75/100

COMP518 Assignment 2: Logical Models & Normalization

Peter Prescott (201442927)

December 9, 2020

I. Logical Data Model

46/60

"Create the appropriate *relations* to represent the *entities* and *relationships* of the E-R diagram."

i. Strong Entities

- **Account** (**ssn**, gender, phoneNumber, email, dateOfBirth)
- **Product** (**id**, time)
- **Organization** (**name**, address)
- **VirtualMachine** (**id**, capability)

ii. Weak Entities

- **DeveloperAccount** (officeAddress)
- **CustomerAccount** (creditCard, homeAddress)
- **Music** (type, quality)
- **Book** (author, title)
- **Video** (title, rank)
- **Image** (size, category)

iii. Relationships

- **DeveloperAccount** (0..*) *Affiliated_to* (1..1) **Organization** (Optional Many-to-One)
- **CustomerAccount** (0..1) *Owns* (1..*) **VirtualMachine** (Optional One-to-Many)
- **Organization** (1..1) *Maintains* (0..*) **VirtualMachine** (One-to-Many)
- **Product** (0..*) *Runs_on* (0..*) **VirtualMachine** (Many-to-Many)
- **DeveloperAccount** (1..*) *Develops* (0..*) **Product** (Many-to-Many)
- **CustomerAccount** (0..*) *Buys* (0..*) **Product** (Many-to-Many)

iii.1 One-to-Many (1:*) binary relationship types

"we post a copy of the primary key of the parent entity into the relation representing the child entity, to act as a foreign key" (Connolly & Begg, 2015, p. 531).

Organization (parent) -> Developer (child)

- **Developer** (officeAddress, *dev_org*) : Foreign Key `dev_org` references Organization(**name**)

CustomerAccount (parent) -> VirtualMachine (child)

- **VirtualMachine** (**id**, capability, *owner*) : Foreign Key `owner` references Account(**id**)

Organization (parent) -> VirtualMachine(child)

- **VirtualMachine** (**id**, capability, *owner*, *mac_org*) : Foreign Key `mac_org` references Organization(**name**)

"in the case where a 1:* relationship has one or more attributes, these attributes should follow the posting of the primary key to the child relation" (Connolly & Begg, 2015, p. 531)

CustomerAccount (parent) -> VirtualMachine (child)

- **VirtualMachine** (**id**, capability, *owner*, date) ,mac_org

iii.2 Many-to-Many (*:*) Relationships

"For each *:* binary relationship, create a relation to represent the relationship and include any attributes that are part of the relationship. We post a copy of the primary key attribute(s) of the entities that participate in the relationship into the new relation, to act as foreign keys. One or both of these foreign keys will also form the primary key of the new relation..." (Connolly & Begg, 2015, p. 535)

- Product *Runs_on* VirtualMachine —> create **MachineUsage** (**product_id**, **mac_id**, product_type)
- DeveloperAccount *Develops* Product —> create **ProductDevelopment** (**dev_ssn**, **product_id**, product_type)
  - Since these two reference `product_id`, ie. the *foreign key* referencing Product(id), but as we will combine the `Product` entity with its disjoint subtypes following Connolly & Begg's instruction referenced below, I have also included `product_type` so that it is not necessary to search through all the product subtype tables to find the required `product_id`.
  - `mac_id` references VirtualMachine(id)
  - `dev_ssn` references Account(id)
- CustomerAccount *Buys* Product —> create **Purchase** (**customer_ssn**, **mac_id**, price, discount)
  - `customer_ssn` references Account(id)
  - `mac_id` references VirtualMachine(id)

iv. Combine Superclass/Subclass Relationships

- DeveloperAccount, CustomerAccount *{mandatory, and}* Account
- Music, Book, Video, Image {mandatory, or} Product

For *{mandatory, and}*: "Single relation (with one or more discriminators to distinguish the type of each tuple" (Connolly & Begg, 2015, p. 534)

- **Account** (**ssn**, gender, phoneNumber, email, dateOfBirth, officeAddress, *dev_org*, creditCard, homeAddress, devFlag, customerFlag)

It is safe to assume that the overlapping on the Accounts is not significant and additionally customer and developer entities participate in completely different relationships. We decide to treat the situation as Mandatory,Or and create two relations, namely DeveloperAccount and CustomerAccount copying all the attributes from the superclass. Therefore we don't need the Account relation.

For *{mandatory, or}*: "Many relations; one relation for each combined superclass/subclass" (Connolly & Begg, 2015, p. 534)

- **Music** (**id**, time, type, quality)
- **Book** (**id**, time, author, title)
- **Video** (**id**, time, title, rank)
- **Image** (**id**, time, size, category)

v. Final Relations

"Designate the *primary key*, and any *alternate* or *foreign* keys of each relation together with their references."

Attributes that form part of the primary key is **bold** – if two attributes are bold they are both necessary to make up the PK.
Foreign keys are shown in *italic* with reference explained underneath.

- **Account** (**ssn**, gender, phoneNumber, email, dateOfBirth, officeAddress, *dev_org*, creditCard, homeAddress, devFlag, customerFlag) Alternate key: email
  - Foreign Key `dev_org` references Organization(name)
- **Organization** (**name**, address)
- **VirtualMachine** (**mac_id**, capability, *owner*, date) ,mac_org
  - Foreign Key `owner` references Account(ssn) FK: mc_org ...
  - Primary Key `mac_id` renamed to distinguish from `product_id`
- **Music** (**product_id**, time, type, quality)
  - Primary Key `product_id` renamed to distinguish from `mac_id` (and below)
- **Book** (**product_id**, time, author, title)
- **Video** (**product_id**, time, title, rank)

- **Image** (**product_id**, time, size, category)
- **MachineUsage** (***product_id***, ***mac_id***, product_type)
  - Foreign Key `product_id` references Product(product_id) found in table given by `product_type`
  - Foreign Key `mac_id` references VirtualMachine(mac_id)
  - Primary key (product_id, mac_id)
- **ProductDevelopment**(***dev_ssn***, ***product_id***, product_type)
  - Foreign Key `product_id` references Product(product_id) found in table given by `product_type`
  - Foreign Key `dev_ssn` references Account(ssn)
  - Primary key(dev_ssn, product_id)
- **Purchase**(***customer_ssn***, ***mac_id***, price, discount)
  - Foreign Key `customer_ssn` references Account(ssn)
  - Foreign Key `mac_id` references VirtualMachine(mac_id)

We decide to also keep the Product relation, even if it is Mandatory,Or, because it participates in many relationships and more importantly, omitting it here causes problems to referential integrity. Then the attribute time would be part of the Product relation only.

29/40

II. Normalization

Decompose the `ConferenceData` table into relations which are in 3rd normal form.

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

i. Identifying Primary Key Using Functional Dependencies

participantID, sessionDate, sessionStartingTime -> sessionLocation, sessionDuration, topicID, topicName, SPCID, SPCName

1. participantID *functionally determines* participantName, participantAddress
2. paperID *functionally determines* paperTitle, SPCID, SPCName, topicID, topicName, sessionLocation, sessionDate, sessionStartTime, sessionDuration
3. topicID *functionally determines* topicName , SPCID, SPCName
4. topicName *functionally determines* topicID
5. SPCID *functionally determines* SPCName
6. sessionDate, topicID *functionally determines* sessionLocation, sessionStartingTime , sessionDuration
7. sessionLocation, sessionDate, sessionStartingTime *functionally determines* topicID, topicName , sessionDuration, SPCID, SPCName

Primary Key: participantID, paperID, sessionDate, topicID

paperID -> topicID, seesionDate (2nd dependency), so the PK should have been paperID, participantID

ii. First Normal Form

"A relation in which the intersection of each row and column contains one and only one value" (Connolly & Begg, 2015, p. 466)

Participants can attend multiple sessions (provided no clash) so attributes related to session (`sessionLocation`, `sessionDate`, `sessionStartTime`, `sessionDuration`, `topicID`, `topicName`) may have multiple values for given participant.

Sessions last up to 2 hours; papers last 20 min, plus 5 min for questions. So up to four papers per session.

Assignment says to assume table is flattened, so multiple values recorded through repeating data on multiple rows. Therefore table is in First Normal Form.

iii. Second Normal Form

"A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key" (Connolly & Begg, 2015, p. 470)

1. Participant(**participantID**, participantName, participantAddress)
2. Paper(**paperID**, paperTitle, SPCID, SPCName, topicID, topicName, sessionLocation, sessionDate, sessionStartTime, sessionDuration)
3. Topic(**topicID**, topicName)
4. SPC(**SPCID**, SPCName) SPCID->SPCName was not a partial dependency. No need to have this table for 2NF.
5. Session(**sessionDate**, **topicID**, sessionLocaton, sessionStartingTime)

You should keep the ConferenceData table with the PK you found in 1NF.

If an attribute is fully dependent on many subset of the PK (e.g. regarding the PK you identified, paperID -> topicName and topicID -> topicName) then you remove it only to one table, otherwise you end up with data redundancy.

iv. Third Normal Form

"A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on the primary key." (Connolly & Begg, 2015, p. 472)

Transitive Dependencies:

- (2) + (3) : paperID –> topicID –> topicName
- (2) + (4) : paperID –> SPCID –> SPCName
- (2) + (5) : paperID –> sessionDate, topicID –> sessionLocation, sessionDuration, sessionStartTime

Finalized Entities in 3rd Normal Form:

1. Participant(**participantID**, participantName, participantAddress)
2. Paper(**paperID**, paperTitle, topicID, sessionDate, SPCID)
3. Session(**sessionDate**, **topicID**, sessionStartingTime, sessionLocation, sessionDuration)
4. Topic(**topicID**, topicName)
5. SPC(**SPCID**, SPCName)

ConferenceData(...

References

Connolly, T. M., & Begg, C. E. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (Sixth Edition). Pearson Education.

Your final tables are close to the correct ones (apart from ConferenceData table, which is crucial in order to connect the rest of the tables, and SPCID which should have been in the Topic table). The main problem with your solution is that even if you identified most of the dependencies correctly, you didn't derived the correct PK based on them (this would be a problem because the PK should appear in the ConferenceData table) and the steps of normalization weren't followed exactly, causing the lack of ConferenceData table and extra tables in 2NF that were only supposed to appear in 3NF.