Database Assignment 2

Saeth Wannasuphoprasit
Student ID: 201585689

Question 1

Background

Cloud computing is changing the world. It is Internet-based computing: hundreds of thousands of remote computers are interconnected to provide all kinds of resouces, including data, applications, as well as computing power. For example, Google provides online services named *Google Docs*[1], which we may consider as an alternative to Microsoft Office. Google Docs differs from conventional software products in the sense that, when using Google Docs, the data and the software locate no longer in the our own machines but rather somewhere "in the cloud." We need not store the data or install any software. What we have to do is to get a Google Docs account and then simply open an Internet browser. This is also the situation when we rent online movies on Amazon[2]. We pay about 4 dollars to enjoy a movie online. We do not store the movie in our own machines, nor do we install any movie players. We need only get an Amazon account to do the business.

One of the key observations from the above two examples is that machines become virtual to users in the cloud era. When we are editing online documents or watching online movies, we are also using certain computing power provided by Google or Amazon.

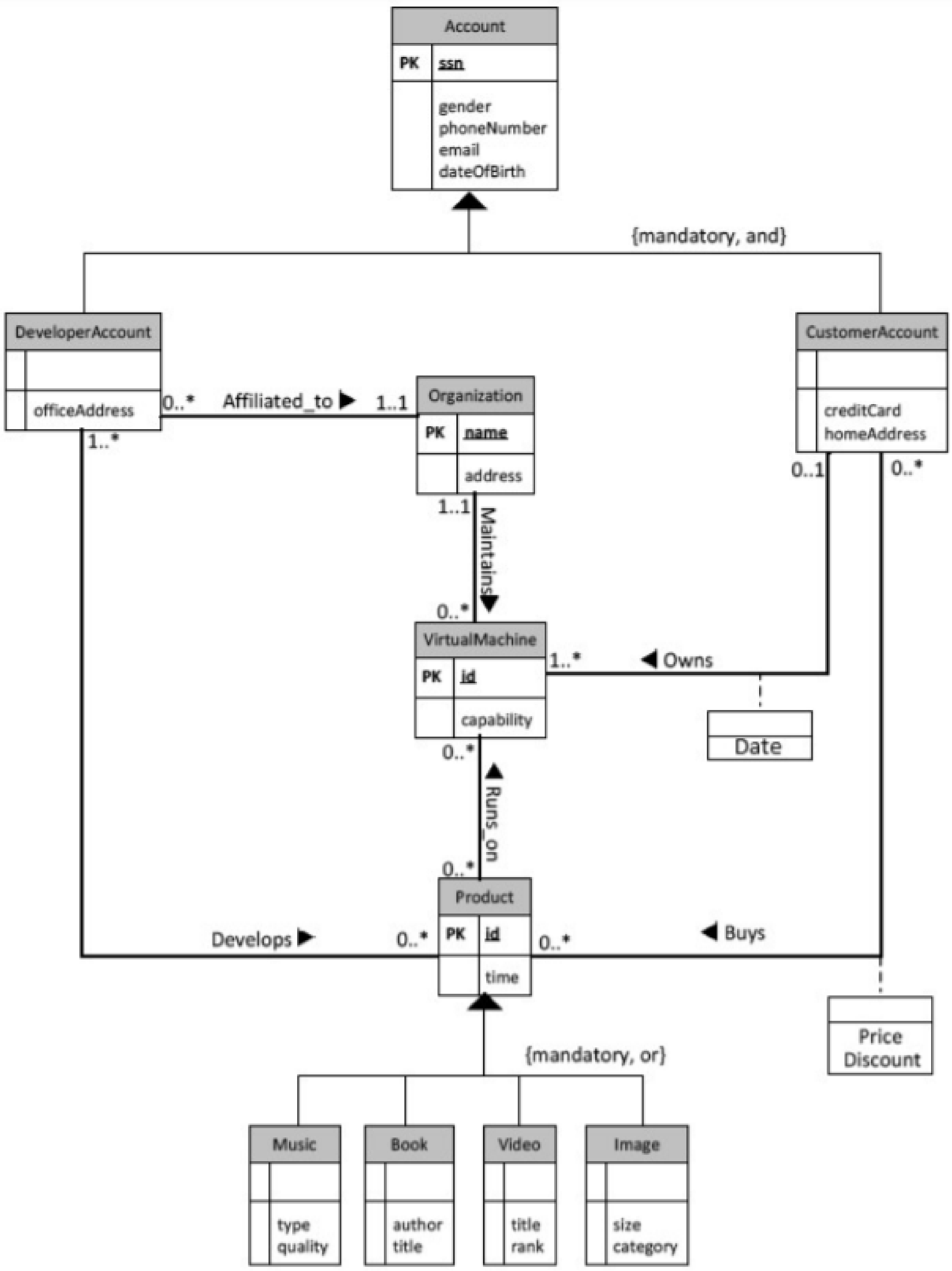
Task

You are given as an input, the Enhanced Entity-Relationship diagram of Figure 1, that is meant to be one possible conceptual representation of the users' requirements. You are required to create the **logical data model**.

1. **(45 marks)** Create the appropriate **relations** to represent the **entities** and **relationships** of the E-R diagram.
2. **(15 marks)** Designate the **primary key**, and any **alternate** or **foreign keys** of each relation together with their **references**.

For all the above, use the methodology covered in the lectures and described in Chapter 17 of [1], providing clear annotations and explanations of all the steps. In case you deviated from the general guidelines, explain in detail why you did so. Also discuss in detail any other possible alternatives you may have considered (if any), and why you decided not to choose them.

[1]See https://docs.google.com.
[2]See http://www.amazon.com.


Account
PK ssn
gender
phoneNumber
email
dateOfBirth
{mandatory, and}
DeveloperAccount
officeAddress
CustomerAccount
creditCard
homeAddress
Organization
PK name
address
0..* Affiliated_to ▶ 1..1
1..*
0..1
0..*
1..1
Maintains
0..*
VirtualMachine
PK id
capability
1..* ◀ Owns
Date
0..*
Runs_on
0..*
Product
PK id
time
Develops ▶
0..*
0..*
◀ Buys
Price
Discount
{mandatory, or}
Music
type
quality
Book
author
title
Video
title
rank
Image
size
category

1. create the relations

(1) Strong Entities

Account(ssn, gender, phoneNumber, email, dateOfBirth)
Organization(name, address)
VirtualMachine(virtualMachineId, capability)
Product(productId, time)

Note that, the names of Id in VirtualMachine and Product are changed to virtualMachineId and productId respectively in order not to cause confusion.

(2) Weak Entities

No weak entity.

(3) one-to-many (1:*) binary relationship types;

Affiliated_to: We will deal with them later since these involve subclasses that are not yet represented.
Maintains: VirtualMachine(virtualMachineId, capability, name)
    name references Organization(name)
Owns: We will deal with them later since these involve subclasses that are not yet represented.

Overall we get:
Account(ssn, gender, phoneNumber, email, dateOfBirth)
Organization(name, address)
VirtualMachine(virtualMachineId, capability, name)
    name references Organization(name)
Product(productId, time)

(4) one-to-one (1:1) binary relationship types;

No one-to-one (1:1) binary relationship types.

(5) one-to-one (1:1) recursive relationship types;

No such relationship.

(6) superclass/subclass relationship types;

6.1 Account consists of DeveloperAccount and CustomerAccount {mandatory, and}
In this case, according to the guideline, we will create all three in a single relation called AllAccount as follow.

AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag)
Note that, developerAccountFlag and customerAccountFlag are discriminators to identify account types.

6.2 Product consists of Music, Book, Video, and Image {mandatory, or}
In this case, according to the guideline, we will create four relations called ProductMusic, ProductBook, ProductVideo, ProductImage as follow.

ProductMusic(productId, time, type, quality)
ProductBook(productId, time, author, BookTitle)
ProductVideo(productId, time, VideoTitle, rank)
ProductImage(productId, time, size, category)

Note that, the names of titles in Book and Video are changed to BookTitle and VideoTitle respectively in order not to cause confusion.

Overall we get:
AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag)
Organization(name, address)
VirtualMachine(virtualMachineId, capability, name)
        name references Organization(name)
ProductMusic(productId, time, type, quality)
ProductBook(productId, time, author, BookTitle)
ProductVideo(productId, time, VideoTitle, rank)
ProductImage(productId, time, size, category)

back to (3), one-to-many (1:*) binary relationship types;
Affiliated_to: AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)
        name references Organization(name)
Owns: VirtualMachine(virtualMachineId, capability, name, ssn, Date)
        name references Organization(name)
        ssn references AllAccount(ssn)

Overall we get:
AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)
        name references Organization(name)
Organization(name, address)
VirtualMachine(virtualMachineId, capability, name, ssn, Date)
        name references Organization(name)
        ssn references AllAccount(ssn)
ProductMusic(productId, time, type, quality)
ProductBook(productId, time, author, BookTitle)
ProductVideo(productId, time, VideoTitle, rank)
ProductImage(productId, time, size, category)

(7) many-to-many (*:*) binary relationship types;

There are three relationships in this case. Three new relations are created namely RunsOn, Develops, and Buys to map between entities that link through them.
Runson: RunsOn(virtualMachineId, productId)
        virtualMachineId references VirtualMachine(virtualMachineId)
        productId references List of all products(productId)
Develops: Developes(ssn, productId)
        ssn references AllAccount(ssn)
        productId references List of all products(productId)
Buys: Buys(ssn, productId, Price, Discount)
        ssn references AllAccount(ssn)
        productId references List of all products(productId)

Note that, this may cause referential integrity because there is no relation that record all productId. So, in order to handle this, we can create a new relation called Product which lists all productId and recreate ProductMusic, ProductBook, ProductVideo, and ProductImage as follow.

Product(productId, time)
Music(productId, type, quality)
  productId references Product(productId)
Book(productId, author, BookTitle)
  productId references Product(productId)
Video(productId, VideoTitle, rank)
  productId references Product(productId)
Image(productId, size, category)
  productId references Product(productId)

And then the previous relations (RunsOn, Develops, and Buys) can be derived as follow.

RunsOn(virtualMachineId, productId)
  virtualMachineId references VirtualMachine(virtualMachineId)
  productId references Product(productId)
Developes(ssn, productId)
  ssn references AllAccount(ssn)
  productId references Product(productId)
Buys(ssn, productId, Price, Discount)
  ssn references AllAccount(ssn)
  productId references Product(productId)

Overall we get:
AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)
  name references Organization(name)
Organization(name, address)
VirtualMachine(virtualMachineId, capability, name, ssn, Date)
  name references Organization(name)
  ssn references AllAccount(ssn)
Product(productId, time)
Music(productId, type, quality)
  productId references Product(productId)
Book(productId, author, BookTitle)
  productId references Product(productId)
Video(productId, VideoTitle, rank)
  productId references Product(productId)
Image(productId, size, category)
  productId references Product(productId)
RunsOn(virtualMachineId, productId)
  virtualMachineId references VirtualMachine(virtualMachineId)
  productId references Product(productId)
Developes(ssn, productId)
  ssn references AllAccount(ssn)
  productId references Product(productId)
Buys(ssn, productId, Price, Discount)
  ssn references AllAccount(ssn)
  productId references Product(productId)

(8) complex relationship types;

No such relationship.

(9) multi-valued attributes.

No such relationship.

Final Relations (not including primary keys, alternate keys, and foreign keys):

AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)
    name references Organization(name)

Organization(name, address)

VirtualMachine(virtualMachineId, capability, name, ssn, Date)
    name references Organization(name)
    ssn references AllAccount(ssn)

Product(productId, time)

Music(productId, type, quality)
    productId references Product(productId)

Book(productId, author, BookTitle)
    productId references Product(productId)

Video(productId, VideoTitle, rank)
    productId references Product(productId)

Image(productId, size, category)
    productId references Product(productId)

RunsOn(virtualMachineId, productId)
    virtualMachineId references VirtualMachine(virtualMachineId)
    productId references Product(productId)

Developes(ssn, productId)
    ssn references AllAccount(ssn)
    productId references Product(productId)

Buys(ssn, productId, Price, Discount)
    ssn references AllAccount(ssn)
    productId references Product(productId)

2. Design the primary keys, alternate keys, foreign keys, and references

AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)
    Primary Key ssn
    Alternate Key email, phoneNumber, creditCard
    Foreign Key name references Organization(name)

Assumption:

1. ssn uniquely identifies each account. So, we can use it as a main primary key.
2. email, phoneNumber, and creditCard are unique for each user account. So, we can use each of it as an alternate key. In addition, email is unlikely to change followed by phoneNumber and creditCard. Thus, we should prioritize email first and then phoneNumber and creditCard respectively.

Organization(name, address)
    Primary Key name

Assumption:

1. name uniquely identifies each Organization. So, we can use it as a main primary key.
2. address can be the same with some Organization. So, we can not use it as an alternative key.

VirtualMachine(virtualMachineId, capability, name, ssn, Date)

Primary Key virtualMachineId

Foreign Key name references Organization(name)

Foreign Key ssn references AllAccount(ssn)

Assumption:

1. virtualMachineId uniquely identifies each VirtualMachine. So, we can use it as a main primary key.
2. There may be the same capability and Date in some VirtualMachine. So, we can not use each of it as an alternate key.

Product(productId, time)

Primary Key productId

Assumption:

1. productId uniquely identifies each Product. So, we can use it as a main primary key.
2. Some Product may have the same time. So, we can not use it as an alternate key.

Music(productId, type, quality)

Primary Key productId

Foreign Key productId references Product(productId)

Assumption:

1. productId uniquely identifies each Music. So, we can use it as a main primary key.
2. Some Music may have the same type and quality. So, we can not use them as an alternate key.

Book(productId, author, BookTitle)

Primary Key productId

Alternate Key BookTitle and author

Foreign Key productId references Product(productId)

Assumption:

1. productId uniquely identifies each Book. So, we can use it as a main primary key.
2. BookTitle and author uniquely identify each Book. So, we can use them together as an alternate key.

Video(productId, VideoTitle, rank)

Primary Key productId

Alternate Key VideoTitle and rank

Foreign Key productId references Product(productId)

Assumption:

1. productId uniquely identifies each Video. So, we can use it as a main primary key.
2. VideoTitle and rank uniquely identify each Video. So, we can use them together as an alternate key.

Image(productId, size, category)

Primary Key productId

Foreign Key productId references Product(productId)

Assumption:

1. productId uniquely identifies each Image. So, we can use it as a main primary key.
2. Some Image may have the same size and category. So, we can not use them as an alternate key.

RunsOn(virtualMachineId, productId)

  Primary Key virtualMachineId, productId
  Foreign Key virtualMachineId references VirtualMachine(virtualMachineId)
  Foreign Key productId references Product(productId)

Assumption:

1. virtualMachineId and productId uniquely identify each RunsOn relation. So, we can use them together as a composite primary key.

Developes(ssn, productId)

  Primary Key ssn, productId
  Foreign Key ssn references AllAccount(ssn)
  Foreign Key productId references Product(productId)

Assumption:

1. ssn and productId uniquely identify each Develops relation. So, we can use them together as a composite primary key.

Buys(ssn, productId, Price, Discount)

  Primary Key ssn, productId
  Foreign Key ssn references AllAccount(ssn)
  Foreign Key productId references Product(productId)

Assumption:

1. ssn and productId uniquely identify each Buys relation. So, we can use them together as a composite primary key.
2. Some items in Buys relation may have the same Price and Discount. So, we can not use them as an alternate key.

Overall, we get.

AllAccount(ssn, gender, phoneNumber, email, dateOfBirth, officeAddress, creditCard, homeAddress, developerAccountFlag, customerAccountFlag, name)

Primary Key ssn
Alternate Key email, phoneNumber, creditCard
Foreign Key name references Organization(name)

Organization(name, address)

Primary Key name

VirtualMachine(virtualMachineId, capability, name, ssn, Date)

Primary Key virtualMachineId
Foreign Key name references Organization(name)
Foreign Key ssn references AllAccount(ssn)

Product(productId, time)

Primary Key productId

Music(productId, type, quality)

Primary Key productId
Foreign Key productId references Product(productId)

Book(productId, author, BookTitle)

Primary Key productId
Alternate Key BookTitle and author
Foreign Key productId references Product(productId)

Video(productId, VideoTitle, rank)

Primary Key productId
Alternate Key VideoTitle and rank
Foreign Key productId references Product(productId)

Image(productId, size, catagory)

Primary Key productId
Foreign Key productId references Product(productId)

RunsOn(virtualMachineId, productId)

Primary Key virtualMachineId, productId
Foreign Key virtualMachineId references VirtualMachine(virtualMachineId)
Foreign Key productId references Product(productId)

Developes(ssn, productId)

Primary Key ssn, productId
Foreign Key ssn references AllAccount(ssn)
Foreign Key productId references Product(productId)

Buys(ssn, productId, Price, Discount)

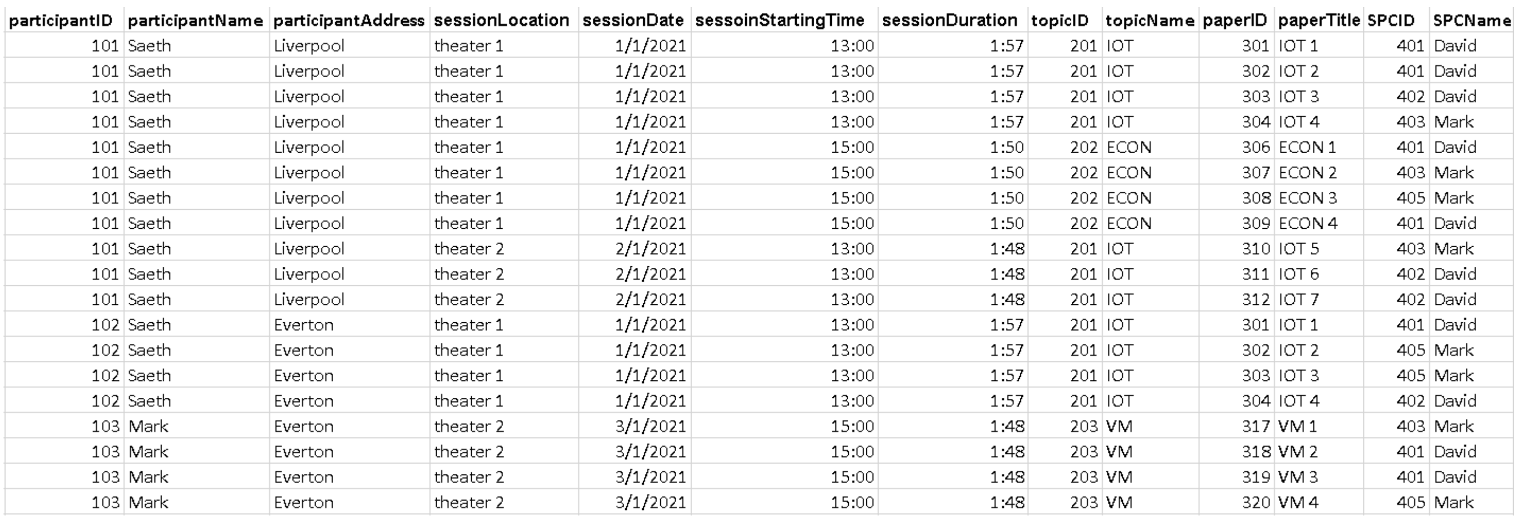
Primary Key ssn, productId
Foreign Key ssn references AllAccount(ssn)
Foreign Key productId references Product(productId)

Question 2

```
ConferenceData (participantID, participantName, participantAddress, sessionLocation,
sessionDate, sessionStartingTime, sessionDuration, topicID, topicName, paperID,
paperTitle, SPCID, SPCName).
```

1. UNF to 1NF: assume flattening method

The picture below illustrates one possibility of the ConferenceData table using the flattening method.



participantID	participantName	participantAddress	sessionLocation	sessionDate	sessoinStartingTime	sessionDuration	topicID	topicName	paperID	paperTitle	SPCID	SPCName
101	Saeth	Liverpool	theater 1	1/1/2021	13:00	1:57	201	IOT	301	IOT 1	401	David
101	Saeth	Liverpool	theater 1	1/1/2021	13:00	1:57	201	IOT	302	IOT 2	401	David
101	Saeth	Liverpool	theater 1	1/1/2021	13:00	1:57	201	IOT	303	IOT 3	402	David
101	Saeth	Liverpool	theater 1	1/1/2021	13:00	1:57	201	IOT	304	IOT 4	403	Mark
101	Saeth	Liverpool	theater 1	1/1/2021	15:00	1:50	202	ECON	306	ECON 1	401	David
101	Saeth	Liverpool	theater 1	1/1/2021	15:00	1:50	202	ECON	307	ECON 2	403	Mark
101	Saeth	Liverpool	theater 1	1/1/2021	15:00	1:50	202	ECON	308	ECON 3	405	Mark
101	Saeth	Liverpool	theater 1	1/1/2021	15:00	1:50	202	ECON	309	ECON 4	401	David
101	Saeth	Liverpool	theater 2	2/1/2021	13:00	1:48	201	IOT	310	IOT 5	403	Mark
101	Saeth	Liverpool	theater 2	2/1/2021	13:00	1:48	201	IOT	311	IOT 6	402	David
101	Saeth	Liverpool	theater 2	2/1/2021	13:00	1:48	201	IOT	312	IOT 7	402	David
102	Saeth	Everton	theater 1	1/1/2021	13:00	1:57	201	IOT	301	IOT 1	401	David
102	Saeth	Everton	theater 1	1/1/2021	13:00	1:57	201	IOT	302	IOT 2	405	Mark
102	Saeth	Everton	theater 1	1/1/2021	13:00	1:57	201	IOT	303	IOT 3	405	Mark
102	Saeth	Everton	theater 1	1/1/2021	13:00	1:57	201	IOT	304	IOT 4	402	David
103	Mark	Everton	theater 2	3/1/2021	15:00	1:48	203	VM	317	VM 1	403	Mark
103	Mark	Everton	theater 2	3/1/2021	15:00	1:48	203	VM	318	VM 2	401	David
103	Mark	Everton	theater 2	3/1/2021	15:00	1:48	203	VM	319	VM 3	401	David
103	Mark	Everton	theater 2	3/1/2021	15:00	1:48	203	VM	320	VM 4	405	Mark

In this case:

- {participantID(Name, Address)} = {101(Saeth, Liverpool), 102(Saeth, Everton), 103(Mark, Everton)}
- {(Location, Date, StartingTime, Duratoin, Topic)} = {(theater 1, 1/1/2021, 13:00, 1:57, IOT), (theater 1, 1/1/2021, 15:00, 1:50, Econ), (theater 2, 2/1/2021, 13:00, 1:48, IOT), (theater 2, 3/1/2021, 15:00, 1:48, VM)}
- Each topic has several papers
- Each paper has 1 SPC

2. identify functional dependencies

2.1 first of all, list all criteria.

2.1.1 each topic can have more than 1 paper/ each paper can only belong to 1 topic

2.1.2 each topic has more than 1 session (1 session per day)/ each session uniquely belongs to 1 topic

2.1.3 each session has more than 1 paper (according to the topic in the session)/ each paper can only belong to 1 session.

2.1.4 each topic can belong to more than 1 session location/ each session location can have more than 1 topic.

2.1.5 each SPC can review more than 1 paper/ each paper can be reviewed by only 1 SPC (Assuming that it is not necessary for each paper to be reviewed by more than 1 SPC. So, only 1 SPC is enough)

2.1.6 each participant can participate in more than 1 session/ each session can be participated by more than 1 participant.

2.2 According to the criteria, we can define functional dependencies with only one attribute on the left side as follow.

2.2.1 participantID -> participantName, participantAddress
(each participantID defines particapantName and participantAddress)

2.2.2 participantName -> no functional dependencies
(It can be some names of participants who can not identify other attributes).

2.2.3 participantAddress -> no functional dependencies
(It can be some Addresses that can not identify other attributes).

2.2.4 sessionLocatoin -> no functional dependencies
(It can be some session locations that can not identify other attributes).

2.2.5 sessionDate -> no functional dependencies.
For example, session date 24 December 2020 can not identify any other attributes.

2.2.6 sessionStartingTime -> no functional dependencies.
For example, the session starting time at 13.00 can not identify any other attributes.

2.2.7 sessionDuration -> no functional dependencies.
For example, session duration time 1:58 can not identify any other attributes.

2.2.8 topicID -> topicName
(each topicID uniquely defines topicName)

2.2.9 topicName -> topicID
(each topic ID is uniquely identified by topic name)

2.2.10 paperID -> paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName
(each paperID uniquely defines paperTitle)
(each paperID defines sessionLocation, sessionDate, sessionStartingTime, sessionDuration, topicID, topicName, SPCID, and SPCName)

2.2.11 paperTitle -> paperID, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName
(each paperTitle uniquely defines paperID)
(each paperTitle defines sessionLocation, sessionDate, sessionStartingTime, sessionDuration, topicID, topicName, SPCID, and SPCName)

2.2.12 SPCID -> SPCName
(SPCID defines SPCName)

2.2.13 SPCName -> no functional dependencies.
For example, it can be some names of SPC that can not identify other attributes.

according to the functional dependencies described above, we can summarize as follow.

participantID -> participantName, participantAddress

topicID -> topicName

topicName -> topicID

paperID -> paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName

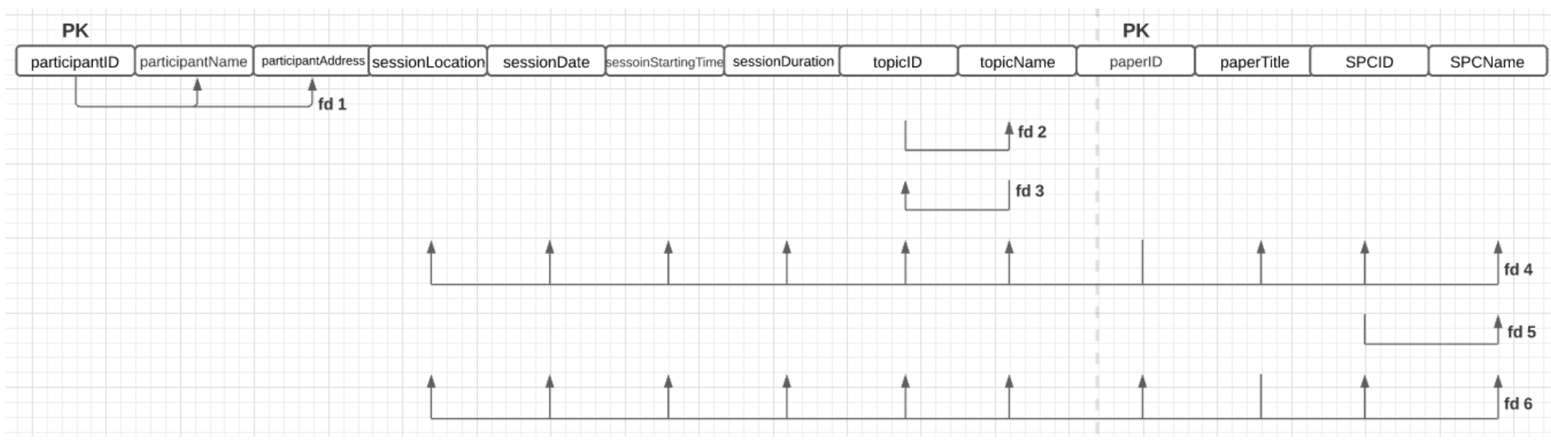
paperTitle -> paperID, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName

SPCID -> SPCName

We don’t need to define more dependencies now because it can be clearly seen that either (participantID with paperID) or (participantID with paperTitle) can be used as the primary key.
However, paperID is unlikely to change. So, participantID and paperID is the primary key.

Primary key: participantID, paperID -> nothing new
Candidate key: participantID, paperTitle -> nothing new

1NF



1NF:
ConferenceData(participantID, paperID, participantName, participantAddress, sessionLocation, sessionDate, sessionStartingTime, sessionDuration, topicID, topicName, paperTitle, SPCID, SPCName)

3. 1NF to 2NF

remove partially dependent on the primary key.

primary key:
participantID, paperID -> nothing new

partially dependent on the primary key:
participantID -> participantName, participantAddress

paperID -> paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName
So, we can decompose from 1NF into 2NF as follow.

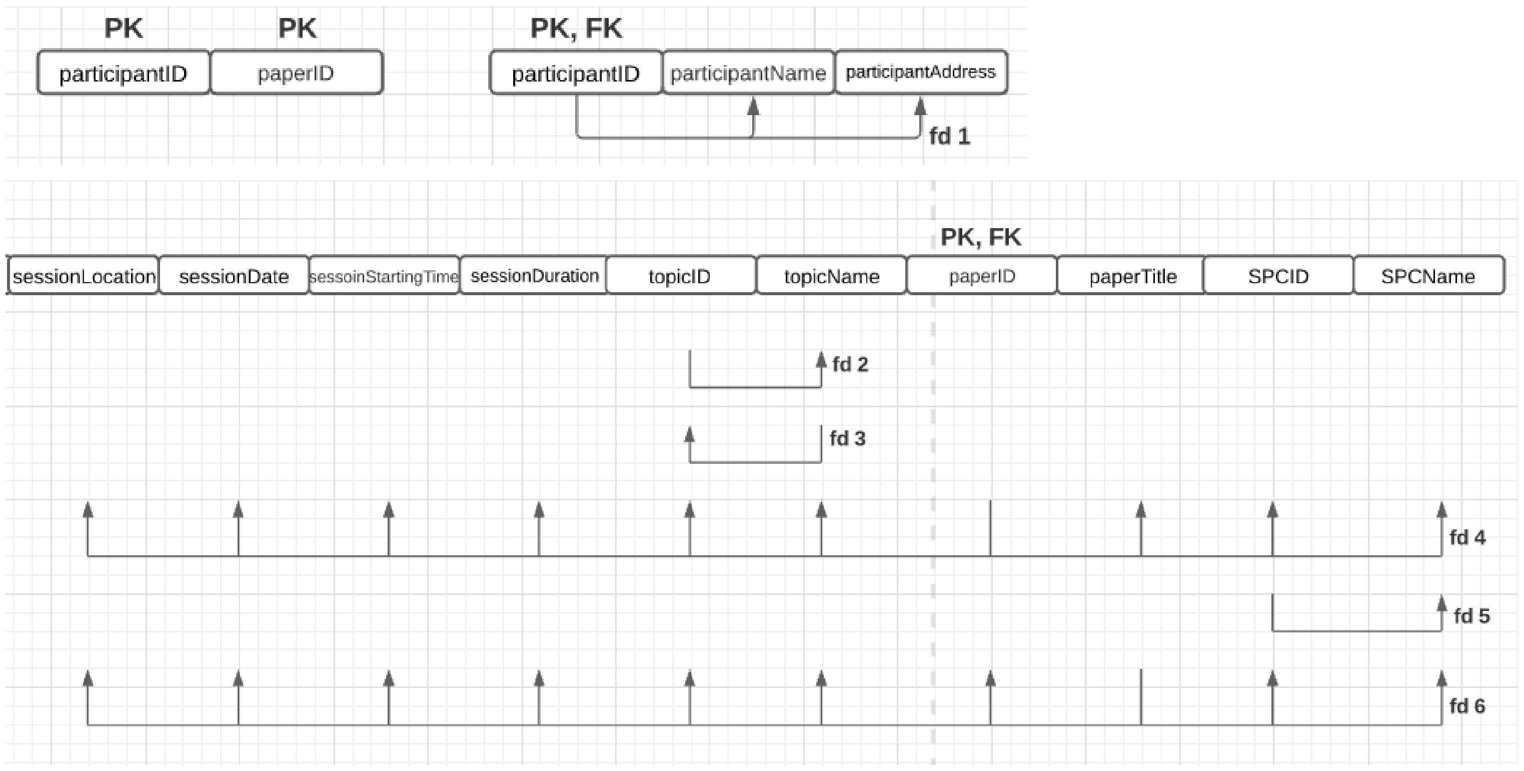
table1 (participantID, paperID)

table2 (participantID, participantName, participantAddress)

table 3 (paperID, paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName)

Then we define the name of each table as follow.
2NF:
ParticipantPaper (participantID, paperID)
ParticipantDetails (participantID, participantName, participantAddress)
PaperSessionTopicSPC (paperID, paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName)

2NF



Note that:

- Foreign Key participantID in ParticipantDetails references ParticipantPaper(participantID)
- Foreign Key paperID in PaperSessionTopicSPC references ParticipantPaper(paperID)

4. 2nf to 3nf

remove transitive dependencies.

ParticipantPaper (participantID, paperID)
no transitive dependency.

ParticipantDetails (participantID, participantName, participantAddress)
no transitive dependency.

PaperSessionTopicSPC (paperID, paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName)

transitive dependencies already derived in the UNF to 1NF step:

SPCID -> SPCName

topicID -> topicName

topicName -> topicID

paperTitle -> sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID, SPCName

transitive dependencies that haven't been dereved yet:

sessionLocation, sessionDate, sessoinStartingTime -> sessionDuration, topicID, topicName
(when we know sessionLocatoin, sessionDate, and sessionStartingTime, we can identify sessionDuration, topicID, and topicName)

sessionDate, sessoinStartingTime, topicID -> sessionDuration, sessionLocation, topicName
(when we know sessionDate, sessionStartingTime, and topicID, we can identify sessionDuration, sessionLocation, and topicName)

topicID, sessionDate -> sessionLocation, sessionStartingTime, sessoinDuration, topicName
(when we know topicID and sessionDate, we can identify sessionLocation, sessionStratingTime, sessionDuration, and topicName)

So, we can decompose from 2NF into 3NF as follow.

1. Table A (SPCID, SPCName)
   PaperSessionTopicSPC (paperID, paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, topicName, SPCID)
   : Foreign Key SPCID in table A references PaperSessionTopicSPC(SPCID)

2. Table B (topicID, topicName)
PaperSessionTopicSPC (paperID, paperTitle, sessionLocation, sessionDate, sessoinStartingTime, sessionDuration, topicID, SPCID)
: Foreign Key topicID in table B references PaperSessionTopicSPC(topicID)

3. Table C (topicID, sessionDate, sessionLocation, sessionStartingTime, sessoinDuration)
PaperSessionTopicSPC (paperID, paperTitle, sessionDate, topicID, SPCID)
: Foreign Key topicID and sessionDate in table C references PaperSessionTopicSPC(topicID, sessionDate)

Overall, we will get the following tables from decomposing PaperSessionTopicSPC.

PaperSessionTopicSPC (paperID, paperTitle, sessionDate, topicID, SPCID)
Table A (SPCID, SPCName)
Table B (topicID, topicName)
Table C (topicID, sessionDate, sessionLocation, sessionStartingTime, sessoinDuration)

Then, we can rename table A, B and C as follow.

PaperSessionTopicSPC (paperID, paperTitle, sessionDate, topicID, SPCID)
SPCDetails (SPCID, SPCName)
TopicDetails (topicID, topicName)
TopicSession (topicID, sessionDate, sessionLocation, sessionStartingTime, sessoinDuration)

overall, the final outcomes are shown below.

3NF:

ParticipantPaper (participantID, paperID)

ParticipantDetails (participantID, participationName, participationAddress)

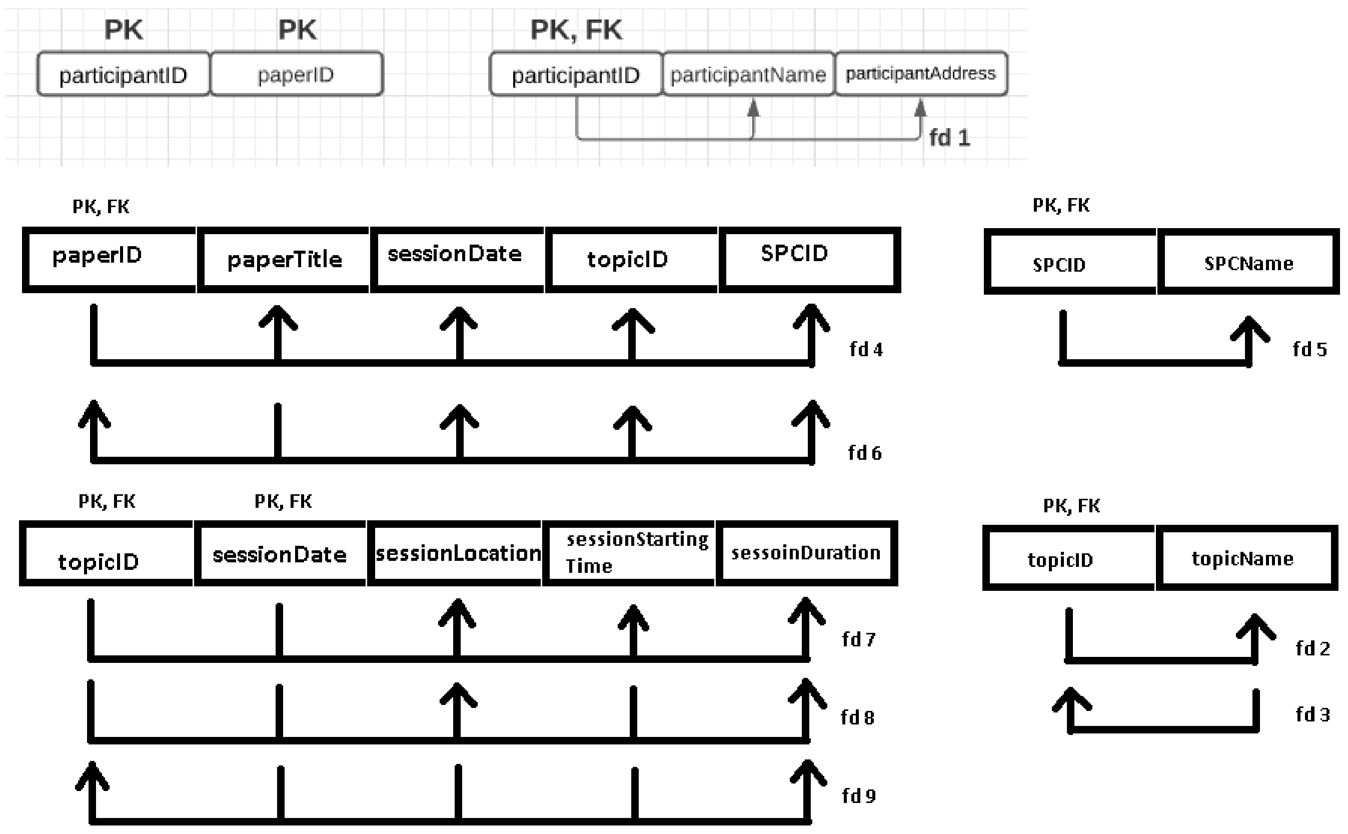
PaperSessionTopicSPC (paperID, paperTitle, sessionDate, topicID, SPCID)

SPCDetails (SPCID, SPCName)

TopicDetails (topicID, topicName)

TopicSession (topicID, sessionDate, sessionLocation, sessionStartingTime, sessoinDuration)

3NF



Note that:

- Foreign Key participantID in ParticipantDetails references ParticipantPaper(participantID)
- Foreign Key paperID in PaperSessionTopicSPC references ParticipantPaper(paperID)
- Foreign Key SPCID in SPCDetails references PaperSessionTopicSPC(SPCID)
- Foreign Key topicID in TopicDetails references PaperSessionTopicSPC(topicID)
- Foreign Key topicID and sessionDate in TopicSession references PaperSessionTopicSPC(topicID, sessionDate)