Assigment 2

An EERD for assignment 1

(page 2)

Mapping EERD to a relational database schema

Company (Cnumber, name, address, phone, Edate)

Person (SSN, Fname, Lname, address, phone)

Trainee (SSN, DoB, photo, company_ID)

MC (SSN)

Mentor (SSN)

Song (number, released_year, name, singer_SSN_fist_performed)

ThemeSong (song_ID)

SongComposedBy (song_ID, composer_SSN)

Singer (SSN, guest_ID)

SingerSignatureSong (SSN, song_name)

Producer (SSN)

ProducerProgram (SSN, Program_name)

SongWriter (SSN)

Season (year, location, themesongID, MC_SSN)

SeasonMentor (year, SSN_mentor)

SeasonTrainee (year, SSN_trainee)

MentorEvaluateTrainee (year, SSN_trainee, SSN_mentor, score)

Episode (year, No, name, datetime, duration)

Stage (year, ep_No, stage_No, is_group, skill, total vote, song_ID)

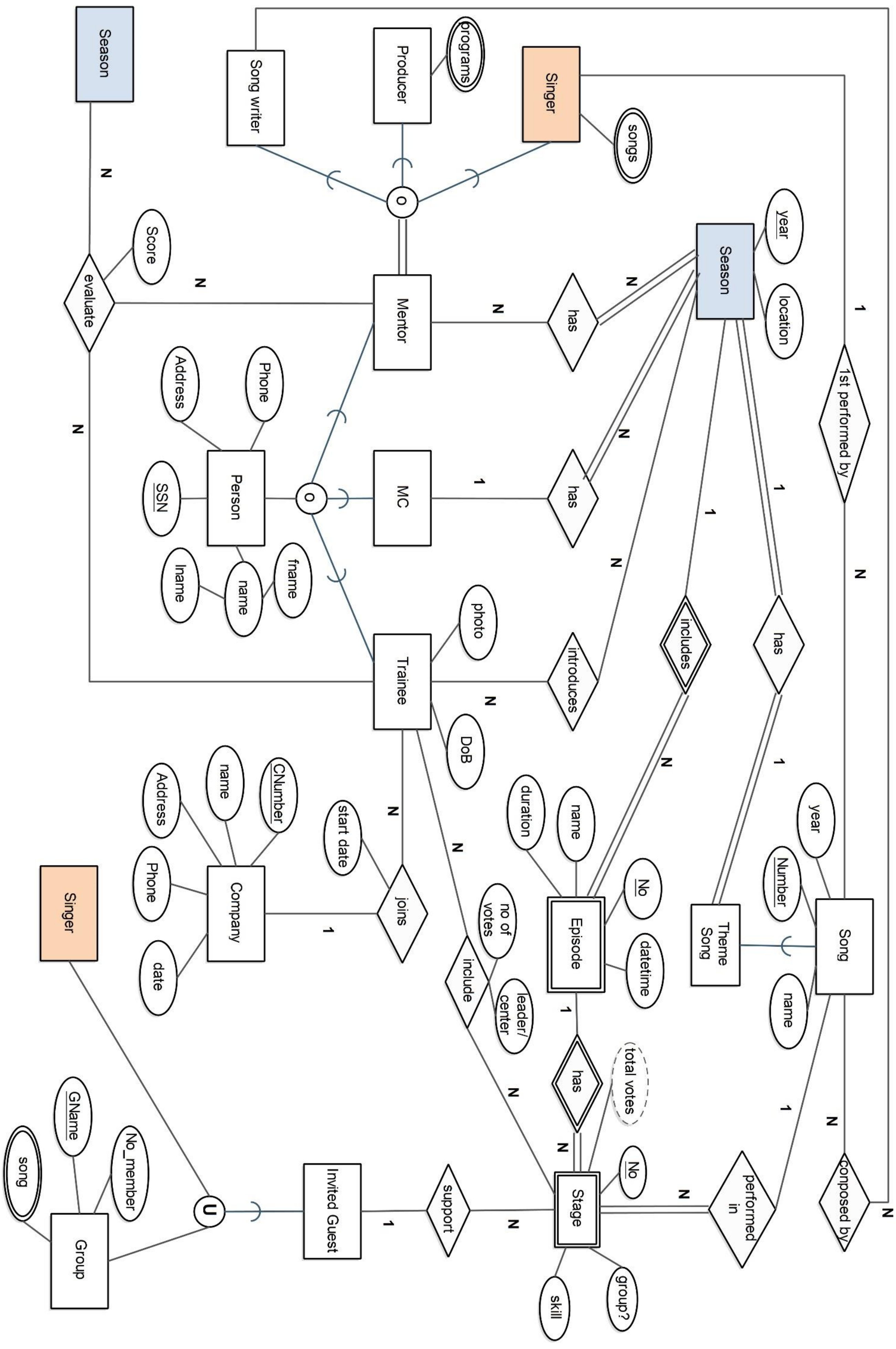
StageIncludeTrainee (year, ep_No, stage_No, SSN_trainee, role, no_of_votes)

InvitedGuest (guest_ID)

Group (Gname, no_of_member, guest_ID)

GroupSignatureSong (Gname, song_name)

GuestSupportStage (guest_ID, year, ep_No, stage_No)


Season
Song writer
Producer
programs
Singer
songs
o
Season
year
location
1st performed by
Score
evaluate
Mentor
has
Address
Phone
Person
SSN
o
MC
has
fname
lname
name
Trainee
photo
DoB
introduces
includes
has
Address
name
CNumber
Phone
Company
date
start date
joins
Singer
duration
name
No
Episode
datetime
no of votes
include
leader/ center
Theme Song
year
Number
Song
name
has
total votes
No
Stage
performed in
composed by
group?
skill
GName
No_member
song
U
Group
Invited Guest
support
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
N
1
1
1
1
1
1
1
1
1

PART 1: CREATE DATABASE (4 points)

I. Create table (2.5 points)

1. **Company (<u>Cnumber</u>, name, address, phone, Edate)**
   - Cnumber: C[0-9][0-9][0-9]
   - Name: not null
   - Phone: unique and not null
   - Edate: datetime (For example: 30/01/1990)
2. **Person (<u>SSN</u>, Fname, Lname, address, phone)**
   - SSN: 12-digit numbers
   - Phone: unique and not null
3. **Trainee (<u>SSN</u>, DoB, photo, company_ID)**
   - SSN: 12-digit numbers, foreign key, references to SSN of Person
   - DoB: datetime (For example: 30/01/1990)
   - Photo: url
   - Company_ID: foreign key, references to Cnumber of Company
4. **MC (<u>SSN</u>)**
   - SSN: 12-digit numbers, foreign key, references to SSN of Person
5. **Mentor (<u>SSN</u>)**
   - SSN: 12-digit numbers, foreign key, references to SSN of Person
6. **Song (<u>number</u>, released_year, name, singer_SSN_fist_performed)**
   - number: S[auto increment integer]
   - singer_SSN_fist_performed: foreign key, references to SSN of Singer
7. **ThemeSong (<u>song_ID</u>)**
   - song_ID: foreign key, references to number of Song
8. **SongComposedBy (<u>song_ID, composer_SSN</u>)**
   - song_ID: foreign key, references to number of Song
   - composer_SSN: foreign key, references to SSN of SongWriter
9. **Singer (<u>SSN</u>, guest_ID)**
   - SSN: 12-digit numbers, foreign key, references to SSN of Mentor
   - Guest_ID: foreign key, references to guest_ID of InvitedGuest
10. **SingerSignatureSong (<u>SSN, song_name</u>)**
    - SSN: foreign key, references to SSN of Singer
11. **Producer (<u>SSN</u>)**
    - SSN: 12-digit numbers, foreign key, references to SSN of Mentor
12. **ProducerProgram (<u>SSN, Program_name</u>)**
    - SSN: foreign key, references to SSN of Producer

13. **SongWriter (SSN)**
    - SSN: 12-digit numbers, foreign key, references to SSN of Mentor
14. **Season (year, location, themesong_ID, MC_SSN)**
    - Year: valid year (For example: 2021, 2022)
    - Themesong_ID: foreign key, references to Number of ThemeSong
    - SSN: foreign key, references to SSN of MC
15. **SeasonMentor (year, SSN_mentor)**
    - Year: foreign key, references to year of Season
    - SSN_mentor: references to SSN of Mentor
16. **SeasonTrainee (year, SSN_trainee)**
    - Year: foreign key, references to year of Season
    - SSN_trainee: foreign key, references to SSN of Trainee
17. **MentorValuateTrainee (year, SSN_trainee, SSN_mentor, score)**
    - Year: foreign key, references to year of Season
    - SSN_trainee: foreign key, references to SSN of Trainee
    - SSN_mentor: references to SSN of Mentor
    - Score: integer [0, 100]
18. **Episode (year, No, name, datetime, duration)**
    - Year: foreign key, references to year of Season
    - No: integer [1, 5]
    - Datetime: for example: 30/01/2022 19:00
    - Duration: in minutes
19. **Stage (year, ep_No, stage_No, is_group, skill, total vote, song_ID)**
    - (Year, ep_No): foreign key, references to (year, No) of Episode
    - Is_group: Boolean. Yes: group stage, No: individual stage
    - Skill: 1 – vocal, 2 – rap, 3 – dance, 4 – mixed (default value)
    - Total vote: derived attribute
    - Song_ID: foreign key, references to number of Song
20. **StageIncludeTrainee (year, ep_No, stage_No, SSN_trainee, role, no_of_votes)**
    - (Year, ep_No, stage_No): foreign key, references to (year, ep_No, stage_No) of Stage
    - Role: 1 – member (default), 2 – leader, 3 – center
    - no_of_votes: integer [0, 500]
    - Note: if ep_No = 5 (final episode), one trainee has 2 stages. The number of votes of trainees in the final episode will be stored in individual stage. For example, the result of Trainee ‘123456789045’ in the final episode of season 2022 as following:
      (2022, 5, 1, ‘123456789045’, 2, null) -- group statge, no_of_votes is null

(2022, 5, 5, ‘123456789045’, 2, 100) – individual stage, no_of_votes of this trainee in final stage is 100 votes

21. **InvitedGuest (<u>guest_ID</u>)**
    - Guest_ID: auto increment integer
22. **Group (<u>Gname</u>, no_of_member, guest_ID)**
    - no_of_member: integer number in [1, 20]
    - guest_ID: foreign key, references to guest_ID of InvitedGuest
23. **GroupSignatureSong (<u>Gname, song_name</u>)**
    - Gname: foreign key, references to Gname of Group
24. **GuestSupportStage (guest_ID, <u>year, ep_No, stage_No</u>)**
    - guest_ID: foreign key, references to guest_ID of InvitedGuest
    - (Year, ep_No, stage_No): foreign key, references to (year, ep_No, stage_No) of Stage

***<u>Note</u>**: Student should indentify other datatypes and constraints, such as primary key, foreign key, unique, not null, etc. (base on the business description in Assignment 1, EERD and above relational database schema).*

II. Insert (1.5 points)

Insert data for all tables in the database.

Requirements: The data in the tables must be meaningful, and each table has at least 4 rows (except table 14 – at least 2 rows).

NOTE: Each group need to create a script for part 1. This script can excute without any error to create a complete database.

PART 2: STORE PROCEDURE, FUNCTION, TRIGGER (3 points)

I. Trigger (2 points)

1. Write trigger to check following constrains:
    a. A trainee can participate 3 seasons at most (**0.5 point**)
    b. If a trainee success to join in a debut night, then he/she cannot register as a trainee of later seasons (**0.5 point**)
2. Write trigger to calculate the total number of votes of a group (derived attribute of table 19) when number of votes of a member (no_of_votes of table 20) is updated (**0.5 point**)
3. Write trigger to ensure that a trainee has at most:
    - one group stage in episode 2, 3, 4 and (**0.25 point**)
    - one group stage and one individual stage in episode 5 (**0.25 point**)

II. Store Procedure/Function (1 point)

1. Create a procedure/function to print list of trainees come to the next episode (or debut if input episode is debut night) ***(0.5 point).*** (Specifying list of trainees can come to next round (or debut) is specified in Asssignment 1)

   Input: year, episode

   Output: list of <ssn, num of votes/ average score>, sorted by number of votes/ average score

ssn	Num of votes/ avg score
123456789012	300
987654321012	250
…	…
765499992222	190

***Note**: When this procedure/function is excuted for episode N, the data for episode N+1 (such as stage, number of votes…) has not been inserted yet.*

2. Create a procedure/function to retrieve the result of a trainee in a season ***(0.5 point)***

   Input: SSN, year

   Output: < episode, Num of votes/ average score > sorted by episode

Episode	Num of votes/ avg score
1	80
2	300
3	250
4	100
5	null

PART 3: BUILDING APPLICATIONS (3 points)

Build an application with the following requirements:

- *Programming environment: optional (desktop application or web application).*
- *Programming language: optional.*
- *The application connects to the database created in Part 1 and Part 2.*
- *Display the data on the form and perform the requirement below. (Do not need to impletement all the desriptions in Assignment 1)*
- *Students need to prepare data, scripts to demo the application when reporting.*

I. Create user (0.5 point)

Log in to the database with DBA privileges, create a user named *sManager* and assign all access rights to this user.

II. Impletement features (2.5 points)

1. Log in, log out (enter the user name/password for *sManager* account to log in/out). (**0.5 point**)
2. Log in to the user *sManager* and do the following:
   a. Add information for a new trainee. **(0.5 point)**
   b. Search information of trainees by name (fullname of trainee includes the search string). Note that searching is case insensitivity. (**0.5 point**)
   c. Select one trainee in search result of (b) to view information. Information includes SSN, name, phone, address, photo, number of seasons participating, best achievement (last episode joining in a season) (**0.5 point)**
   d. Retrive result of a trainee in a season (must call a store procedure/ function of Part 2). **(0.5 point)**

III. Bonus (1 point)

- User interface is attractive and friendly. **(0.5 point)**
- Building a data model (in mvc architecture) to communicate with database. **(0.5 point)**

***Note**: Each student must take part in all parts of assignment 2:*

- *Part 1.1: Create table*
- *Part 1.2: Insert data.*
- *Part 2: Store procedure, function, trigger*
- *Part 3: Building application*

------ HAPPY IMPLEMENTATION -------