

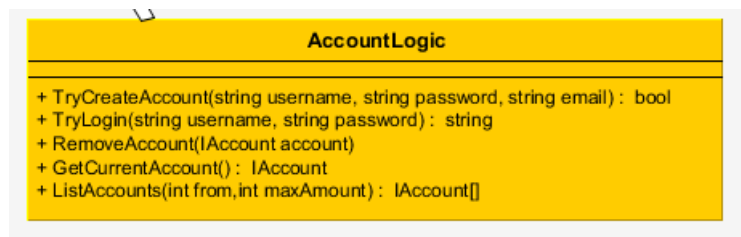
Assinment41

pebj, smot

October 24, 2014

1 OOSE

1.1 Specify relevant pre- and postconditions



The screenshot shows a UML class diagram for a class named **AccountLogic**. The class is represented by a yellow rectangle with a black border. Inside the rectangle, the class name **AccountLogic** is written in bold black text at the top. Below the name, there is a list of five public methods, each preceded by a '+' sign. The methods are: `TryCreateAccount(string username, string password, string email) : bool`, `TryLogin(string username, string password) : string`, `RemoveAccount(IAccount account)`, `GetCurrentAccount() : IAccount`, and `ListAccounts(int from, int maxAmount) : IAccount[]`.

```
/* To remove an account you have to be logged in as the account or  
* be a moderator.*/
```

```
context AccountLogic::RemoveAccount(account) pre:  
    account.equals(Season.CurrentAccount()) or  
    Season.CurrentAccount()->IsModerater()
```

```
/*The accounts references will become null  
* The account will not be accessable in the database*/
```

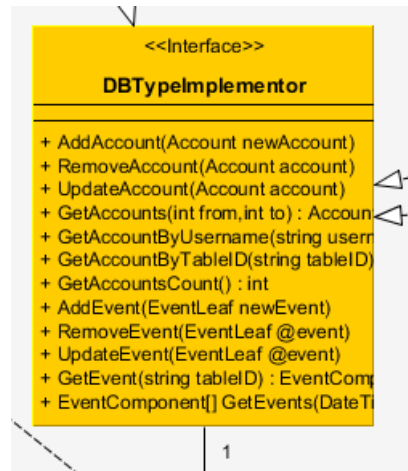
```
context AccountLogic::RemoveAccount(account) post:  
    account.equals(null) and  
    OnlineContext.GetAccount(username) is ObjectNotFoundException
```

```
/*The username may not exist in the database  
* you have to be online*/
```

```
context AccountLogic::TryCreateAccount(username, password, email) pre:  
    no Season.GetAccount(username) and  
    OnlineContext->isOnline()
```

```
/*An account in the database will be created  
* It will be possible to login with the new account afterwards*/
```

```
context AccountLogic::TryCreateAccount(username, password, email) post:  
    Season.GetAccount(username).username.equals(username)  
    AccountLogic->TryLogin(username,password)
```



/*The events TableID may not be ""

* The given events TableID is valid*/

context DBTypeImplementor::UpdateEvent(event) **pre:**

not event.TableID.equals("") and

GetEvent(event.itemIndex).equals(a event)

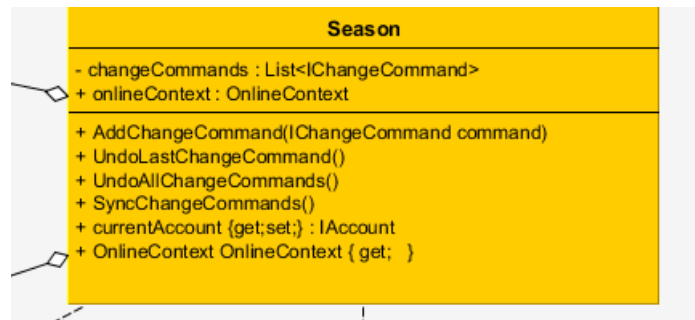
/*An updated event' content is equals the event from GetEvent(self.tableID);

* There will be a corresponding command in season*/

context DBTypeImplementor::UpdateEvent(event) **post:**

GetEvent(event.tableID).equals(event)

Season.changeCommands->contains(UpdateEvent)



/*The login account is a moderater*/

context Season::GetAccounts(from,to) **pre:**

CurrentAccount()->IsModerater() and

from > 0 and

to < GetAccountsCount()

/*You will get an array containing all accounts from "from" to "to"*/

context Season::GetAccounts(from,to) **post:**

GetAccounts().length = to - from

EventLogic
+ AddEvent(OnlineSeason season,string description, DateTime dateFrom, DateTime dateTo, Notification[] notifications) : EventLeaf + GetEventById(OnlineSeason season,string name) : EventComponent + GetEvents(OnlineSeason season,DateTime from, DateTime to) : EventComponent[] + RemoveEvent(OnlineSeason season,EventComponent anEvent) + UpdateEvent(OnlineSeason season,EventComponent anEvent) : boo + JoinComponentsToComposite(OnlineSeason season,params EventComponent[] components) : boo + AddComponentsToComposite(OnlineSeason season,EventsComposite composite,params EventComponent[] components) : bool + GetSharedEventLink(OnlineSeason season,EventComponent component) : string

```

/*components contains at lest 2 EventComponent
* An account is loggetin*/
context EventLogic::JoinComponentsToComposite(OnlineSeason, components)
pre:
    components.length >= 2 and
    not OnlineSeason.CurrentAccount().equals(null)

/*you will get an EventsComposite contaning all EventComponents
* in components
* There will be a corosponding command in OnlineSeason*/
context EventLogic::JoinComponentsToComposite(OnlineSeason, components)
post:
    JoinComponentsToComposite(OnlineSeason, components).eventComponents-
    >contains(components[0])
    OnlineSeason.changeCommands->contains(UpdateEvent)

```

1.2 Specify invariants

```

/*There can't be two referencing to the same EventComponent's*/
/*in an EventsComposite*/
context EventsComposite inv:
    eventComponents.contains(x => x).where(x == event).Count() = 1

/* An account have to be logged in to use EventLogic*/
context EventLogic inv:
    OnlineSeason.CurrentAccount() != null

/* If there is internet connection, OnlineContext uses Online. Otherwise,
* OnlineContext uses Offline*/
context OnlineContext inv:
    if isOnline() then
        _syncStrategy is Online
    else
        _syncStrategy is Offline

```

1.3 Object Model

The Object model below is a splitted up version of the original so as to have space for it in this document. The complete one image version of the Object

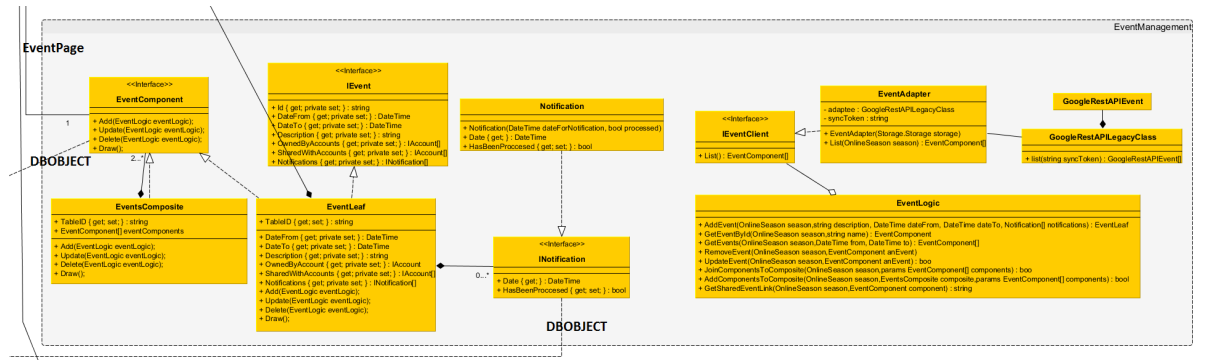


Table 1: EventManagement

Model can be seen in the same folder as this document.

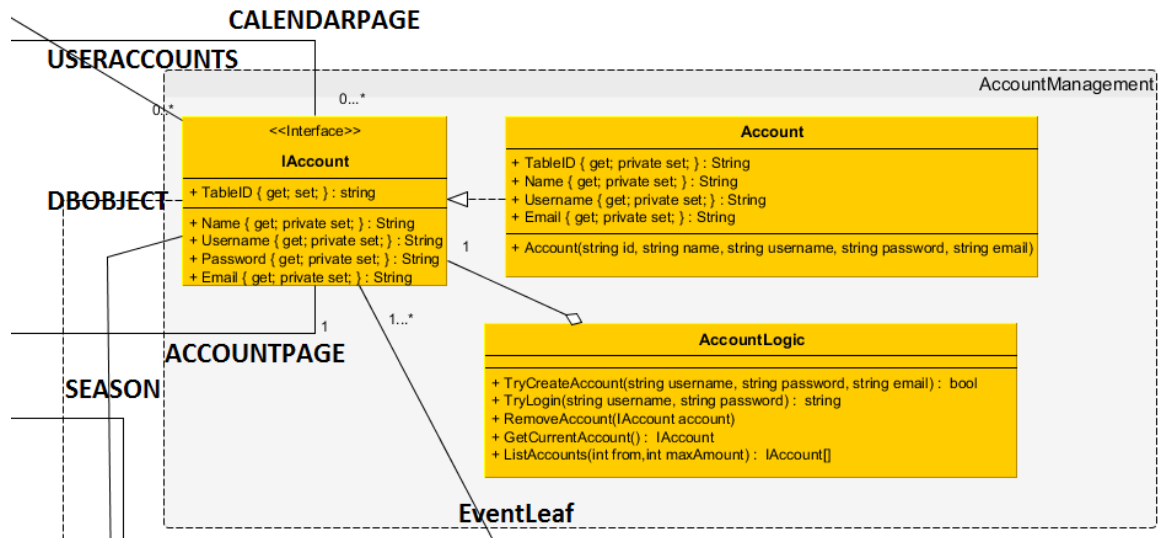


Table 2: AccountManagement

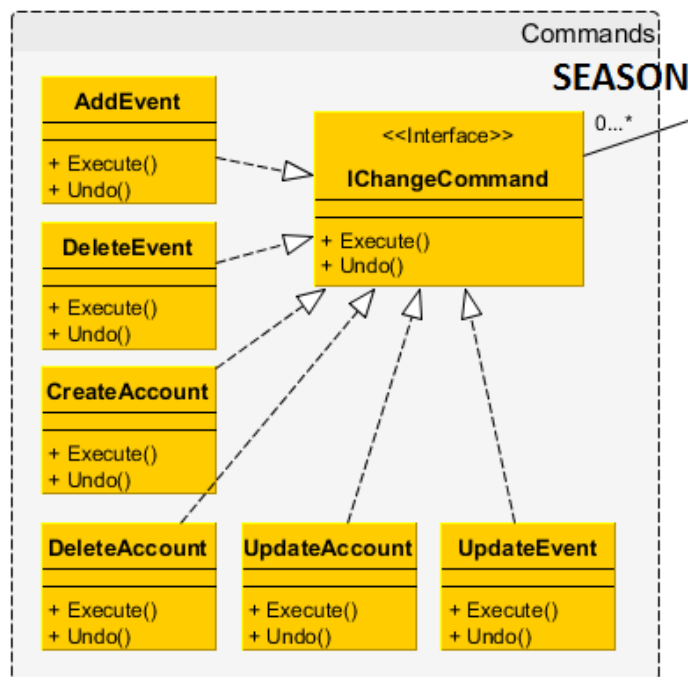


Table 3: Commands

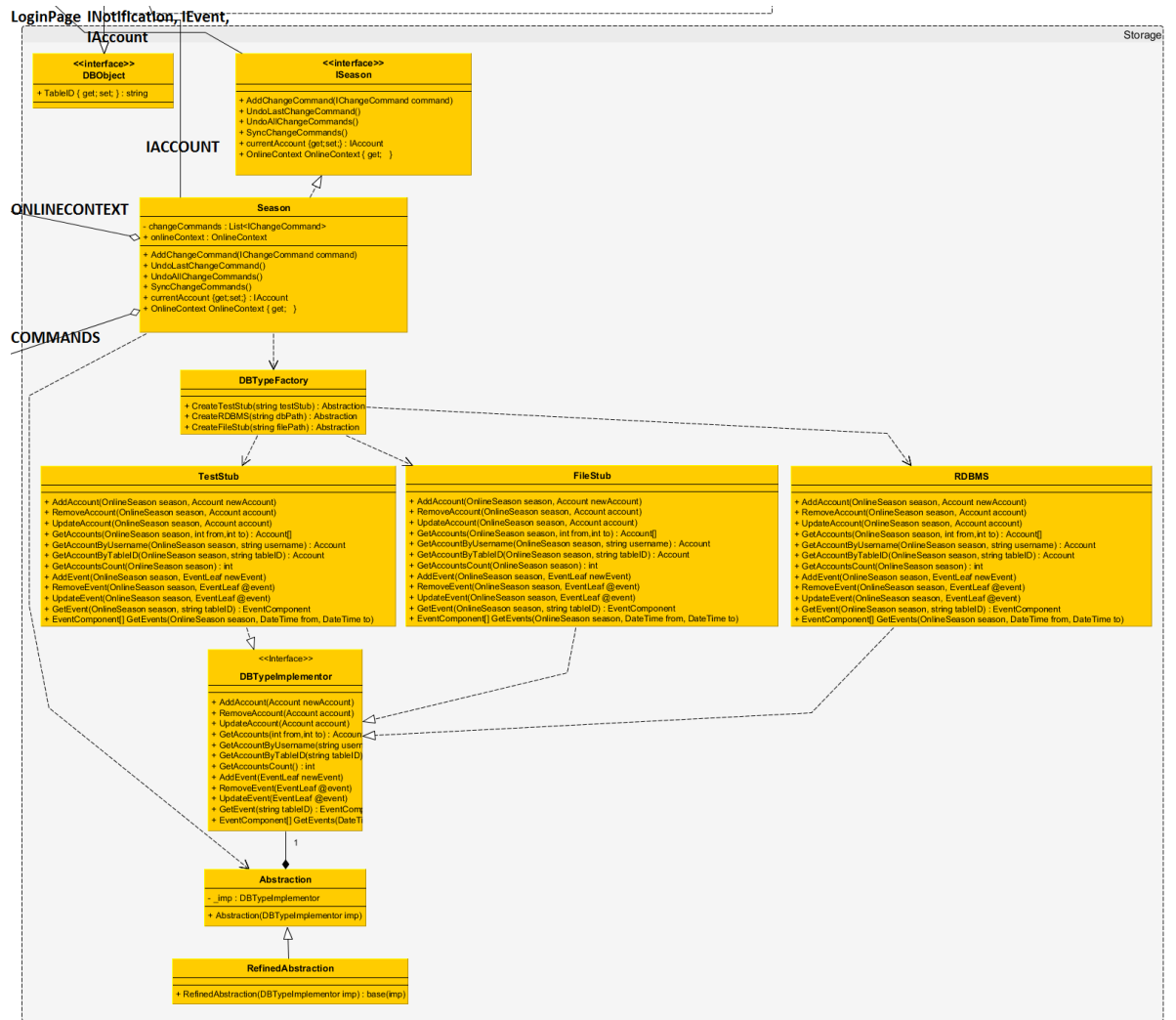


Table 4: Storage

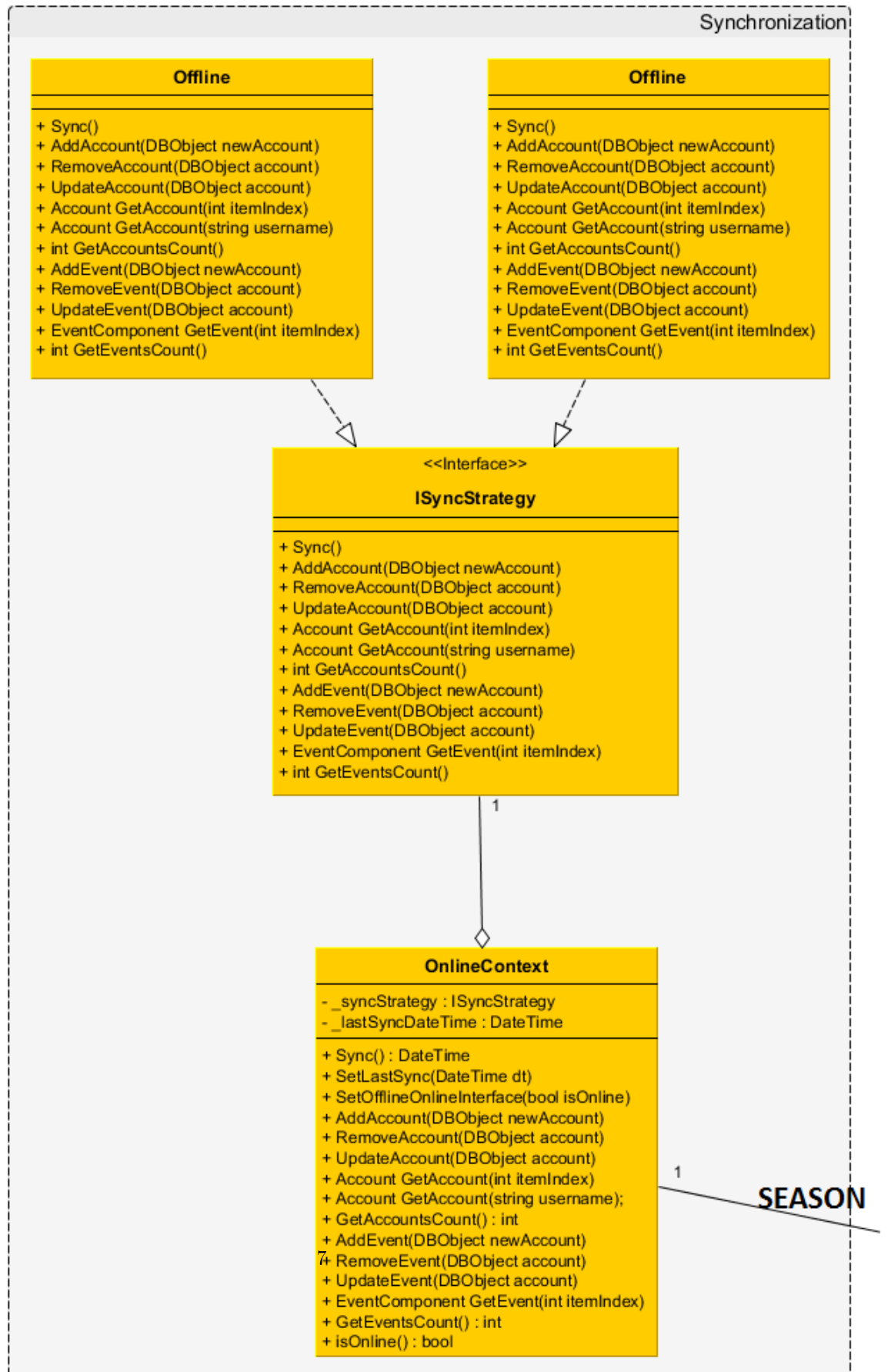


Table 5: Synchronization

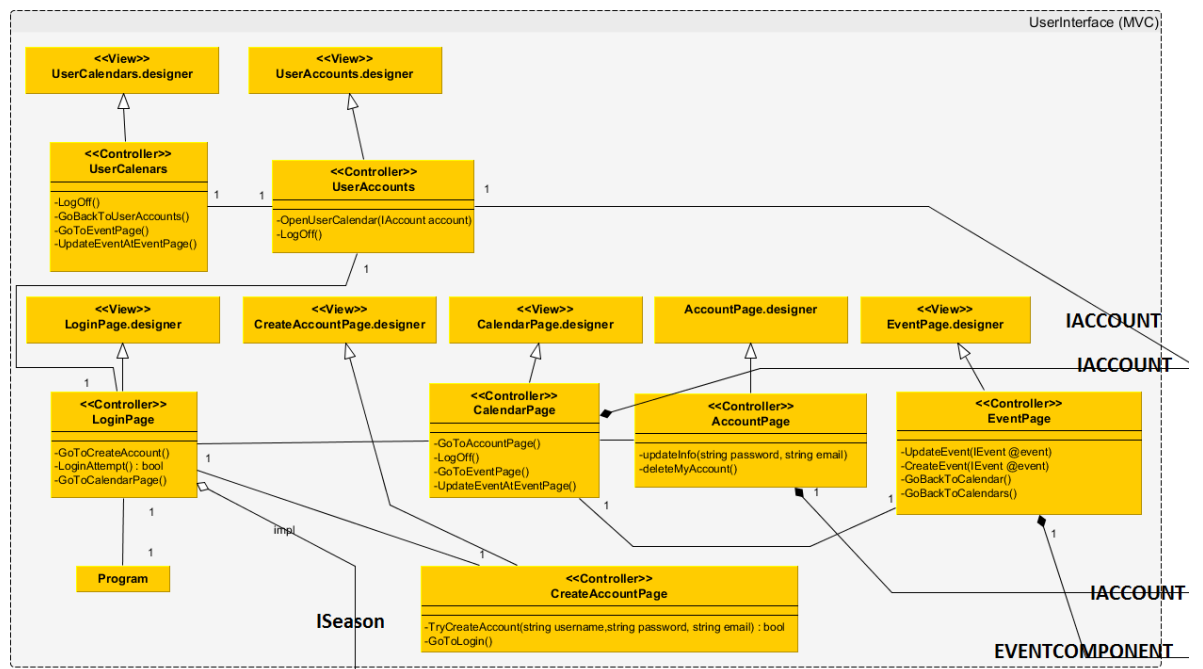


Table 6: UserInterface _MVC

Accounts

Primary Key

<u>ID</u>	Username	Password	Email	isModerator
0	Stinus	Stinus123	smot@itu.dk	False
1	Patrick	Patrick123	pebj@itu.dk	True

Candidate Key

Candidate Key

Candidate Key

Events

Primary Key

<u>ID</u>	DateFrom	DateTo	Description	OwnerAccount	PartOfComposite
0	24-10-2014	26-10-2014	Going to the park.	0	null
1	24-10-2014	26-10-2014	LAN Party at ITU	1	0
2	27-10-2014	31-10-2014	Schooltime	1	0

Foreign Key

Foreign Key

EventShares

Primary Key

<u>ID</u>	UserID	EventID
0	0	1

Foreign Key

Foreign Key

Composite

Primary Key

<u>ID</u>	PartOfComposite
0	1
1	null

Foreign Key

Notifications

Primary Key

<u>ID</u>	EventID	Date	HasBeenProcessed
0	1	25-10-2014	False

Foreign Key

Table 7: OO-to-RDBMS mapping