the name and the email of every user who got a score higher than 5000, at least once, in the game 'number racer'.

```
SELECT   DISTINCT u.username, u.email
FROM     Users u, Scores s
WHERE    u.username = s.username AND
         s.game_name = 'number racer' AND
         s.score > 5000;
```

Tables repeated here for your convenience

```
CREATE TABLE Users (
    email       TEXT,
    password    TEXT,
    username    TEXT,
    age_group   TEXT,
    PRIMARY KEY (username));
```

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')

```
CREATE TABLE Games (
    name        TEXT,
    category    TEXT,
    PRIMARY KEY (name));
```

Example tuple: ('number racer', 'math')

```
CREATE TABLE Scores (
    username                TEXT,
    game_name               TEXT,
    timestamp_played        DATE,
    score                   INTEGER,
    PRIMARY KEY (username, game_name, timestamp_played),
    FOREIGN KEY (username) REFERENCES Users,
    FOREIGN KEY (game_name) REFERENCES Games(name));
```

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)

2. [5 marks] Write an SQL query that returns the usernames of all users who played both the game 'number racer' and the game 'text racer'.

```
--SOLUTION 1:
SELECT username
FROM Scores
WHERE game_name = 'number racer'
INTERSECT
SELECT username
FROM Scores
WHERE game_name = 'text racer';
```

```
--SOLUTION 2:
SELECT DISTINCT username
FROM Scores s1, Scores s2
WHERE s1.username = s2.username
   AND s1.game_name = 'number racer'
   AND s2.game_name = 'text racer';
```

Tables repeated here for your convenience

```
CREATE TABLE Users (
    email       TEXT,
    password    TEXT,
    username    TEXT,
    age_group   TEXT,
    PRIMARY KEY (username));
```

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')

```
CREATE TABLE Games (
    name        TEXT,
    category    TEXT,
    PRIMARY KEY (name));
```

Example tuple: ('number racer', 'math')

```
CREATE TABLE Scores (
    username            TEXT,
    game_name           TEXT,
    timestamp_played    DATE,
    score               INTEGER,
    PRIMARY KEY (username, game_name, timestamp_played),
    FOREIGN KEY (username) REFERENCES Users,
    FOREIGN KEY (game_name) REFERENCES Games(name));
```

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)

3. [9 marks] Create a VIEW "Averages" with 3 columns named "gname", "agroup", and "avg_score" that computes for each combination of a game (gname) and an age group (agroup) the average score (avg_score) for that game in that age group. Fill in the missing parts of the following query template:

```
CREATE VIEW Averages (gname, agroup, avg_score) AS
SELECT      s.game_name, u.age_group, AVG(s.score)
FROM        Scores s, Users u
WHERE       u.username = s.username
GROUP BY    s.game_name, u.age_group
```

Tables repeated here for your convenience

```
CREATE TABLE Users (
   email        TEXT,
   password     TEXT,
   username     TEXT,
   age_group    TEXT,
   PRIMARY KEY (username));
```

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')

```
CREATE TABLE Games (
    name        TEXT,
    category    TEXT,
    PRIMARY KEY (name));
```

Example tuple: ('number racer', 'math')

```
CREATE TABLE Scores (
    username                TEXT,
    game_name               TEXT,
    timestamp_played        DATE,
    score                   INTEGER,
    PRIMARY KEY (username, game_name, timestamp_played),
    FOREIGN KEY (username) REFERENCES Users,
    FOREIGN KEY (game_name) REFERENCES Games(name));
```

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)

4. [5 marks] Complete the following SQL query so that it finds the usernames of players who have played all games.

```
SELECT u.username
FROM Users u
WHERE
```

```
--SOLUTION 1:
NOT EXISTS (
   SELECT name FROM Games
   EXCEPT
   SELECT game_name FROM Scores s WHERE u.username = s.username);
```

```
--SOLUTION 2:
 (SELECT COUNT(DISTINCT game_name)
   FROM Scores s WHERE s.username = u.username)
 =
  (SELECT COUNT(*) FROM Games);
```

Tables repeated here for your convenience

CREATE TABLE **Users** (
email TEXT,
password TEXT,
username TEXT,
age_group TEXT,
PRIMARY KEY (username));

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')

CREATE TABLE **Games** (
name TEXT,
category TEXT,
PRIMARY KEY (name));

Example tuple: ('number racer', 'math')

CREATE TABLE **Scores** (
username TEXT,
game_name TEXT,
timestamp_played DATE,
score INTEGER,
PRIMARY KEY (username, game_name, timestamp_played),
FOREIGN KEY (username) REFERENCES Users,
FOREIGN KEY (game_name) REFERENCES Games(name));

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)

5. [3 marks] Write an SQL statement that updates the email address of the user 'smartypants' to 'polysmart007@gmail.com'.

```
UPDATE Users
SET email = 'polysmart007@gamil.com'
WHERE username = 'smartypants'
```

Tables repeated here for your convenience

CREATE TABLE **Users** (
email TEXT,
password TEXT,
username TEXT,
age_group TEXT,
PRIMARY KEY (username));

Example tuple: ('bs23@gmail.com', '****', 'brainiac0', '20-25')

CREATE TABLE **Games** (
name TEXT,
category TEXT,
PRIMARY KEY (name));

Example tuple: ('number racer', 'math')

CREATE TABLE **Scores** (
username TEXT,
game_name TEXT,
timestamp_played DATE,
score INTEGER,
PRIMARY KEY (username, game_name, timestamp_played),
FOREIGN KEY (username) REFERENCES Users,
FOREIGN KEY (game_name) REFERENCES Games(name));

Example tuple: ('brainiac0', 'number racer', '2018-02-10 17:36:23', 124)

6. [3 marks] Consider the following SQL query

```
SELECT s.username, s.game_name, s.score
FROM Scores s
EXCEPT
SELECT s1.username, s1.game_name, s1.score
FROM Scores s1, Scores s2
WHERE s1.score < s2.score AND s1.game_name = s2.game_name;
```

What is the result of this query? Select only one option!

- ☐ The empty set.
- ☐ A single result tuple (*u, g, s*) that represents a user *u* who achieved the highest score *s* overall, i.e., independent of the game *g*.
- ☐ A single result tuple (*u, g, s*) that represents a user *u* who achieved the lowest score *s* overall, i.e., independent of the game *g*.
- ☒ Each result tuple (*u, g, s*) represents a user *u* who achieved the high-score *s* for game *g*. The result contains for each game a tuple for every user who got the high-score.
- ☐ Each result tuple (*u, g, s*) represents a user *u* who achieved the lowest score *s* in game *g*. The result contains for each game a tuple for every user who got the lowest score.
- ☐ The result contains all tuples (*u, g, s*) from the Scores table (without the timestamp_played), except for the tuples that represent the high scores for each game.