A5: Relational Schema, validation and schema refinement

GROUP1714, 18/03/2018

- Daniel Ribeiro de Pinho - up201505302@fe.up.pt
- Francisco José Sousa Silva - up201502860@fe.up.pt
- Rui André Rebolo Fernandes Leixo - up201504818@fe.up.pt
- Vitor Emanuel Fernandes Magalhães - up201503447@fe.up.pt

1. Relational Schema

> The Relational Schema includes the relation schemas, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK.
> Relation schemas are specified in the compact notation:

Relation reference	Relation Compact Notation
R01	Event_EventAdmin(**idEvent->Event__, __idEventAdmin->EventAdmin**)
R02	Event_Member(**idEvent->Event__, __idMember->Member**)
R03	Comment(**idComment**, text **NN**, timestamp **NN**, idEvent->Event, idMember->Member)
R04	Ticket(**idTicket**, idTicketType->TicketType, idMember->Member)
R05	TicketType(**idTicketType**, type **NN**, price **CK price > 0 NN**, initialQuantity **CK initialQuantity > 0 NN**, availableQuantity **CK availableQuantity > 0 NN**, description)
R06	Invoice(**idInvoice**, name, taxPayerNumber, adress, quantity **NN**, amount **NN**, date **NN**)
R07	Community(**idCommunity**, name **NN**, description **NN**, creationDate **NN**, imagePath, publicLink **UK**, isPublic **NN**)
R08	Community_Member(**idCommunity->Community__, __idMember->Member**)
R09	Community_CommunityCategory(**idCommunity->Community__, __idCommunityCategory->CommunityCategory**)
R10	Community_CommunityAdmin(**idCommunity->Community__, __idCommunityAdmin->CommunityAdmin**)
R11	Member(**idMember**, name **NN**, username **UK NN**, password **NN**, birthdate **NN**, email **UK NN**, country, city, address, taxPayerNumber, about, profilePicture, registrationDate **NN**, sentEmailVerification **NN**, verifiedEmail **NN**, isWebSiteAdmin **NN**)
R12	Friend(**idFriend1->Member__, __idFriend2->Member**, accepted **NN**)
R13	CommunityAdmin(**idCommunityAdmin->Member**)
R14	CommunityCategory(**idCommunityCategory**, name **NN**)

R15	EventAdmin(**idEventAdmin->Member**)
R16	Report(**idReport**, timestamp, context **NN**, idCommunity->Community, idMember->Member, idEvent-Event, idComment->Comment)
R17	Notification(**idNotification**, idCommunity->Community, idMember->Member, idEvent-Event, idComment->Comment)
R18	Event(**idEvent**, name **NN**, description **NN**, imagePath, date **NN**, country **NN**, city **NN**, address **NN**, publicLink **UK**, isPublic **NN**, idCommunity->Community)
R19	EventCategory(**idEventCategory**, name **NN**)
R20	Event_EventCategory(**idEvent->Event__, __idEventCategory->EventCategory**)

2. Domains

The specification of additional domains can also be made in a compact form, using the notation:

Domain Name	**Domain Specification**
Today	DATE DEFAULT CURRENT_DATE
Priority	ENUM ('High', 'Medium', 'Low')

3. Functional Dependencies and schema validation

Table R01 (Event_EventAdmin)	
Keys: { idEvent, idEventAdmin }	
Functional Dependencies	
FD0101	{idEvent} → {idEvent, idEventAdmin}
FD0102	{idEventAdmin} → {idEventAdmin, idEvent}
Normal Form	BCNF

Table R02 (Event_Member)	
Keys: { idEvent, idMember }	
Functional Dependencies	
FD0201	{idEvent} → {idEvent, idMember}
FD0202	{idMember} → {idEvent, idMember}
Normal Form	BCNF

Table R03 (Comment)	
Keys: { idComment }	
Functional Dependencies	

FD0301	{idComment} → {idComment, text, timestamp, idEvent, idMember}
Normal Form	BCNF

Table R04 (Ticket)	
Keys: { idTicket }	
Functional Dependencies	
FD0401	{idTicket} → {idTicket, idTicketType, idMember}
Normal Form	BCNF

Table R05 (TicketType)	
Keys: { idTicketType }	
Functional Dependencies	
FD0501	{idTicketType} → {idTicketType, type, price, initialQuantity, availableQuantity, description}
Normal Form	BCNF

Table R06 (Invoice)	
Keys: { idInvoice }	
Functional Dependencies	
FD0601	{idInvoice} → {idInvoice, name, taxPayerNumber, adress, quantity, amount, date}
Normal Form	BCNF

Table R07 (Community)	
Keys: { idCommunity }	
Functional Dependencies	
FD0701	{idCommunity} → {idCommunity, name, description, creationDate, imagePath, publicLink, isPublic}
Normal Form	BCNF

Table R08 (Community_Member)	
Keys: { idCommunity, idMember }	
Functional Dependencies	
FD0801	{idCommunity} → {idCommunity, idMember}
FD0802	{idMember} → {idCommunity, idMember}
Normal Form	BCNF

Table R09 (Community_CommunityCategory)

Keys: { idCommunity, idCommunityCategory }	
Functional Dependencies	
FD0901	{idCommunity} → {idCommunity, idCommunityCategory}
FD0902	{idCommunityCategory} → {idCommunity, idCommunityCategory}
Normal Form	BCNF

Table R10 (Community_CommunityAdmin)	
Keys: { idCommunity, idCommunityAdmin }	
Functional Dependencies	
FD1001	{idCommunity} → {idCommunity, idCommunityAdmin}
FD1002	{idCommunityAdmin} → {idCommunity, idCommunityAdmin}
Normal Form	BCNF

Table R11 (Member)	
Keys: { idMember }	
Functional Dependencies	
FD1101	{idMember} → {idMember, name, username, password, birthdate, email, country, city, address, taxPayerNumber, about, profilePicture, registrationDate, sentEmailVerification, verifiedEmail, isWebSiteAdmin}
Normal Form	BCNF

Table R12 (Friend)	
Keys: { idFriend1, idFriend2 }	
Functional Dependencies	
FD1201	{idFriend1} → {idFriend1, idFriend2}
FD1202	{idFriend2} → {idFriend1, idFriend2}
FD1203	{idFriend1, idFriend2} → {idFriend1, idFriend2, accepted}
Normal Form	BCNF

Table R13 (CommunityAdmin)	
Keys: { idCommunityAdmin }	
Functional Dependencies	
FD1301	{idCommunityAdmin} → {idCommunityAdmin}
Normal Form	BCNF

Table R14 (CommunityAdmin)	
Keys: { idCommunityCategory }	
Functional Dependencies	
FD1401	{idCommunityCategory} → {idCommunityCategory, name}
Normal Form	BCNF

Table R15 (CommunityAdmin)	
Keys: { idEventAdmin }	
Functional Dependencies	
FD1501	{idEventAdmin} → {idEventAdmin}
Normal Form	BCNF

Table R16 (Report)	
Keys: { idReport }	
Functional Dependencies	
FD1601	{idReport} → {idReport, timestamp, context, idCommunity, idMember, idEvent, idComment}
Normal Form	BCNF

Table R17 (Notification)	
Keys: { idNotification }	
Functional Dependencies	
FD1701	{idNotification} → {idNotification, idCommunity, idMember, idEvent, idComment}
Normal Form	BCNF

Table R18 (Event)	
Keys: { idEvent }	
Functional Dependencies	
FD1801	{idEvent} → {idEvent, name, description, imagePath, date, country, city, address, publicLink, isPublic, idCommunity}
Normal Form	BCNF

Table R19 (EventCategory)	
Keys: { idEventCategory }	
Functional Dependencies	

FD1901	{idEventCategory} → {idEventCategory, name}
Normal Form	BCNF

Table R20 (Event_EventCategory)	
Keys: { idEvent, idEventCategory }	
Functional Dependencies	
FD2001	{idEvent} → {idEvent, idEventCategory}
FD2002	{idEventCategory} → {idEvent, idEventCategory}
Normal Form	BCNF

> If necessary, description of the changes necessary to convert the schema to BCNF.
> Justification of the BCNF.

The Boyce-Codd Normal Form (BCNF) is used here since it is a slightly stronger database normalization method when compared to the 3NF, better ensuring that there are no data anomalies when making changes to the database.

A relational schema is said to be in the BCNF when, for each of its functional dependencies (X → Y), one of the following happens:

- X → Y is a trivial functional dependency (i.e. Y is contained within X)
- X is a superkey for the schema

4. SQL Code

Link to raw in repository

```/* ---------------------------------------------------- // Generated by Enterprise Architect Version 13.5 // Created On : 18-Mar-2018 01:33:48 // DBMS : PostgreSQL // ---------------------------------------------------- */

/* Drop Tables */

DROP TABLE IF EXISTS Comment CASCADE;

DROP TABLE IF EXISTS Community CASCADE;

DROP TABLE IF EXISTS CommunityAdmin CASCADE;

DROP TABLE IF EXISTS CommunityCategory CASCADE;

DROP TABLE IF EXISTS Event CASCADE;

DROP TABLE IF EXISTS EventAdmin CASCADE;

DROP TABLE IF EXISTS EventCategory CASCADE;

DROP TABLE IF EXISTS Friend CASCADE;

DROP TABLE IF EXISTS Invoice CASCADE;

DROP TABLE IF EXISTS Member CASCADE;
```

```
DROP TABLE IF EXISTS Notification CASCADE;

DROP TABLE IF EXISTS Report CASCADE;

DROP TABLE IF EXISTS Ticket CASCADE;

DROP TABLE IF EXISTS TicketType CASCADE;

/* Create Tables */

CREATE TABLE Comment(
idComment integer NOT NULL,
text text NOT NULL,
timestamp timestamp w ithout time zone NOT NULL,
idEvent integer,
idMember integer,
);

CREATE TABLE Community(
idCommunity integer NOT NULL,
name varchar(64) NOT NULL,
description varchar(256) NOT NULL,
creationDate date NOT NULL,
imagePath path NULL,
publicLink path NULL,
isPublic boolean NOT NULL
);

CREATE TABLE CommunityAdmin(
idCommunityAdmin integer NOT NULL
);

CREATE TABLE CommunityCategory(
idCommunityCategory integer NOT NULL,
name varchar(50) NULL
);

CREATE TABLE Event(
idEvent integer NOT NULL,
name varchar(64) NOT NULL,
description varchar(516) NOT NULL,
imagePath path NULL,
date date NOT NULL,
country varchar(50) NOT NULL,
city varchar(50) NOT NULL,
address varchar(100) NOT NULL,
publicLink path NULL,
isPublic boolean NOT NULL
);

CREATE TABLE EventAdmin(
idEventAdmin integer NOT NULL
);

CREATE TABLE EventCategory(
idEventCategory integer NOT NULL,
name varchar(50) NOT NULL
```

```
);

CREATE TABLE Invoice(
idInvoice integer NOT NULL,
taxPayerNumber int NULL,
name varchar(50) NULL,
address text NULL,
quantity int NOT NULL CHECK (quantity>0),
amount int NOT NULL CHECK (amount>0),
date date NOT NULL
);

CREATE TABLE Member(
idMember integer NOT NULL,
username varchar(16) NOT NULL UNIQUE,
passw ord text NOT NULL,
name varchar(50) NOT NULL,
birthdate date NOT NULL,
email varchar(50) NOT NULL UNIQUE,
country varchar(50) NULL,
city varchar(50) NULL,
address text NULL,
taxPayerNumber varchar(20) NULL UNIQUE,
about varchar(256) NULL,
profilePicture path NULL,
registrationDate date NOT NULL,
sentEmailVerification boolean NOT NULL,
verifiedEmail boolean NOT NULL,
isWebsiteAdmin boolean NOT NULL
);

CREATE TABLE Notification(
idNotification integer NOT NULL,
acceptedInvitation boolean NULL
);

CREATE TABLE Report(
idReport integer NOT NULL,
timestamp timestamp w ithout time zone NOT NULL,
context text NOT NULL
);

CREATE TABLE Ticket(
idTicket integer NOT NULL,
idTicketType integer NULL,
idInvoice integer NOT NULL,
idMember integer NOT NULL
);

CREATE TABLE TicketType(
idTicketType integer NOT NULL,
price double precision NOT NULL CHECK (price>0),
initialQuantity integer NOT NULL CHECK (initialQuantity>0),
"#availableQuantity" integer NULL,
descripiton text NULL,
);
```

```
/* Tables to model Many to Many relations */

/------------Community_Member------------/
CREATE TABLE Community_Member(
idCommunity integer NOT NULL,
idMember integer NOT NULL
);

ALTER TABLE Community_Member ADD CONSTRAINT PK_Community_Member
PRIMARY KEY (idCommunity, idMember);

ALTER TABLE Community_Member ADD CONSTRAINT FK_Community_Member_Community
FOREIGN KEY (idCommunity) REFERENCES Community(idCommunity);

ALTER TABLE Community_Member ADD CONSTRAINT FK_Community_Member_Member
FOREIGN KEY (idMember) REFERENCES Member(idMember);

/------------Community_CommunityCategory------------/
CREATE TABLE Community_CommunityCategory(
idCommunity integer NOT NULL,
idCommunityCategory integer NOT NULL
);

ALTER TABLE Community_CommunityCategory ADD CONSTRAINT PK_Community_CommunityCategory
PRIMARY KEY (idCommunity, idCommunityCategory);

ALTER TABLE Community_CommunityCategory ADD CONSTRAINT FK_Community_CommunityCategory_Community
FOREIGN KEY (idCommunity) REFERENCES Community(idCommunity);

ALTER TABLE Community_CommunityCategory ADD CONSTRAINT FK_Community_CommunityCategory_CommunityCategory
FOREIGN KEY (idCommunityCategory) REFERENCES CommunityCategory(idCommunityCategory);

/------------Community_CommunityAdmin------------/
CREATE TABLE Community_CommunityAdmin(
idCommunity integer NOT NULL,
idCommunityAdmin integer NOT NULL
);

ALTER TABLE Community_CommunityAdmin ADD CONSTRAINT PK_Community_CommunityAdmin
PRIMARY KEY (idCommunity, idCommunityAdmin);

ALTER TABLE Community_CommunityAdmin ADD CONSTRAINT FK_Community_CommunityAdmin_Community
FOREIGN KEY (idCommunity) REFERENCES Community(idCommunity);

ALTER TABLE Community_CommunityAdmin ADD CONSTRAINT FK_Community_CommunityAdmin_CommunityAdmin
FOREIGN KEY (idCommunityAdmin) REFERENCES CommunityAdmin(idCommunityAdmin);

/------------Event_EventAdmin------------/
CREATE TABLE Event_EventAdmin (
idEvent integer NOT NULL,
idEventAdmin integer NOT NULL
);

ALTER TABLE Event_EventAdmin ADD CONSTRAINT PK_Event_EventAdmin
PRIMARY KEY (idEvent, idEventAdmin);

ALTER TABLE Event_EventAdmin ADD CONSTRAINT FK_Event
```

```
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Event_EventAdmin ADD CONSTRAINT FK_Event
FOREIGN KEY (idEventAdmin) REFERENCES EventAdmin (idEventAdmin);

/------------Event_EventCategory------------/
CREATE TABLE Event_EventCategory (
idEvent integer NOT NULL,
idEventCategory integer NOT NULL
);

ALTER TABLE Event_EventAdmin ADD CONSTRAINT PK_Event_EventAdmin
PRIMARY KEY (idEvent, idEventCategory);

ALTER TABLE Event_EventCategory ADD CONSTRAINT FK_Event
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Event_EventCategory ADD CONSTRAINT FK_EventCategory
FOREIGN KEY (idEventCategory) REFERENCES EventCategory (idEventCategory);

/------------Event_Member------------/
CREATE TABLE Event_Member(
idEvent integer NOT NULL,
idMember integer NOT NULL
);

ALTER TABLE Event_Member ADD CONSTRAINT PK_Event_Member
PRIMARY KEY (idEvent, idMember);

ALTER TABLE Event_Member ADD CONSTRAINT FK_Event
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Event_Member ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

/------------Friend------------/
CREATE TABLE Friend(
idF1 integer NOT NULL,
idF2 integer NOT NULL,
accepted boolean NULL
);

ALTER TABLE Friend ADD CONSTRAINT PK_Friend
PRIMARY KEY (idF1, idF2);

ALTER TABLE Friend ADD CONSTRAINT FK_Event
FOREIGN KEY (idF1) REFERENCES Member (idMember);

ALTER TABLE Friend ADD CONSTRAINT FK_Member
FOREIGN KEY (idF2) REFERENCES Member (idMember);

/* Create Primary Keys, FK, Indexes, Uniques, Checks */

/------------------- Comment----------------------/
ALTER TABLE Comment ADD CONSTRAINT PK_Comment
PRIMARY KEY (idComment);

ALTER TABLE Comment ADD CONSTRAINT FK_Event
```

```
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Comment ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

/---------- Community----------/
ALTER TABLE Community ADD CONSTRAINT PK_Community
PRIMARY KEY (idCommunity);

ALTER TABLE CommunityAdmin ADD CONSTRAINT PK_Member
PRIMARY KEY (idCommunityAdmin);

ALTER TABLE CommunityCategory ADD CONSTRAINT PK_CommunityCategory
PRIMARY KEY (idCommunityCategory);

ALTER TABLE Event ADD CONSTRAINT PK_Event
PRIMARY KEY (idEvent);

ALTER TABLE EventAdmin ADD CONSTRAINT PK_Member
PRIMARY KEY (idEventAdmin);

ALTER TABLE EventCategory ADD CONSTRAINT PK_EventCategory
PRIMARY KEY (idEventCategory);

ALTER TABLE Invoice ADD CONSTRAINT PK_Invoice
PRIMARY KEY (idInvoice);

ALTER TABLE Member ADD CONSTRAINT PK_Member
PRIMARY KEY (idMember);

ALTER TABLE Notification ADD CONSTRAINT PK_Notification
PRIMARY KEY (idNotification);

ALTER TABLE Report ADD CONSTRAINT PK_Report
PRIMARY KEY (idReport);

ALTER TABLE Ticket ADD CONSTRAINT PK_TicketidEventCategory
PRIMARY KEY (idTicket);

ALTER TABLE Ticket ADD CONSTRAINT FK_TicketType
FOREIGN KEY idTicketType REFERENCES TicketType(idTicketType);

ALTER TABLE Ticket ADD CONSTRAINT FK_Member
FOREIGN KEY idMember REFERENCES Member(idMember);

ALTER TABLE TicketType ADD CONSTRAINT PK_TicketType
PRIMARY KEY (idTicketType);

/* Create Foreign Key Constraints */
ALTER TABLE Community_Member ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

ALTER TABLE Community_Member ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);

ALTER TABLE Community_CommunityCategory ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);
```

```
ALTER TABLE Community_CommunityCategory ADD CONSTRAINT FK_Category
FOREIGN KEY (idCommunityCategory) REFERENCES CommunityCategory (idCommunityCategory);

ALTER TABLE Community_CommunityAdmin ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);

ALTER TABLE Community_CommunityAdmin ADD CONSTRAINT FK_CategoryAdmin
FOREIGN KEY (idCommunityAdmin) REFERENCES CommunityAdmin (idCommunityAdmin);

ALTER TABLE Ticket ADD CONSTRAINT FK_TicketType
FOREIGN KEY (idTicketType) REFERENCES TicketType (idTicketType);

ALTER TABLE Ticket ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

ALTER TABLE Report ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);

ALTER TABLE Report ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

ALTER TABLE Report ADD CONSTRAINT FK_Event
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Report ADD CONSTRAINT FK_Comment
FOREIGN KEY (idComment) REFERENCES Comment (idComment);

ALTER TABLE Notification ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);

ALTER TABLE Notification ADD CONSTRAINT FK_Member
FOREIGN KEY (idMember) REFERENCES Member (idMember);

ALTER TABLE Notification ADD CONSTRAINT FK_Event
FOREIGN KEY (idEvent) REFERENCES Event (idEvent);

ALTER TABLE Notification ADD CONSTRAINT FK_Comment
FOREIGN KEY (idComment) REFERENCES Comment (idComment);

ALTER TABLE Event ADD CONSTRAINT FK_Community
FOREIGN KEY (idCommunity) REFERENCES Community (idCommunity);
```