calendar-app

Calendar app for cs-302 project

Final Project Report

Steven Lin

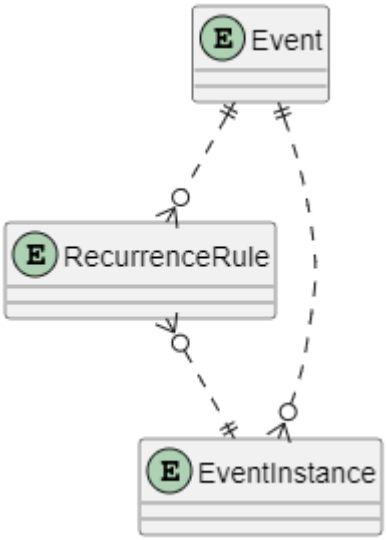
Calendar App

Introduction

The goal of this project is to build a calendar management applicaion, with funcionalities similar to the likes of outlook calendar and google calendar. The goal and expected challengee were clear from the beginning: Management of recurrent events. In order for events to repeat, rules must be set up to create and keep track of recurring events. As there are many different recurrence patterns I wanted to implement, I expected most of my work to be around managing them.

First Iteration

The initial design involved 3 tables, as shown below:



Events store detailed information.

Events have associated RecurrenceRules, and RecurrenceRules contain information used to build EventInstances.

EventInstances can join with Event to retrieve the detailed informations.

EventInstances of a recurring event can be altered independently, while still being attached to a parent event.

Automated triggers create EventInstances upon creation of an Event and its associated RecurrenceRule. Event and RecurrenceRule are expected to always be created together

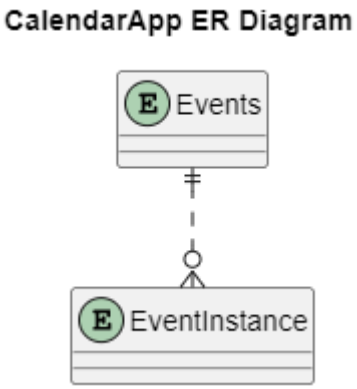
This approach worked well enough, however there was one issue: When createing EventInstances, all instances are created at once. This means it cannot work for events with no defined end date, as it would attempt to create infinite event instances. As such, this version requires users to input a defined end date for recurring events.

One issue was that I was constantly confusing ER diagram and logical design.

Second Iteration

In the latter half of the project, I reworked the triggers to make EventInstances generate in real time when needed. A new stored procedure is added, updateDisplay, which take startdate and enddate as input, and re-generate all the instances. This is made with GUI in mind, where a user would select a range of dates to display.

Another change was made that got rid of the RecurrenceRule table. Its content was mostly integrated into Event.

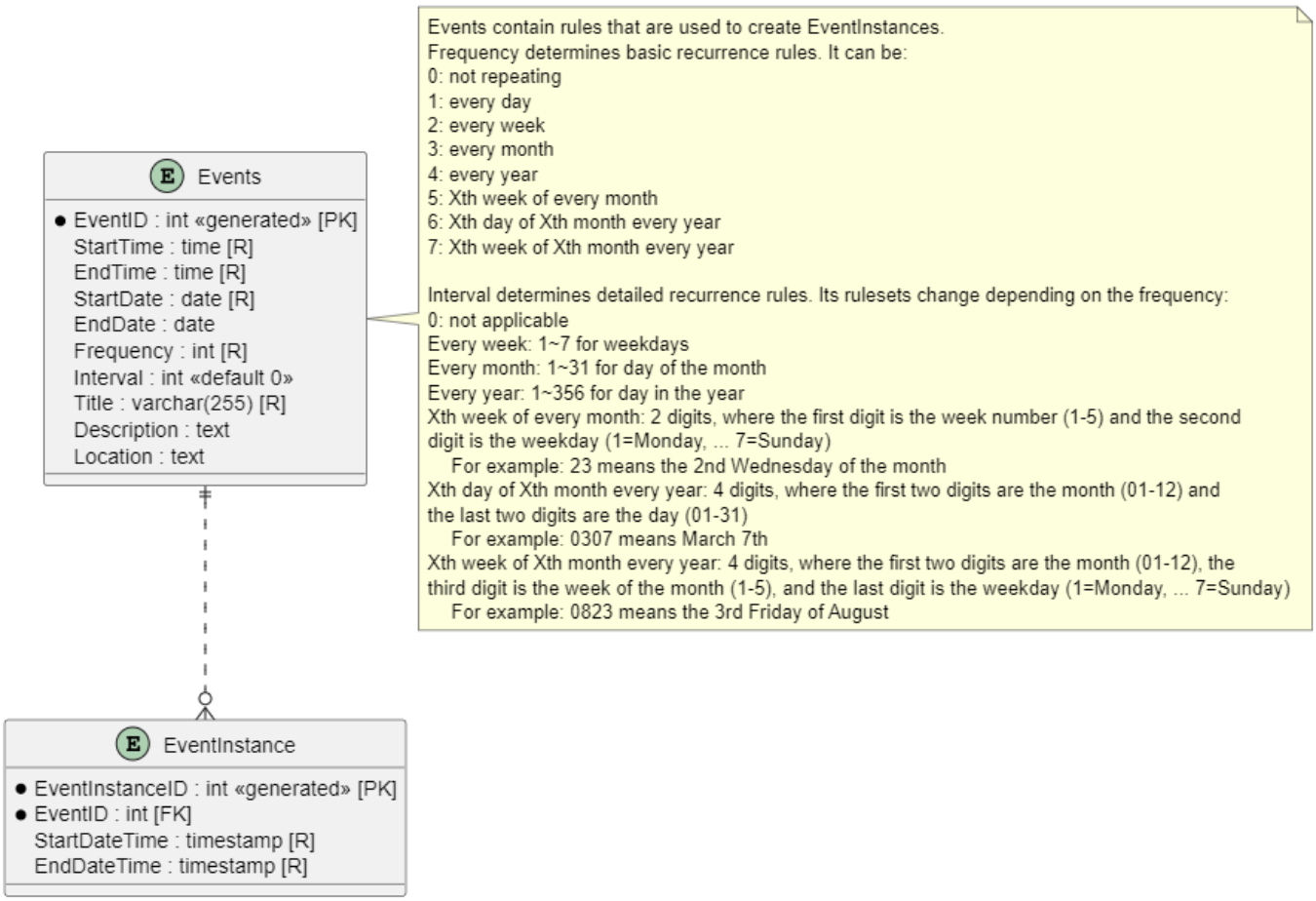


This change was made to simplify instance generation logic. As EventInstance would be refreshed constantly, performance and simplicity was paramount. Simplifying the link between events and instances improved performance.

In addition, the original need for RecurrenceRule table was to help keep track of altered event instances, where I can alter an instance of an event, or alter "this and all following events" in a chain of recurring events. The plan was to use nested recurrence rules and flags in EventInstances to keep track of such changes. However the move to using dyanmic generation of EventInstances made the logic for keeping track of altered instances too complex, and I was unable to come up with a solution. I cut the feature along with the RecurrenceRule table to keep the complexity under control.

Logic for incrementing time

Much effort was spent on designing and implementing the rulesets used to create EventInstances. Detail as follow:



The implementation for case 0 to 4 were simple: just increment by one day/week/month/year. Implementation of case 5 to 7 involved parsing the Interval attribute into date, then splitting them into day/week/month/year before processing addition, then adding them back together to form the new date.

Conclusion

Compared to other projects presented in class, this one is more software design than database design, as the database itself was kept simple. That said. the project as a whole was still a good practice in writing database functions.