Database Design Laboratory

➢ Timetable

I. Relational Databases

- ACB Spanish Basketball League – MS SQL Server (3 weeks)
- SQL Reports (3 weeks)
- Formula 1 Database – MS SQL Server Functions (2 week)
- Hotel/Turism Database, Genealogy Database – Stored Procedures, Triggers (2 weeks)

II. NoSQL Databases (4 weeks)

- **Individual assignment – build your own NoSQL database, on the desired domain**

  - **Deadlines:**

    November 15 – establish project title

    January 12, 19 – presentation

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. 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 won by each team per season. If a certain team won no match in a certain season, the 0 value will be displayed.

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

- **Documentation:**
  INSERT (Transact-SQL) - SQL Server | Microsoft Docs
  UPDATE (Transact-SQL) - SQL Server | Microsoft Docs
  DELETE (Transact-SQL) - SQL Server | Microsoft Docs
  CREATE VIEW (Transact-SQL) - SQL Server | Microsoft Docs

Laboratory Work No. 3

1. 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.

   https://s3.amazonaws.com/erwin-us/Support/95/CA+ERwin+Data+Modeler+r9+5-ENU/Bookshelf_Files/HTML/ERwin%20Overview/index.htm?toc.htm?254734.html

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

Laboratory Work No. 4

1. On the Spanish Basketball League Database, created in Microsoft SQL Server, Management Studio, create the following simple reports using the *Tableau* environment:

- **By performing direct connection to SQL Server:**

a. Actor's full name, nationality, and the number of teams where each player belongs. Implement a filter to display only those actors whose name begins with the ‚A' letter.
b. Name, acb-id and year of appearance for those teams that activated during the season 2016.
c. Acb-id of the team, year of appearance, the total number of games played as host team and the total number of games played as a guest team. Create a dashboard and include there the sheet, a relevant image, also a chart (Objects/Extensions/Add on charts) in order to illustrate the acb-id of the team and the number of games played as a host team.
d. The name of the host team, the name of the guest team, the competition phase and location for the games played during the year 2017.
e. The full name and acb-id of the actor, the name of the actual team and the number of matches to which he had participated during the current year. If the actor didn't participate to any matches this year, the 0 value should appear.

- **By connecting to Character Delimited Files, exported from SQL Server:**

  a. The nationalities and the number of players for each nationality.
  b. The full name, nationality and birth place for those actors who were active in both 2014 and 2016 seasons.
  c. The full name, nationality and birth place for all the referees in the database.

- Useful hyperlinks:
  https://www.tableau.com/academic/students
  https://help.tableau.com/current/pro/desktop/en-us/getstarted_buildmanual_ex1basic.htm

Laboratory Work No. 5

1. Taking into account the Spanish Basketball League database, implemented in Microsoft SQL Server, build the following complex reports in *Tableau*:
   a. A sheet(workbook) in order to illustrate the full actor name, the nationality, date and place of birth, as well as the number of games for the actor whose display name is provided as parameter. Create a dashboard and include there the sheet, a relevant image, also a chart (Objects/Extensions/Add on charts) in order to illustrate the names of the actors and the number of games played by each actor.
   b. A report (sheet) in order to illustrate all the games played by the team whose name is provided as parameter: name of the home team, the name of the guest team, the competition phase, the venue and the kickoff time.
   c. A report (sheet) in order to display the acbid and the name of the team, as well as the number of games played by the team as host team, during the years 2014, 2015 and 2016. The report should be created in tabular form, using the pivoting operation within an SQL SELECT statement, so all these fields should be specified in the table head.
   d. A report in order to display the full name of the actor, the nationality and also the details regarding the teams where the actor belongs, as a subreport. The subreport should contain the acbid of the team, the founding year, the team name and the corresponding season. Relate the two reports using the *team_id* field.
   e. A report in order to display the full name, nationality, birthplace, birthdate, position and height of the actor, and also the participant data, as a subreport. Regarding the participant data, instead of team_id, display the team_name. Relate the two reports using the *actor_id* field.

- **Documentation:**

  Create Parameters - Tableau
  Filtering with parameters (tableau.com)
  https://help.tableau.com/current/pro/desktop/en-us/qs_area_charts.htm
  Using PIVOT and UNPIVOT - SQL Server | Microsoft Docs
  https://help.tableau.com/current/guides/get-started-tutorial/en-us/get-started-tutorial-build.htm
  https://community.tableau.com/s/question/0D54T00000C67AcSAJ/how-to-access-sub-reportsas-a-separate-sheet-from-parent-report-summarise-report-to-detail-report
  How to use Sheet as a Filter in Tableau - Analytics Tuts (analytics-tuts.com)

Laboratory Work No. 6

Assessment for Laboratory Work 4 and Laboratory Work 5.

Laboratory Work No. 7

1. Export the Formula 1 World Championship database from the Kaggle website, address https://www.kaggle.com/rohanrao/formula-1-world-championship-1950-2020?select=status.csv in an Excel file. Then, import this database in Microsoft SQL Server, using the facility from the context menu Tasks/Import Data. Convert all the primary key, respectively foreign key fileds, to the *integer* type. Build the database diagram and study the table structure, respectively the connections between the tables.

2. Extend the structure of the Formula 1 World Championship database by implementing the concept of *class hierarchy*. Add tables that correspond to some subclasses of the class *Drivers, that* have specific attributes, as follows:
    - Begginers – education_level, *varchar (30)*; number of races in which he participated, *int*
    - Experienced_drivers – number of years of experience *int*; number of races won, *int*

3. Create, within the Formula 1 database, the following Transact-SQL functions, in order to return a single value (*scalar functions*):
    a. The age in years for the driver whose code is provided as parameter.
    b. The code of the driver who obtained the maximum number of points during the year provided as parameter.
    c. The year in which the driver whose code is given as parameter obtained the minimum number of points.
    d. The average score achieved by the participating drivers in the race whose name is given as parameter.
    e. The average score obtained by the constructor whose name is provided as parameter.

4. Create, within the Formula 1 database, the following Transact-SQL functions, in order to obtain a record set (*table-value functions*):
    a. Name, surname, date of birth, nationality and code for those drivers that took part in competitions during the year provided as parameter and obtained a score greater than a threshold, given as parameter.
    b. Name, nationality and URL for those constructors who obtained a score greater than the average score of the constructors to the race

whose id is provided as parameter.

c. Name, surname and nationality, as well as the name of the constructor for those drivers who participated in the maximum number of laps to the race whose identifier is provided as parameter.
d. Year, URL, the season and the number of races per year in which the driver whose code is given through a parameter was involved. If the specified driver didn't participate at races in a certain year, the ‚no races' string should be displayed.
e. Name, location, country and number of races for those circuits taking place at a latitude greater than a certain limit, given as parameter.

- **Documentation:**
  https://docs.microsoft.com/en-us/sql/integration-services/import-export-data/start-the-sql-server-import-and-export-wizard?view=sql-server-2017

  https://dba.stackexchange.com/questions/16543/supertype-subtype-deciding-between-category-complete-disjoint-or-incomplete-ove

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

Laboratory Work No. 8

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. 9

1. **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, in order 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.

2. **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 in order 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.

- Documentation:

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-ver15
https://www.sqlservertutorial.net/sql-server-basics/sql-server-recursive-cte/

Laboratory Work No. 10

1. **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.

2. **Assessment of Laboratory Work 7, Laboratory Work 8, Laboratory Work 9**

Laboratory Work No. 11

1. 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 .

2. 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*.

3. 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.

4. 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

Laboratory Work No. 14

- **Assessment for Laboratory Works 11-13**