Coursework 2: Logical Models and Normalisation

201374125 *University of Liverpool*

Part A: Logical Database Modelling

Strong Entities

Account (username, phoneNo, email, dateOfBirth)

Device (uuid, macAddress, serialNo, opSystem, opVersion)

Product (uuid, displayName)

Weak Entities

DevAccount (bankNo, sortCode, officeAddress, website)

CustAccount (cardNo, cardExpiry, homeAddress)

App (name, size)

Music (genre, artist)

Book (title, author, publisher)

Movie (title, genre, rating)

Downloads (price, purchaseDate)

Weak entities rely on strong entities, because of this they have no primary key which they take from the strong entity they are associated with.

Combine Superclass/subclass Relationships

DevAccount, CustAccount *{mandatory, and}* Account

Account (username, phoneNo, email, dateOfBirth, bankNo, sortCode, officeAddress, website, cardNo, cardExpiry, homeAddress, dAccountFlag, cAccountFlag)

Primary Key username

As the participation constraint is mandatory and nondisjoint, all attributes from the parent entity, Account, and the two child entities, DevAccount, and CustAccount have been combined. The two attributes dAccountFlag and cAccountFlag have been added as discriminators, to determine whether an account is one or both types. This entity takes the primary key from the original strong entity.

App, Music, Book, Movie {mandatory, or} Product

App (uuid, displayName, name, size)

Primary Key uuid

Music (uuid, genre, displayName, artist)

Primary Key uuid

Book (uuid, displayName, author, publisher)

Primary Key uuid

Movie (uuid, displayName, title, genre, rating)

Primary Key uuid

As the participation constraint is mandatory and disjoint, each child entity forms its own entity, taking the Primary key from the parent entity.

I have chosen to start by combining the superclass and subclass relationships as when deriving the one to many relationships of the DevAccount and CustAccount, a primary key was requried, this is provided by the Account superclass so must be combined first.

One to Many Relationships

DevAccount *develops* Product (1:*)

App (uuid, displayName, name, size, username)

Primary Key uuid

Foreign Key username **references** Account(username)

Music (uuid, displayName, genre, artist, username)

Primary Key uuid

Foreign Key username **references** Account(username)

Book (uuid, displayName, title, author, publisher, username)

Primary Key uuid

Foreign Key username **references** Account(username)

Movie (uuid, displayName, title, genre, rating, username)

Primary Key uuid

Foreign Key username **references** Account(username)

The relationship between DevAccount and Product is one to many. As DevAccount takes the primary key from Account, username, this is used as the Foreign key in Product, as Product is the child entity of DevAccount.

As shown above, the product entity has been replaced by entities that represent the product type, App, Music, Book, or Movie

CustAccount *owns* Device (1:*)

Device (uuid, macAddress, serialNo, opSystem, opVersion, username)

Primary Key uuid

Foreign Key username **references** Account(username)

The relationship between CustAccount and Device is one to many. As CustAccount takes the primary key from Account, username, this is used as the Foreign key in Device, as Device is the child entity of CustAccount.

Many to Many Relationships

CustAccount *downloads* Product (*:*)

Downloads (username, uuid, price, purchaseDate)

Primary Key username, uuid

Foreign Key username **references** Account(username)

Foreign Key uuid **references** Product(uuid)

A relation is created to represent the downloads relationship, with the attributes from downloads included. Both primary keys from the entities participating in this relation, CustAccount, and Product are copied into the new relation, and act as foreign keys. The foreign keys together also act as the primary key.

Procut *runsOn* Device (*:*)

runsOn (uuidp, uuidd)

Primary Key uuidp, uuidd

Foreign Key uuidp **references** Product(uuid)

Foreign Key uuidd **references** Device(uuid)

A relation is created to represent the runsOn relationship. Both primary keys from the entities participating in this relation, Product, and Device are copied into the new relation, and act as foreign keys. The foreign keys together also act as the primary key. They have been renamed as they originally shared the same name.

Final Relations

Account (username, phoneNo, email, dateOfBirth, bankNo, sortCode, officeAddress, website, cardNo, cardExpiry, homeAddress, dAccountFlag, cAccountFlag)

Primary Key username

Device (uuid, macAddress, serialNo, opSystem, opVersion, username)

Primary Key uuid

Foreign Key username **references** Account(username)

Downloads (username, uuid, price, purchaseDate)

Primary Key username, uuid

Foreign Key username **references** Account(username)

Foreign Key uuid **references** Product(uuid)

runsOn (uuidp, uuidd)
Primary Key uuidp, uuidd
Foreign Key uuidp **references** Product(uuid)
Foreign Key uuidd **references** Device(uuid)

App (uuid, displayName, name, size, username)
Primary Key uuid
Foreign Key username **references** Account(username)

Music (uuid, displayName, genre, artist, username)
Primary Key uuid
Foreign Key username **references** Account(username)

Book (uuid, displayName, title, author, publisher, username)
Primary Key uuid
Foreign Key username **references** Account(username)

Movie (uuid, displayName, title, genre, rating, username)
Primary Key uuid
Foreign Key username **references** Account(username)

Part B: Normalisation

ConferenceData (delegateID, delegateName, delegateAddress, sessionLocation, sessionDate, sessionStartingTime, sessionDuration, topicID, topicName, paperID, paperTitle, refereeID, refereeName)

Identifying Functional Dependencies

fd1 delegateID → delegateName, delegateAddress

fd2 paperID → topicID, topicName, paperTitle, delegateID, refereeID, delegateName, refereeName

fd3 topicID → topicName

fd4 topicName → topicID

fd5 refereeID → refereeName, paperID, paperTitle

fd6 topicID, sessionDate → topicName, sessionStartingTime, sessionLocation, sessionDuration

fd7 sessionLocation, sessionDate, sessionStartingTime → topicID, topicName

Primary Key:

delegateID, paperID, topicID, sessionDate (fd1, fd2, fd5)

First Normal Form

Delegates can choose more than one session so;

sessionLocation, sessionDate, sessionStartingTime, topicID, topicName

may have multiple values in one attribute.

Up to 16 papers per session means *paperID, paperTitle* may also have multiple values.

It is assumed that as this table is already in use, the data is repeated through ‘flattening’. Therefore this table is currently in First Normal Form.

Second Normal Form

fd1, fd2, fd3, fd6, fd7 Partial dependencies on Primary Key.

ConferenceData (delegateID, ~~delegateName, delegateAddress~~, sessionLocation, sessionDate, sessionStartingTime, sessionDuration, topicID, ~~topicName~~, paperID, ~~paperTitle,~~ ~~refereeID, refereeName~~)

fd1 **Delegate** (delegateID, delegateName, delegateAddress)

fd2 **Paper** (paperID, paperTitle, topicID, topicName, refereeID, refereeName, delegateID)

fd3 No longer a partial dependency

fd6 **Session** (sessionDate, topicID, delegateID, paperID, sessionStartingTime, sessionLocation, sessionDuration)

fd7 No longer a partial dependency

Third Normal Form

Transitive Dependencies:

fd2 + fd3:

paperID → topicID → topicName

fd2 + fd5:

paperID → refereeID → refereeName

Delegate (delegateID, delegateName, delegateAddress)

Paper (paperID, paperTitle, topicID, refereeID, delegateID)

Session (sessionDate, topicID, delegateID, paperID, sessionStartingTime, sessionLocation, sessionDuration)

Topic (topicID, topicName)

Referee (refereeID, refereeName)

Boyce-Codd Normal Form

Session (sessionDate, topicID, delegateID, paperID, sessionStartingTime, sessionLocation, sessionDuration)

fd1 sessionDate, topicID → delegateID, paperID, sessionStartingtime, sessionLocation, sessionDuration

fd2 sessionStartingTime, sessionLocation, sessionDate → topicID, delegateID, paperID

fd3 paperID → topicID

Since paperID can determine the partial part of the primary key (topicID) they both need to be split off into a seperate table,with paperID as the primary key, and topicID acting as a foreign key.

PaperTopic (paperID, topicID)

Session (sessionDate, topicID, delegateID, sessionStartingTime, sessionLocation, sessionDuration)

Finished Normalisation

Delegate (delegateID, delegateName, delegateAddress)

Paper (paperID, paperTitle, topicID, refereeID, delegateID)

Session (sessionDate, topicID, delegateID, sessionStartingTime, sessionLocation, sessionDuration)

PaperTopic (paperID, topicID)

Topic (topicID, topicName)

Referee (refereeID, refereeName)