Университет ИТМО

Факультет программной инженерии и компьютерной техники

Курсовая работа по "Информационные системы и базы данных". Часть №3

Работу выполнили:

Хузин Рамиль P33112

Иманзаде Фахри P33111

Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург
2021 год

Описание предметной области

- …as the Talmud says!

*- I don’t give a sh*t what he says. Okay?*

Источник: https://sinagoga.jeps.ru/sinagoga/chto-takoe-sinagoga.html

Информационная система в виде виртуальной синагоги, в которой участники общины могут совершать различные религиозные обряды, молитвы (иначе говоря: мероприятие, известен его тип, описание, дата и время проведения), также имеется библиотека, к которой имеет доступ каждый общинник, она обязательно должна содержать ряд религиозных книг: Пятикнижие, Мишну, Талмуд. О книге известно название, краткое содержание.

В синагоге принято по субботам и праздникам выступать с лекциями на темы глав Торы или по проблемам еврейского законодательства, такую беседу проводит наиболее знающий член общины или приглашенный раввин. Также, по субботам собираются группы для изучения Торы. Такие собрания проводятся в определенном отделении синагоги, также указывается максимальное количество посетителей. В зависимости от количества посетителей рассчитывается количество пищи.

У членов общины имеются свои роли, такие как: рядовой общинник, раввин, хазан, шамаш, габай. Каждая роль обладает требованиями к общиннику и определенными полномочиями.

Синагога может иметь разные размеры и быть построенной в любом архитектурном стиле. Она состоит из комнат (например: вестибюль, основное помещение, балкон, эзрат нашим – помещение для женщин). Синагога обладает храмовыми

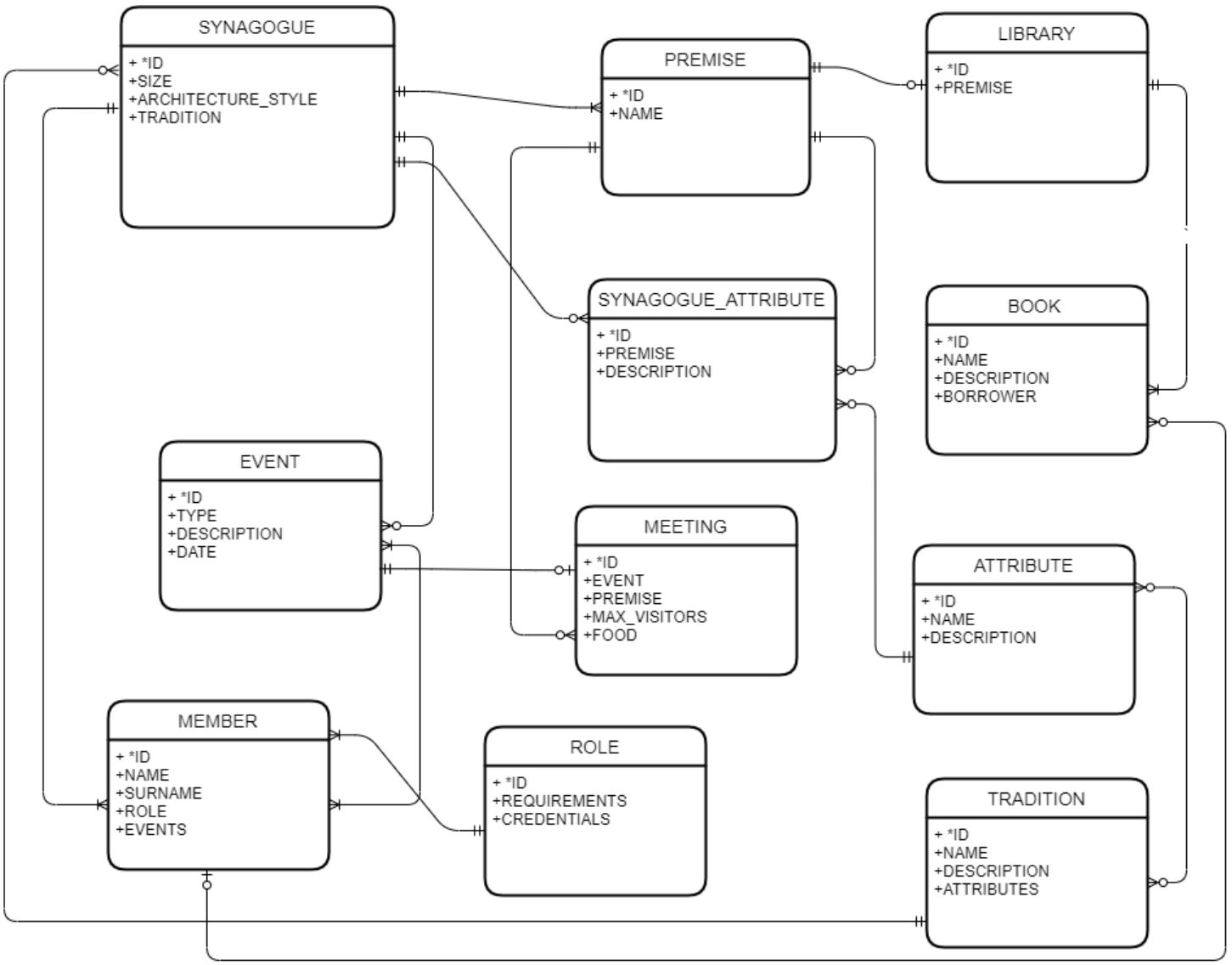
атрибутами, о которых известно местоположение (помещение), название, назначение.

Синагога также определяется принадлежностью к определенной традиции (общности): ашкеназской или сефардской, ашкеназская делится в свою очередь на две группы: хасидскую и нехасидскую. Для каждой традиции указываются характерные храмовые атрибуты.

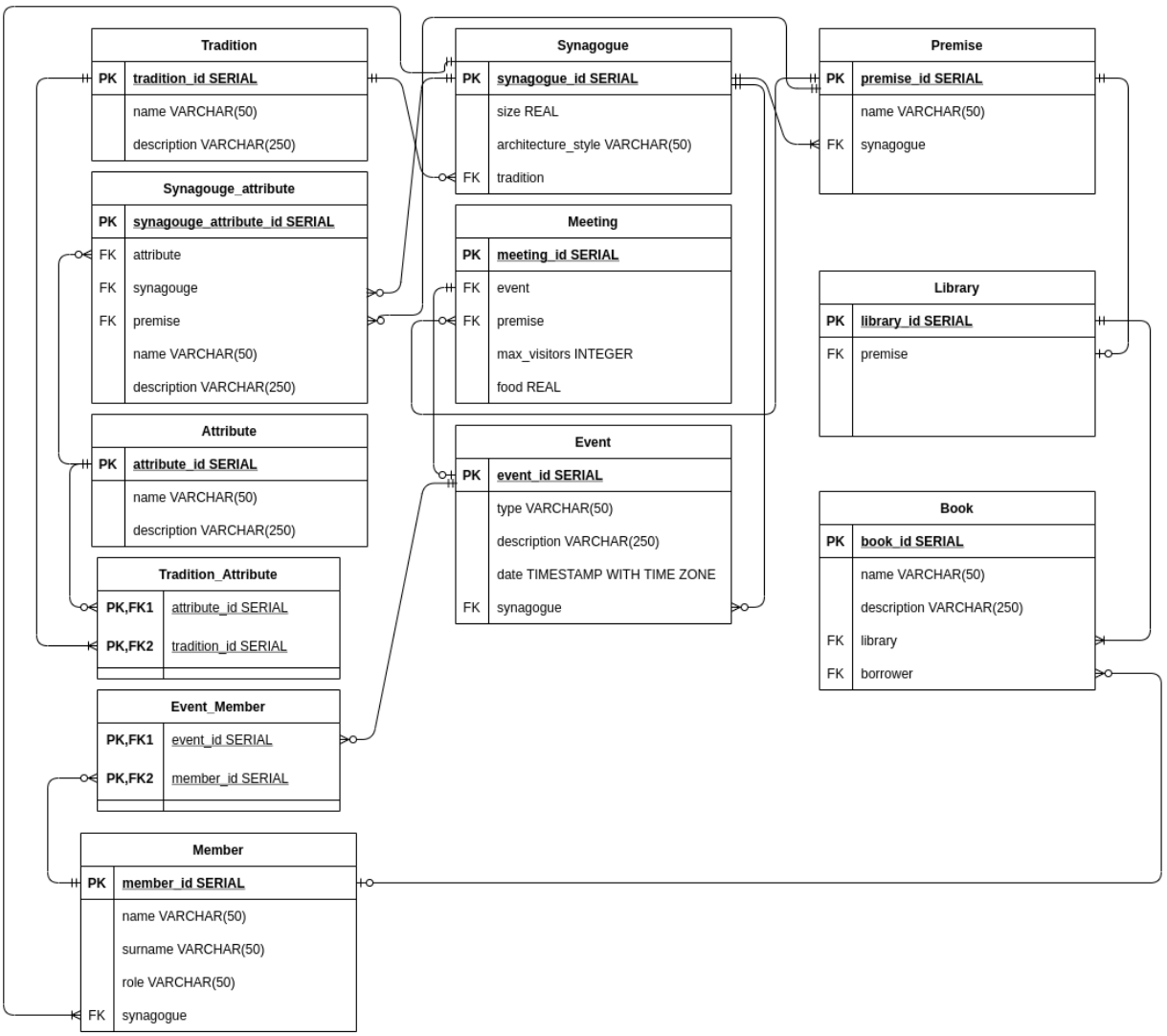
Бизнес-процессы

1. Общинник может запланировать мероприятие, указав при этом приглашенных людей
2. Общинник может взять книгу из библиотеки
3. Общинник(user) может вернуть книгу в библиотеку
4. Габай (руководитель общины) может запланировать собрание общины
5. Общинник может забронировать место на собрание

Инфологическая модель



Даталогическая модель



Создание таблиц

```
create table tradition
(
    tradition_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(250)
);

create table synagogue
(
  synagogue_id SERIAL PRIMARY KEY,
  size REAL NOT NULL CHECK ( size >= 0 AND size <= 100000),
  architecture_style VARCHAR(50),
  tradition INTEGER REFERENCES tradition ON DELETE SET NULL
);

create table premise
(
  premise_id SERIAL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  synagogue INTEGER REFERENCES synagogue ON DELETE CASCADE NOT NULL
);

create table library
(
  library_id SERIAL PRIMARY KEY,
  premise INTEGER REFERENCES premise ON DELETE CASCADE NOT NULL
);

create table member
(
    member_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
    role VARCHAR(50) NOT NULL,
    synagogue INTEGER REFERENCES synagogue ON DELETE SET NULL
);

create table book
(
    book_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(250) NOT NULL,
    library INTEGER REFERENCES library ON DELETE CASCADE NOT NULL,
    borrower INTEGER REFERENCES member ON DELETE SET NULL,
    is_available BOOLEAN
);
```

```
create table event
(
    event_id SERIAL PRIMARY KEY,
    type VARCHAR(50) NOT NULL,
    description VARCHAR(250) NOT NULL,
    date TIMESTAMP WITH TIME ZONE NOT NULL,
    synagogue INTEGER REFERENCES synagogue ON DELETE CASCADE NOT
NULL
);

create table meeting
(
    meeting_id SERIAL PRIMARY KEY,
    event INTEGER REFERENCES event ON DELETE CASCADE NOT NULL,
    premise INTEGER REFERENCES premise ON DELETE SET NULL,
    max_visitors INTEGER NOT NULL,
    food REAL
);

create table attribute
(
    attribute_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(250) NOT NULL
);

create table synagogue_attribute
(
    synagogue_attribute_id SERIAL PRIMARY KEY,
    attribute INTEGER REFERENCES attribute ON DELETE CASCADE NOT
NULL ,
    synagogue INTEGER REFERENCES synagogue ON DELETE CASCADE NOT
NULL ,
    premise INTEGER REFERENCES premise ON DELETE SET NULL ,
    name VARCHAR(50) NOT NULL ,
    description VARCHAR(250) NOT NULL
);

create table tradition_attribute
(
    attribute_id SERIAL REFERENCES attribute ON DELETE CASCADE,
    tradition_id SERIAL REFERENCES tradition ON DELETE CASCADE,
    primary key (attribute_id, tradition_id)
);

create table event_member
(
    event_id SERIAL REFERENCES event ON DELETE CASCADE,
    member_id SERIAL REFERENCES member ON DELETE CASCADE,
    primary key (event_id, member_id)
);
```

Заполнение таблиц

```
INSERT INTO tradition (name, description)
VALUES
       ('Ашкеназская', 'Названа по имени ашкеназов - субэтнической
группы евреев, сформировавшихся в Центральной Европе, представители
говорят на идише'),
       ('Сефардская', 'Названа по имени ашкеназов - субэтнической
группы евреев, сформировавшихся на Пиренейском полуострове из
потоков миграции иудеев внутри Римской империи, а затем внутри
Халифата');

INSERT INTO synagogue (size, architecture_style, tradition)
VALUES
       (750, 'неомавританский', 1);

INSERT INTO premise (name, synagogue)
VALUES
       ('зал', 1), ('библиотека', 1), ('эзрат нашим', 1),
('вестибюль', 1);

INSERT INTO library (premise)
VALUES (2);

INSERT INTO member (name, surname, role, synagogue)
VALUES
       ('Лев', 'Ландау', 'раввин', 1),
       ('Бенедикт', 'Спиноза', 'хазан', 1),
       ('Леонид', 'Канторович', 'шамаш', 1),
       ('Эрих', 'Фромм', 'габай', 1),
       ('Егор', 'Остряков', 'прихожанин', 1);

INSERT INTO book (name, description, library, borrower)
VALUES
    ('Пятикнижие', 'пять первых книг Танаха и Ветхого Завета: Бытие,
Исход, Левит, Числа и Второзаконие. Составляет первую часть Танаха,
именуемую также Торой', 1, NULL),
    ('Талмуд', 'свод правовых и религиозно-этических положений
иудаизма, охватывающий Мишну и Гемару в их единстве', 1, NULL),
    ('Шулхан арух', 'кодекс практических положений устного закона,
составленный в XVI веке раввином Йосефом Каро', 1, 5);

INSERT INTO event (type, description, date, synagogue)
VALUES
       ('Ханука', 'Праздничная молитва', TIMESTAMP WITH TIME ZONE
'2021-11-28 17:00:00+03', 1),
       ('Иврит Торы', 'учимся читать и понимать тексты на святом
языке в оригинале', TIMESTAMP WITH TIME ZONE '2021-11-02
19:00:00+03', 1),
```

```
       ('Пурим с подробностями', 'Обсуждение истории праздника',
TIMESTAMP WITH TIME ZONE '2022-02-22 19:00:00+03', 1);

INSERT INTO meeting (event, premise, max_visitors, food)
VALUES (3, 1, 100, 50);

INSERT INTO attribute (name, description)
VALUES
       ('Арон акодеш', 'Шкаф или ниша, где хранятся свитки Торы'),
       ('Бима', 'Возвышение в центре, с которого читается Тора, на
нем установлен стол для свитка'),
       ('Амуд', 'Пюпитр между бимой и арон кодеш');

INSERT INTO synagogue_attribute (attribute, synagogue, premise,
name, description)
VALUES (2, 1, 1, 'Бима', 'Возвышение в центре');

INSERT INTO tradition_attribute (attribute_id, tradition_id)
VALUES (3, 1);

INSERT INTO event_member (event_id, member_id)
VALUES
       (1, 1), (1, 2), (1, 5),
       (2, 2), (2, 5),
       (3, 1), (3, 2), (3, 3), (3, 4), (3, 5);

INSERT INTO event_member values (3, 6);
```

Очищение таблиц

```
DELETE FROM tradition;
DELETE FROM synagogue;
DELETE FROM member;
DELETE FROM attribute;
```

Удаление таблиц

```
drop table if exists book;

drop table if exists library;

drop table if exists meeting;

drop table if exists synagogue_attribute;

drop table if exists premise;

drop table if exists tradition_attribute;

drop table if exists attribute;

drop table if exists event_member;

drop table if exists member;

drop table if exists event;

drop table if exists synagogue;

drop table if exists tradition;
```

Функции и триггеры

```
create or replace function check_max_visitors_function() returns
trigger as $check_max_visitors_function$
declare
    max_visitors           integer;
    current_visitors_count integer;
BEGIN
    SELECT count(*)
    INTO current_visitors_count
    FROM event_member
    WHERE event_member.event_id = NEW.event_id;
    SELECT meeting.max_visitors INTO max_visitors FROM meeting
    WHERE event = NEW.event_id;
    IF (current_visitors_count = max_visitors) THEN
        RAISE EXCEPTION 'Visitors are limited, try next time!';
    END IF;

    return NEW;
END;

$check_max_visitors_function$ LANGUAGE plpgsql;


create trigger check_max_visitors
    before INSERT on event_member
    for each row execute procedure check_max_visitors_function();

create function book_was_borrowed_function() returns trigger as
$book_was_borrowed_function$
BEGIN
    IF borrower is null THEN
        UPDATE book
        SET is_available = true
        WHERE book_id = NEW.book_id;
    ELSE
        UPDATE book
        SET is_available = false
        WHERE book_id = NEW.book_id;
    END IF;
    RETURN new;
END;

$book_was_borrowed_function$ LANGUAGE plpgsql;

create trigger cell_was_updated
    after UPDATE of borrower on book
    for each row execute procedure book_was_borrowed_function();
```

```
create function update_role(in our_member_id integer, in new_role
varchar(50)) returns text as $$
BEGIN
    update member
    set role = new_role
    where member_id = our_member_id;
    return 'New role successfully updated!';
END;

$$ LANGUAGE plpgsql;
```

Индексы

Популярные запросы

1. Поиск запланированных мероприятий или собраний с указанием временных пределов
2. Поиск книг

Скрипты

1. Btree index для атрибута **date** в таблице **Event (**часто необходимо искать запланированные мероприятия в каких-то временных пределах)

```
create index if not exists event_date on event (date);
```

2. Hash index для атрибута **event** в таблице **Meeting** (информация для поиска собраний содержится в таблицах **Event,** поэтому необходим хеш индекс для быстрого сопоставления найденной event таблицы и соответствующей ей meeting таблицы)

```
create index if not exists meeting_event_id
on meeting using hash (event);
```

3. Btree index для атрибута **name** в таблице **Book** (часто используется поиск книги по ее названию, btree индекс ускоряет поиск по подстроке)

```
create index if not exists book_name on book (name);
```

TODO:
добавить name для таблицы Synagogue

добавить в таблицу Member password, login

добавить dropdown roles на фронте для отображения на фронте и отправка в теле POST

Технологии:

Frontend - React, Redux, Saga, Chakra Ui
Backend - Spring Boot, Jpa, Postgre Sql

Страницы(план):

1. Login Page - для авторизации
   login
   password
   auth/login
2. Registration Page - для регистраиции
   2.1. ролей (Общинник, Габай) статично на фронте
   2.2 GET synagogue(name)
   2.3 login
   2.4 password
   2.5 name
   2.6 surname
   auth/register
3. Main Page - информация про синагогу, переходы по страничкам и ткд
4. Library Page - информация про книги, фильтр available books,
   4.1 все книги - GET /books - получаем все книги и информацию, если есть тот кто взял книгу пишем в ячейки таблицы его usermame и ячейка available становится false, если нет чела который взял книгу то ячейка borrower бывает пустой, а ячейка available бывает true (без токена).
   4.2 все доступные книги - ПЕРХОД -> Кнопка забронировать книгу -> /books?available=true получу список книг которые доступны и можно забронировать.
   любая кнопка бронирование книги - Post books/borrow - с токеном
   4.3 мои книги - информация GET /books/my с токеном (заголовки Autorization-token)
   рядом с книгой будет кнопка вернуть книгу Post books/return - с токен
5. User Profile - информация про user из таблицы Member /member?id=userid

Changes in DB:
1. Added name attribute into Synangogue table