Database Design Laboratory

➢ Timetable

I. Relational Databases (8 weeks)

- Database Design and SQL Queries, Microsoft SQL Server - ACB Spanish Basketball League (3 weeks)
- SQL views, functions, stored procedures, triggers, OOP extensions - ACB Spanish Basketball League (3 weeks)
- Design and implement an original database with around 10 tables & the above mentioned elements – MS SQL Server (2 weeks)

II. NoSQL Databases (6 weeks)

- MongoDB database design and implementation, queries, GUI
- Design and build your own NoSQL database on the desired domain, also a small software application that communicates with this database

➢ Deadlines:

- **Individual assessment after each section**
- **January 10, 17 –** final presentation & final laboratory mark

Laboratory Work No. 1

1. Build a relational structure in a Microsoft SQL Server Database that describes the Spanish Basketball League, having the following tables:
   - Actors
   - Matches
   - Participants
   - Teams
   - Team descriptions.

1.1. The complete specifications can be retrieved at the following address: https://www.kaggle.com/jgonzalezferrer/acb-spanish-basketball-league-results

Build the database diagram.
Is the database normalized? Normalize the database and bring it at least in the 3rd Normal Form (3NF).
Employ statements such as Create Table, Alter Table, Insert INTO new_table, Select Distinct, in Microsoft SQL Server (Transact-SQL).

1.2. Impose unicity constraints (primary keys, candidate keys, unique index), user defined domain constraints, referential integrity constraints, participation constraints.

1.3. Solve the following SQL queries:

1.Display the first 10 distinct team names, from the alphabetically ordered team list.

2.Display the complete name, nationality and birth place for all the trainers within the database.

3.Display the distinct names of the players and the names of the teams where they belong. Are there players who belong to at least 2 teams?

4. Display the acb-id-s of the teams that have activated both in 2014 and in 2016.

5. Display the number of matches played in each year.

6.Find the names of the players and the number of matches(games) to which each player has participated. List also the players that have not participated to any game.

7. List the names of the teams and the number of members for each team.

8. List the name and the age for the youngest actor within the database and the number of games to which he participated.

9. List the name of the host team, the name of the guest team and the location, for the games that were played during the year 2017.

10.List the names of the host teams that arrived in the final of the competition during the last 3 years.

1.4. Distinguish the identifying/nonidentifying relationships and the weak/strong entity sets.

- **References:**

  CREATE TABLE (Transact-SQL) - SQL Server | Microsoft Docs
  ALTER TABLE (Transact-SQL) - SQL Server | Microsoft Docs
  DROP TABLE (Transact-SQL) - SQL Server | Microsoft Docs
  SELECT (Transact-SQL) - SQL Server | Microsoft Docs

Laboratory Work No. 2

1. Use the Transact-SQL statements INSERT, UPDATE şi DELETE in order to perform, within the Spanish Basketball League Database, insert, update and delete operations, as indicated below.

   • **INSERT operations**
   a. Insert at least two new actors within the database, using the Transact-SQL *INSERT* statement;
   b. Insert records in the Participant table, associating the players with differnt teams and with games which took place after January 1, 2017.
   c. Insert images within the database – each team must have its' own logo. Use the Transact-SQL type varbinary(max). Instantiate the field values using the script from the following example:

   - UPDATE flower SET image = (SELECT BulkColumn FROM OPENROWSET (Bulk 'D:\tmp\img_floare.jpg', SINGLE_BLOB) AS BLOB) WHERE flower_id = 1;

   Obtain the value of the image field using the following script:

   - EXEC xp_cmdshell 'BCP "SELECT image FROM flower WHERE flower_id = 1" queryout D:\tmp\img_floare.jpg -T '

   • **UPDATE operations**
   a. The player Pablo Aguilar Bermudez becomes a striker (A).
   b. The player Albert Oliver Campos becomes a coach for the MORABANC ANDORRA team.
   c. The year of foundation for the team that has, during the season 2016, the name MOVISTAR ETUDIANTES is actually 1950, not 1948. Correct the error.
   d. All the players older than 35 become referees, as well.

   • **DELETE operations**
   a. Delete the actors who are older than 42 years.
   b. Delete those games in which the actor Albert Oliver was involved.

2. Solve the following queries using SQL:
   a. The Acb-id-s and the founding year for each team that contains players having at least three different nationalities.
   b. The Acb-id-s of those teams that have won all the games to which they participated.
   c. The Acb-id-s of those teams that have won, during the season 2016, more games than during the season 2017.
   d. The names of those players that have activated at least 2 consecutive years.
   e. The names of the teams that haven't won any game.
   + other 5 queries that you consider useful.

3. Considering the Spanish Basketball League database, implement the

concept of class (type-subtype) hierarchies, where the main type is the Actor and the derived types are: Player(Id ⇔ Actor_Id, Team_Id, Start_Date, End_Date, Position, No-of_goals), Coach(Id⇔Actor_Id, Team_Id, Start_Date, End_Date), Referee(Id⇔Actor_Id, Start_Date, No_of_refereed_matches). Note that the subclasses inherit all the attributes of the main class.

- **Documentation:**
  INSERT (Transact-SQL) - SQL Server | Microsoft Docs
  UPDATE (Transact-SQL) - SQL Server | Microsoft Docs
  DELETE (Transact-SQL) - SQL Server | Microsoft Docs
  SELECT (Transact-SQL) - SQL Server | Microsoft Docs
  https://pressbooks.pub/cmiller1137/back-matter/supertypes-and-subtypes/

Laboratory Work No. 3

1. Assessment of Laboratory Work No. 1 and Laboratory Work No. 2.

Laboratory Work No. 4

1. Create within the Spanish Basketball League database the following views:

a. The names of the teams that correspond to each season and the year of appearance for each team.

b. The full player name, the nationality and the names of the teams where he belonged. If the actor didn't belong to any team, the „without team" string should be displayed.

c. The name of the host team, the name of the guest team, the start date and hour, as well as the location for all the games played in the final quarter phase.

d. The team name, the season (year) and the number of games played by each team per season. If a certain team played no match in a certain season, the 0 value will be displayed.

e. The name of the team and number of matches played by each team, as a guest team.

2. Create, within the Spanish Basketball League database, the following Transact-SQL functions, in order to return a single value (*scalar functions*):
   a. The number of games won by the team whose name is provided as a parameter, during the year provided as parameter.
   b. The name of the team that won al the games she played during the year provided as a parameter.
   c. The full name of the coach who trained a minimum number of teams, provided as parameter.
   d. The number of teams to which the actor whose acb-id is provided as parameter belonged.
   e. The name of the team that arrived at least once in the final phase of the competition since the year provided as parameter.

3. Create, within the Spanish Basketball League database, the following Transact-SQL functions, in order to obtain a record set (*table-value functions*):
   a. The names of the actors that acted as players for the team provided by parameter (by team-name), during the year provided as parameter.
   b. The names of the referees that arbitrated at least two games for the team whose acb-id is provided as a parameter.
   c. The names of the teams to which the actor whose name is provided as

parameter belonged.

d. The names of the actors that activated in two seasons, provided as parameters.

https://learn.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql?view=sql-server-2017

Laboratory Work No. 5

1. Create, in Microsoft SQL Server, Management Studio, a database for hotel management. Consider one of the following websites as a model:
https://www.tourismguide.ro/tgr/hotel_piemonte_predeal_1357.php
http://www.vilaeuropa.ro/?gclid=CNuJ_s7y3aACFQkeZwodOml0Mg

1.1. Design and implement the following entity types:

- Room types
- Stays – period of vacation
- Services
- Rooms – number, type, state: free/occupied
- Clients – name and surname, CNP, occupation
- Bookings – client, room, service, stay, total cost

as well as other entity types, specific to the web-site. The price for each room varies as a function of period. Populate this database with appropriate data, including images for different room types.

1.2. Build the database diagram, illustrating the relationships between the tables. Define primary and foreign keys, unicity constraints and referential integrity constraints.

2. Create a ***cursor*** in order to browse the Bookings table and to insert in a *History_Reservations* table those bookings from the current year for which the staying period has ended (the end date is less than the current date).

3. Create the following ***stored procedures***:

a. A stored procedure to illustrate the services and room types that are available in a certain period, specified through parameters (start date, end date), also the corresponding prices.
b. A stored procedure to insert bookings initiated by the client whose name is specified by a parameter, the room type and reservation period being also specified through parameters. The procedure should also find one of the free rooms which corresponds to the required type and period and to modify the state from „free” to „occupied”. Instantiate an output parameter with the number of the room that became occupied.
c. A stored procedure to insert clients into the database, having the following parameters: name, surname, CNP, occupation.
d. A stored procedure to modify the discount associated to a period of vacation, identified through the following parameters: start date, end date for that period.
e. A stored procedure for deleting a booking, receiving as parameters the CNP of the client, the start date, respectively the end date.

- **Documentation:**

https://docs.microsoft.com/en-us/sql/t-sql/language-elements/declare-cursor-transact-sql?view=sql-server-ver15
https://www.mssqltips.com/sqlservertip/1599/cursor-in-sql-server/
http://msdn.microsoft.com/en-us/library/ms187926.aspx

Laboratory Work No. 6

1. **Create a stored procedure that returns a cursor, in order to browse the items from Clients, then use this procedure in order to delete those clients who are older than 90, respectively to display the data regarding the other clients.**

2. **Create, within the Hotel Management database, the following triggers:**
   a. Trigger for insert/update in the Clients table: all the registered clients must be older than 18 years.
   b. Trigger for delete on the Bookings table, to disallow the deletion of those reservations made for a period that includes the current date; if other reservations are being deleted, these should be inserted in the History_Reservations table, having the same structure with the Bookings table.
   c. Trigger to disallow the reservation of an already occupied room.
   d. Trigger to disallow the deletion from the database of those clients having the occupation of programmer.
   e. Trigger for insert/update in Rooms table: do not allow the existence of more than 5 rooms of type 'suite' in the database.
   f. Trigger to disallow the deletion of those stays that have their discount situated among the first three most increased discounts in the database.

3. **Recursive structures and queries – part 1:**

Create a database for modeling the tree of mathematicians, to offer information upon each mathematician and his supervisors, as indicated at http://genealogy.math.ndsu.nodak.edu/**.**

a. Define a view for displaying the direct supervisors of each mathematician.
b. Define an SQL Server scalar function to display the level within the tree of a certain node, corresponding to a mathematician identified through his name.
c. Define a stored procedure to display the number of disciples of gender 'M' for a certain node, respectively the number of disciples of gender 'F' of the same node, provided as parameter.
d. Define a trigger for restricting the removal of a certain node if this node has at least two descendants.

4. a. Extend the hotel management database to represent multiple hotels, by creating a new table called Hotels.
   b. Create a new SQL Server datatype called Location, characterized through the GPS coordinates - latitude and longitude. Integrate in the Hotel table a column to represent the location (optional).

Documentation:
https://learn.microsoft.com/en-us/sql/t-sql/statements/create-procedure-transact-sql?view=sql-server-ver16&redirectedfrom=MSDN

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15
https://www.sqlservertutorial.net/sql-server-basics/sql-server-recursive-cte/
https://learn.microsoft.com/en-us/sql/t-sql/statements/create-type-transact-sql?view=sql-server-ver16
https://www.sqlshack.com/an-overview-of-user-defined-sql-server-types/

Laboratory Work No. 7

1. Window functions in SQL Server

Create, in Microsoft SQL Server, within the Hotel Management database, the following views, by employing Window Functions:

a. A view in order to determine the number of clients per month during the year 2022, that had booked at least one reservation during 2022.
b. A view for determining the total price of the reservations for each hotel in the database, for each month, during 2022.
c. A view for determining the cheapest, as well as the most expensive room type in the database, together with the corresponding price, for each hotel.
d. A view to display the start date date, end date and discount of the stay that corresponds to the current row, respectively the start date, end date and discount that corresponds to the previous row in the Stays table. The stays must pe ordered by period (start date, end date).
e. A view to display the name, surname and age of the oldest client that has made reservation, for each hotel in the database.

https://www.sqlservertutorial.net/sql-server-window-functions/
https://www.sqlshack.com/use-window-functions-sql-server/

2. Recursive structures and queries – part 2:

Create a database for modeling the genealogy tree of the British Royal Family, https://britroyals.com/royalfamily.asp .

a. Define a query to display the number of descendants for each node.
b. Define a view to display each node with the corresponding direct descendants.
c. Create a stored procedure to display all the descendants of a certain node, provided as parameter.
d. Define a trigger to restrict the deletion of a certain node if this node has at least one descendant.

Laboratory Work No. 8-10

I. Choose a relevant domain and build a complex relational database, containing 8-10 tables, many-to-many relationships and also recursive relationships, then build the database diagram. Implement unicity constraints, user-defined domain constraints, also referential integrity rules. Include the following elements: views, window functions, sql server functions (both scalar and table-valued functions), cursors, stored procedures (for insert, update, delete, search), also triggers. Consider, as well, the possibility to implement class-hierarchies, also user-defined datataypes for the fields of your tables, this will certainly increase the value of your assignment.

II. Write a documentation in HTML or PDF format, for presenting the approached domain, then for describing in details your database and the corresponding elements.

Laboratory Work No. 11

1. Assessment for Laboratory Works 4-10.

2. In order to enable the work with NoSQL databases, install the MongoDB Shell and MongoDB Compass environments, from the following address: https://www.mongodb.com/try/download/compass .

3. In MongoDB Compass, create a new database, in order to manage the data regarding some companies. Create two collections, *Companies* and *Employees*. For *Employees*, consider the following attributes: name, surname, date of birth, date of employment, company name, function, salary. For *Companies*, consider the attributes name, address, city, country, foundation date, manager name. Import the data from previously created JSON or CSV files. Add new instances using the function Add Data/Insert Document. Visualize the structure of the database using the left hand side window, respectively the data from the collection, from the section View, using the options: *list view*, *JSON view* and *table view*.

4. Project the data from the *Employees* collection, using only the attributes name, surname, company, salary. Filter the data from Companies, in order to retain only those companies from Romania, Bucharest, sorted by the company name. Visualize the result in analytical form (Schema/Analyze). Run the corresponding NoSQL query in MongoDB Shell.

5. Build the following aggregations in MongoDB Compass:
   (a.) The name of the company and the name of the employee who has the highest salary, from each company. Sort the result set by the company name.
   (b.) The name of the city and the number of companies from each city, for the companies from Holland.

- **Documentation**:
  https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-compass
  https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/
  https://docs.mongodb.com/v4.4/mongo/

Laboratory Work No. 12

1. Choose your own domain and correspondingly build your own NoSQL database in the MongoDB environment. Define projections, filters and aggregation pipelines. Display the corresponding NoSQL queries.

2. Build a Graphical User Interface (GUI), preferably web interface, for your NoSQL database, using HTML and PHP (recommended). Connect to your MongoDB database, build an authentication form, and also adequate interfaces in order to perform the CRUD operations.

- **Documentation**:
  https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/
  https://www.mongodb.com/atlas/database
  https://www.php.net/manual/en/set.mongodb.php

Laboratory Work No. 13

1. Enhance your NoSQL MongoDB database using complex queries, involving the JOIN operations, as well as NoSQL Views.

2. Enhance your PHP web application by employing the Fat-Free Framework (F3)

3. Assessment for Laboratory Works 11-13

- **Documentation**:

  https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction
  https://www.mongodb.com/docs/manual/core/views/
  https://www.mongodb.com/docs/manual/core/views/create-view/#db.createview---syntax
  https://www.mongodb.com/docs/manual/core/views/join-collections-with-view/#std-label-manual-views-lookup
  https://fatfreeframework.com/3.6/home
  https://fatfreeframework.com/3.6/user-guide