Практика по базам данных
ОТЧЕТ

Смирнов Александр
241 группа

Предметная область: «Библиотека»

Содержание

ОПИСАНИЕ СИСТЕМЫ

Требования

Система предназначена для учета выдачи и возврата книг в библиотеке редких книг.

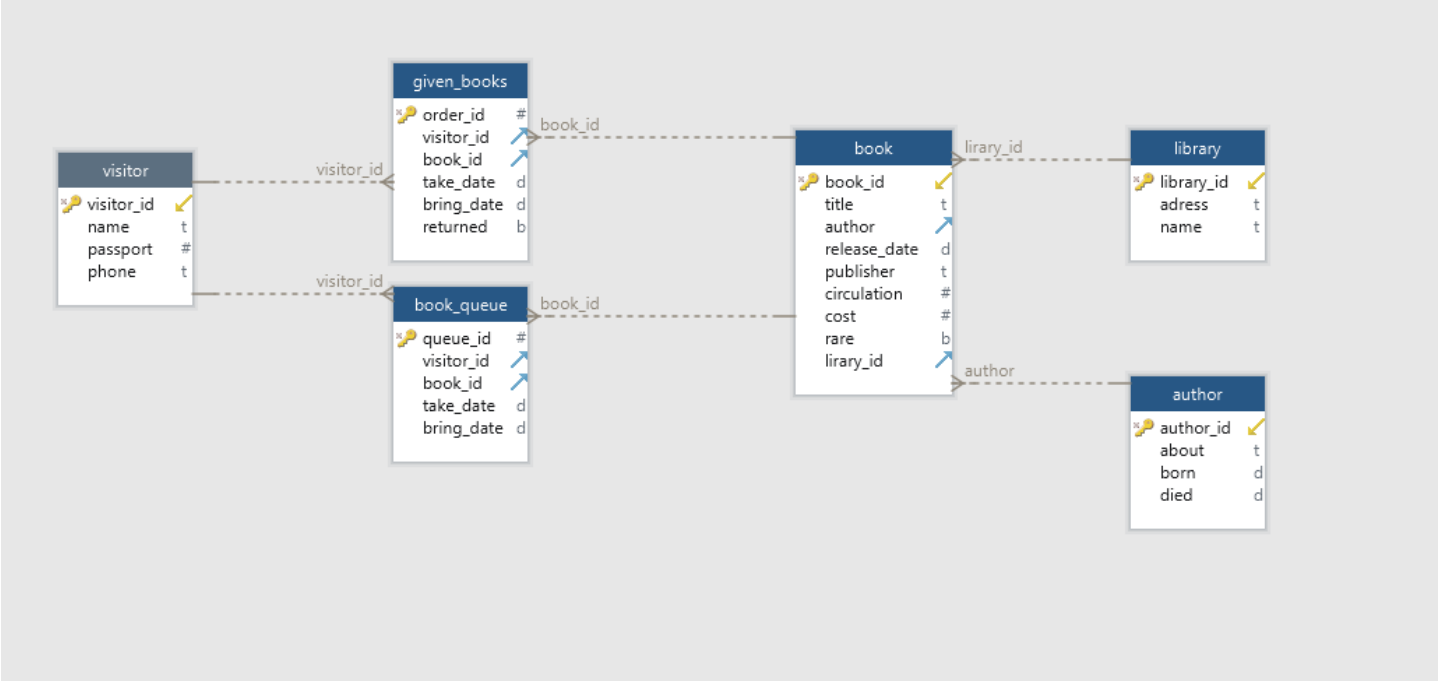
Сотрудник библиотеки регистрирует выданные посетителю и возвращенные им книги. Данные о выданных книгах сохраняются в истории выдач. Если нужная посетителю книга в данный момент находится на руках, сотрудник библиотеки регистрирует запрос на нее и ставит его в очередь ожидания.

Если посетитель хочет зарезервировать за собой книгу на определенный период времени, сотрудник библиотеки проверяет, нет ли заказов на резервирование этой книги на этот период и, если их нет, резервирует книгу для данного посетителя на данный период.

Информация, которую нужно хранить о посетителях – минимальна, она включает ФИО, паспортные и контактные данные.

Информация о книгах включает не только название и авторов, но и год выпуска, издательство, тираж, оценочная стоимость. Особенностью библиотеки редких книг является то, что каждое произведение здесь в единственном экземпляре.

Модель данных



Функциональность

Серверная часть

Хранимые процедуры\функции	*Реализация*	*Комментарии*
Добавление нового посетителя	AddNewVisitor	(имя, возраст, паспорт, телефон)
Обработка возврата книги	BookReturned	(номер заказа)
Количество книг в библиотеке	GetNumberOfBooks	(номер библиотеки)
Добавление заказа		
Добавление книги		
Добавление библиотеки		
Удаление запроса на книгу		
Количество поситетелей в очереди		
…	…	…

Триггеры	*Реализация*	*Комментарии*
Запрет на удаление посетителя	TR_del_visitor	для сохранения пользователей
Запрет на удаление книжки (кол-во = 0)	TR_del_book	для сохранения наименований
Выставление редкости книги если она одна	TR_ins_book_rare	для отслеживания редких книг
Контроль повторного добавления клиента		откат при совпадении ФИО и адреса
Невозвоможность выдачи заказа клиенту с долгом		
После снижения редкости цена понижается в два раза		
Переноса из лист ожидания в отданные книги при наличии экземпляра		
Перенос книги из одной библиотеки в другую если суммарная стоимость книг в одной библиотеке в 2 раза больше, чем в другой		для равномерного распределения бюджетов библиотек
…	…	…

Представления	*Реализация*	*Комментарии*
Книги со стоимостью выше заданной	V_books	всё про книгу
Выданные книжки и кто их взял	V_given_books_with_visitors	(название книжки, имя посетителя)
Реестр авторов и их книжек	V_books_authors	(название книжки, имя автора)
Должники и дата возврата		(имя, время возврата, наименование книги)
Авторы и суммарная стоимость книг		(имя автора, суммарная стоимость)
Имена посетителей и количество взятых книг		(имя, кол-во книг)
Издатель и количество экземпляров		(название издателя, кол-во экземпляров)
Количество книг в библиотеке		(библиотека, кол-во книг)
…	…	…

Клиентская часть

Экранные формы основные	*дополнительные*	*Реализация (запрос)*	*Что здесь можно использовать из серверной части*
Реестр посетителей			
	Новый посетитель		AddNewVisitor()
	Изменить данные посетителей		
Реестр книг			
	Добавить книгу		
	Изменить книгу		
	Фильтр	(04) Названия книг на букву (05) Книги, которые ещё не брали (06) Библиотеки с книгами в них (08) Суммарная стоимость всех книг (10) Все редкие книжки	
	Список книг		
Реестр авторов		(01) Данные об авторах	
	Добавить		
	Изменить		
Реестр взявших книгу		(09) Выданные книжки с датой возврата (11) Общий список из очереди на книжку и выданных книжек (12) Все заказы человека по ID	
	Должники	(07) Выбрать всех, кто не вернул книгу	
	Взявшие больше n книг		
	Принёс книгу		
	Выдать книгу		
	Добавить		
	Изменить		
	Читатели в очереди	(03) Читатели в очереди	
Служебные запросы		(02) Читатели не бравшие книг	

СКРИПТЫ

Серверная часть

Хранимые процедуры и функции

```
-- (1) Добавление нового посетителя

CREATE PROCEDURE AddNewVisitor
@var_name AS VARCHAR(100),
@var_age AS INT,
@var_passport AS INT,
@var_phone AS VARCHAR(15)
AS
BEGIN
  DECLARE @var_new_visitor_id INTEGER;
  SELECT @var_new_visitor_id = MAX(visitor.visitor_id) + 1 FROM visitor;
  INSERT INTO visitor(visitor_id, name, age, passport, phone)
        VALUES(@var_new_visitor_id, @var_name, @var_age, @var_passport, @var_phone);
END;
-- Пример вызова процедуры:
--EXECUTE AddNewVisitor @var_name='Пенская Таисия Андреевна', @var_age = 29, @var_passport=8563829,
@var_phone='89117635645'


-- (2) Обработка возврата книги (перевод в состояние возвращено)

CREATE PROCEDURE BookReturned
@var_order_id AS INTEGER
AS
BEGIN
  UPDATE given_books SET returned = 1
       WHERE given_books.order_id = @var_order_id

END;
-- Пример вызова процедуры:
--EXECUTE BookReturned @var_order_id = 4;


-- (3) Количество книг в библиотеке по ID

CREATE FUNCTION GetNumberOfBookInLibrary (@library_id INT)
  RETURNS INTEGER
  BEGIN
      DECLARE @BookNumber INTEGER;
      SELECT @BookNumber = COUNT(*) FROM  book WHERE book.lirary_id = @library_id
     RETURN @BookNumber;
END;
-- Пример вызова функции:
--SELECT dbo.GetNumberOfBookInLibrary (3)
```

Триггеры

```
-- (1) При добавлении автора с пустым полем описания, оно будет заменено на «Описание отсутствует»

CREATE TRIGGER TR_ins_author ON author FOR INSERT
    AS
    BEGIN
        IF (SELECT about FROM INSERTED) = ''
            UPDATE author SET about = 'Описание отсутствует' WHERE author.author_id = (SELECT author_id
FROM INSERTED)
    END
--INSERT INTO author(author_id, name, about, born, died) VALUES (6, 'Кен Элтон Кизи', '', '1935-09-17',
'2001-11-10');


-- (2) Кол-во книг 0 вместо удаления книги

CREATE TRIGGER TR_del_book ON book INSTEAD OF DELETE
AS
   UPDATE book SET book.circulation = 0
       WHERE book_id = (SELECT book_id FROM DELETED)
--DELETE FROM book WHERE book_id = 1


-- (3) Если у добавляемой книги 1 экземпляр, то обозначить её редкой

CREATE TRIGGER TR_ins_book_rare ON book FOR INSERT
    AS
    BEGIN
        IF (SELECT circulation FROM INSERTED) = 1
            UPDATE book SET rare = 1 WHERE book.book_id = (SELECT book_id FROM INSERTED)
    END
--INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (8, 'Идиот', 2, '1868-01-01', 'Русский вестник', 5, 228000, 0, 1)
```

Представления

```
-- (1) Книги со стоимостью выше заданной
CREATE VIEW V_books AS
       SELECT *
       FROM book
       WHERE cost > 500000
--SELECT * FROM V_books

-- (2) Выданные книжки и кто их взял
CREATE VIEW V_given_books_with_visitors
AS
  SELECT visitor.name, book.title
    FROM given_books JOIN visitor ON given_books.visitor_id = visitor.visitor_id
                     JOIN book ON book.book_id = given_books.book_id
--SELECT * FROM V_given_books_with_visitors

-- (3) Авторы и книжки
CREATE VIEW V_books_authors
AS
  SELECT book.title, author.name
    FROM book JOIN author ON book.author = author.author_id
SELECT * FROM V_books_authors
```

Клиентская часть

(запросы для экранных форм и отчетов)

```sql
-- (01) Данные об авторах, включая количество книг

SELECT author.name, author.about, book_number
   FROM author JOIN (SELECT author.author_id, COUNT(*) AS book_number
                        FROM author JOIN book ON author.author_id = book.author
                        GROUP BY author.author_id) ttt
               ON author.author_id= ttt.author_id



-- (02) Читатели, ни разу не бравшие книгу за всё время

SELECT visitor.name from visitor
       WHERE NOT EXISTS (SELECT * FROM given_books
                            WHERE given_books.visitor_id = visitor.visitor_id)



-- (03) Читатели, которые стоят в очереди, отсортированные по возрасту

SELECT visitor.name, visitor.age FROM visitor
          WHERE EXISTS (SELECT * FROM book_queue
              WHERE book_queue.visitor_id = visitor.visitor_id)
       ORDER BY age DESC


-- (04) Название книги на П и М с их авторами

SELECT book.title, author.name
  FROM book
       JOIN author ON author.author_id= book.author
    WHERE book.title LIKE 'П%' OR book.title LIKE 'М%'
    ORDER BY book.title;


-- (05) Книги, которых ещё никогда никогда не брали

SELECT DISTINCT book.title FROM given_books, book
       WHERE book.book_id NOT IN (SELECT given_books.book_id FROM given_books)



-- (06) Библиотеки с книгами в них
SELECT library.name, book.title
       FROM book
            RIGHT JOIN library ON library.library_id = book.lirary_id



-- (07) Выбрать всех тех, кто не вернул книгу.
SELECT visitor.name
    FROM visitor
       JOIN given_books ON given_books.visitor_id = visitor.visitor_id
       WHERE returned = 0


--(08) Суммарная стоимость всех книг в Московской библиотеке

SELECT SUM(book.cost) 'Суммарная стоимость всех книг' FROM book WHERE book.lirary_id = 1
```

```sql
-- (09) Выданные книжки с датой возврата в промежутке

SELECT * FROM given_books
      WHERE bring_date BETWEEN '2017-01-01' AND '2017-07-11'
      ORDER BY 5 DESC;


-- (10) Выбрать все редкие книжки

SELECT * FROM book WHERE book.rare = 1
      ORDER BY book.cost DESC


-- (11) Общий список, состоящий из очереди на книжки и из выданных книжек

SELECT bring_date, 'Время возврата отданных книжек' FROM given_books
UNION
SELECT bring_date, 'Время возврата забронированных книжек' FROM book_queue



-- (12) Все заказы человека с ID = 2

SELECT * FROM given_books WHERE visitor_id = 2


-- (13) Все посетители, вернувшие больше одной книжки


SELECT visitor_id, SUM(CAST(returned AS INT)) AS Returned_book_number
FROM given_books
GROUP BY visitor_id
HAVING SUM(CAST(returned AS INT)) > 1
```

Создание и заполнение базы данных

```
-- CREATE DATABASE library
-- GO
-- USE library

-------------------------------------------------------------------------------------------------------------------
-----------------------------------------------
-- Создание таблиц и PK
-------------------------------------------------------------------------------------------------------------------
-----------------------------------------------

CREATE TABLE author (
	author_id		int  NOT NULL  ,
	name			varchar(50),
	about			varchar(200)    ,
	born			date    ,
	died			date    ,
	CONSTRAINT pk_authors_authors_id PRIMARY KEY ( author_id )
 )

CREATE TABLE visitor(
	visitor_id		int  NOT NULL  ,
	name			varchar(100)    ,
	age			int      ,
	passport		int    ,
	phone			varchar(15)    ,
	CONSTRAINT pk_person_id PRIMARY KEY ( visitor_id )
 )

CREATE TABLE library (
	library_id		int  NOT NULL  ,
	adress			varchar(100)    ,
	name			varchar(100)    ,
	CONSTRAINT pk_library_library_id PRIMARY KEY ( library_id )
 )

CREATE TABLE book (
	book_id			int  NOT NULL  ,
	title			varchar(50)    ,
	author			int NOT NULL    ,
	release_date		date    ,
	publisher		varchar(50)    ,
	circulation		int    ,
	cost			int    ,
	rare			bit    ,
	lirary_id		int    ,
	CONSTRAINT pk_book_book_id PRIMARY KEY ( book_id )
 )

CREATE TABLE given_books (
	order_id		int  NOT NULL  ,
	visitor_id		int    ,
	book_id			int    ,
	take_date		date    ,
	bring_date		date    ,
	returned		bit    ,
	CONSTRAINT pk_given_books_order_id PRIMARY KEY ( order_id )
 )

CREATE TABLE book_queue (
	queue_id		int  NOT NULL  ,
	visitor_id		int    ,
	book_id			int    ,
	take_date		date    ,
	bring_date		date    ,
	CONSTRAINT pk_book_queue_queue_id PRIMARY KEY ( queue_id )
```

```
)

-----------------------------------------------------------------------------------------------------------
-----------------------------------------------
-- Создание FK
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------


ALTER TABLE book ADD CONSTRAINT fk_author FOREIGN KEY ( author ) REFERENCES author( author_id ) ON
DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE book ADD CONSTRAINT fk_library FOREIGN KEY ( lirary_id ) REFERENCES library( library_id )
ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE book_queue ADD CONSTRAINT fk_visitor FOREIGN KEY ( visitor_id ) REFERENCES visitor(
visitor_id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE book_queue ADD CONSTRAINT fk_book FOREIGN KEY ( book_id ) REFERENCES book( book_id ) ON
DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE given_books ADD CONSTRAINT fk_visitor_given_books FOREIGN KEY ( visitor_id ) REFERENCES
visitor( visitor_id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE given_books ADD CONSTRAINT fk_book_given_books FOREIGN KEY ( book_id ) REFERENCES book(
book_id ) ON DELETE NO ACTION ON UPDATE NO ACTION;


-----------------------------------------------------------------------------------------------------------
-----------------------------------------------
-- Заполнение таблиц тестовыми данными
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------


INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (1, 'Смирнов Александр Львович', 15,
40137591,'89119727982');
INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (2, 'Литвинов Степан Сергеевич', 22,
63758264,'89210938070');
INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (3, 'Жилкин Фёдор Игоревич', 12,
92758264,'88005553535');
INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (4, 'Амрани Илиас Магомедович', 44,
82647834,'89212283645');
INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (5, 'Филиппов Марк Дмитриевич', 60,
09473625,'83645346756');
INSERT INTO visitor(visitor_id, name, age, passport, phone) VALUES (6, 'Бодкин Вячеслав Сергеевич', 22,
75864354,'89992347654');

INSERT INTO library(library_id, adress, name) VALUES (1, 'Москва, р-н Арбат, ул. Воздвиженка, 3/5',
'Российская государственная библиотека');
INSERT INTO library(library_id, adress, name) VALUES (2, 'Санкт-Петербург, Садовая ул., 18',
'Российская национальная библиотека');
INSERT INTO library(library_id, adress, name) VALUES (3, 'Санкт-Петербург, Сенатская площадь, дом 3',
'Президентская библиотека имени Б. Н. Ельцина');

INSERT INTO author(author_id, name, about, born, died) VALUES (1, 'Нестор Летописец', 'Древнерусский
летописец', '1056-01-01', '1114-01-01');
INSERT INTO author(author_id, name, about, born, died) VALUES (2, 'Фёдор Михайлович Достоевский',
'Русский писатель, мыслитель, философ и публицист. Член-корреспондент Петербургской АН с 1877 года',
'1821-10-30', '1881-01-28');
INSERT INTO author(author_id, name, about, born, died) VALUES (3, 'Александр Сергеевич Пушкин',
'Русский поэт, драматург и прозаик, заложивший основы русского реалистического направления, критик и
теоретик литературы, историк, публицист', '1799-05-26', '1837-01-29');
INSERT INTO author(author_id, name, about, born, died) VALUES (4, 'Карл Генрих Маркс', 'Немецкий
философ, социолог, экономист, писатель, поэт, политический журналист, общественный деятель', '1818-05-
05', '1883-03-14');
```

```
INSERT INTO author(author_id, name, about, born, died) VALUES (5, 'Джон Рональд Руэл Толкин',
'Английский писатель и поэт, переводчик, лингвист, филолог', '1892-01-03', '1973-09-02');

INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (1, 'Повесть временных лет', 1, '1110-01-21', 'Перо', 1, 1000000, 1, 2);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (2, 'Руслан и Людмила', 3, '1820-01-01', 'Сын отечества', 100, 728000, 0, 2);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (3, 'Манифест Коммунистической партии', 4, '1848-02-21', 'Союз справедливых', 1, 215000, 1, 3);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (4, 'Борис Годунов', 3, '1831-01-01', 'Игра слов', 5000, 500, 0, 1);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (5, 'Капитал. Критика политической экономии', 4, '1867-01-01', 'Dietz Verlag Berlin', 1, 825000,
1, 2);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (6, 'Властелин колец', 5, '1955-01-01', 'George Allen & Unwin', 500, 378000, 0, 3);
INSERT INTO book(book_id, title, author, release_date, publisher, circulation, cost, rare, lirary_id)
VALUES (7, 'Преступление и наказание', 2, '1866-01-01', 'Русский вестник', 10000, 550000, 0, 1);

INSERT INTO book_queue(queue_id, visitor_id, book_id, take_date, bring_date) VALUES (1, 3, 2, '2018-01-
22', '2018-01-28');
INSERT INTO book_queue(queue_id, visitor_id, book_id, take_date, bring_date) VALUES (2, 1, 3, '2018-02-
10', '2018-02-13');
INSERT INTO book_queue(queue_id, visitor_id, book_id, take_date, bring_date) VALUES (3, 1, 4, '2018-06-
22', '2018-06-29');
INSERT INTO book_queue(queue_id, visitor_id, book_id, take_date, bring_date) VALUES (4, 4, 5, '2018-09-
01', '2018-10-01');

INSERT INTO given_books(order_id, visitor_id, book_id, take_date, bring_date, returned) VALUES (1, 2,
5, '2017-09-12', '2017-10-12', 1);
INSERT INTO given_books(order_id, visitor_id, book_id, take_date, bring_date, returned) VALUES (2, 2,
4, '2017-01-01', '2017-01-12', 1);
INSERT INTO given_books(order_id, visitor_id, book_id, take_date, bring_date, returned) VALUES (3, 3,
3, '2017-02-12', '2017-04-21', 1);
INSERT INTO given_books(order_id, visitor_id, book_id, take_date, bring_date, returned) VALUES (4, 4,
2, '2017-05-13', '2017-06-05', 0);
INSERT INTO given_books(order_id, visitor_id, book_id, take_date, bring_date, returned) VALUES (5, 6,
1, '2017-09-11', '2017-10-17', 0);



-------------------------------------------------------------------------------------------------
-- Создание индексов
-------------------------------------------------------------------------------------------------


CREATE INDEX author_index ON author (born);
CREATE INDEX book_index ON book (rare);
CREATE INDEX visitor_index ON visitor (name);


-------------------------------------------------------------------------------------------------
-- Удаление таблиц
-------------------------------------------------------------------------------------------------


/*
DROP TABLE book_queue
DROP TABLE given_books
DROP TABLE visitor
DROP TABLE book
DROP TABLE library
DROP TABLE author
*/

-------------------------------------------------------------------------------------------------
```