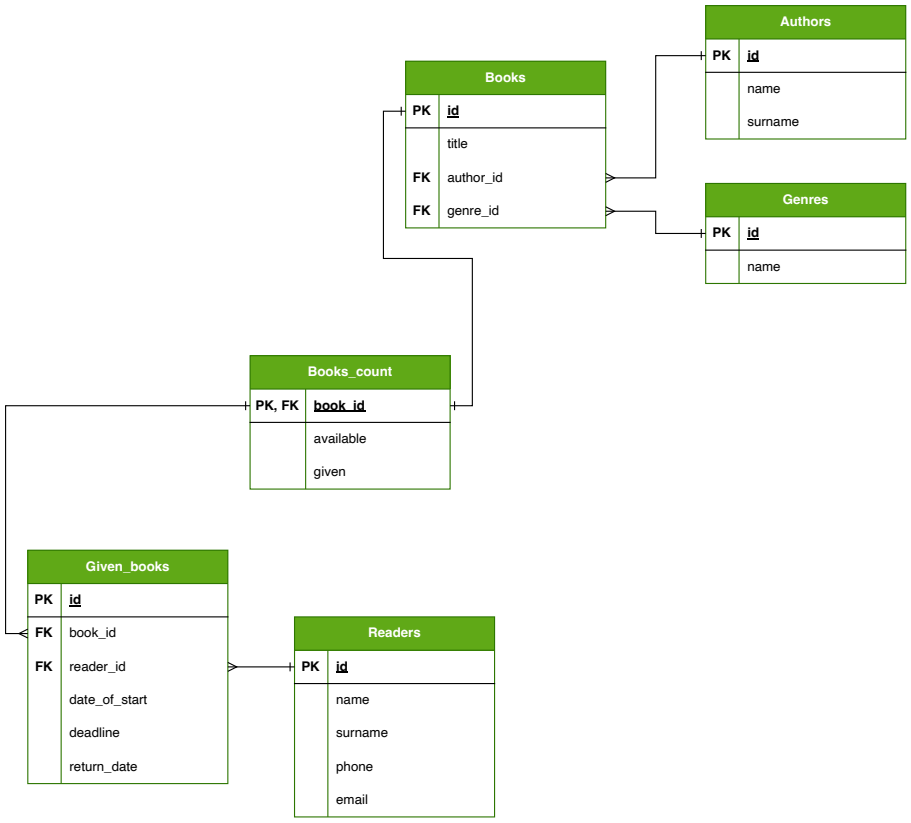
Домашняя работа после первого технического собеседования.
Волков Никита

Спроектировать БД для учета книг в библиотеке в 3НФ (типа снежинки):
1. ER-диаграмма (таблицы, атрибуты, связи между таблицами, ключи таблиц)
2. Краткое описание таблиц, их изменений/обновлений и ограничений

Решение

ER-диаграмма спроектированной БД выглядит следующим образом:



Здесь роль центральной связующей таблицы (таблицы фактов), в соответствии со схемой «Снежинка», выполняет таблица **Books_count** — выбрана в качестве таковой, так как содержит в себе основную информацию, требуемую по заданию — учет книг.

Диаграмма содержит 6 таблиц:
1. **Authors** — содержит информацию об авторах книг
2. **Genres** — информацию о жанрах
3. **Books** — информацию о книгах, содержащихся в библиотеке

4. **Books_count** — информацию учета книг: конкретная книга, количество экземпляров, доступных в библиотеке, количество экземпляров, выданных на дом
5. **Given_books** — информацию о выданных на дом книгах: id выдачи, id выданной книги, id читателя, дата выдачи, срок выдачи (в днях), дата возврата (NULL, если книгу еще не вернули)
6. **Readers** — информацию о читателях

Ограничения в таблицах:

1. **Authors**:
   name NOT NULL,
   surname NOT NULL

2. **Genres**:
   name NOT NULL

3. **Books**:
   title NOT NULL,
   author_id NOT NULL,
   genre_id NOT NULL,
   все внешние ключи: ON DELETE CASCADE, ON UPDATE CASCADE

4. **Books_count**:
   available DEFAULT 0 CHECK (available >= 0),
   given DEFAULT 0 CHECK (given >= 0),
   внешний ключ: ON DELETE CASCADE, ON UPDATE CASCADE

5. **Given_books**:
   book_id NOT NULL,
   reader_id NOT NULL,
   date_of_start DEFAULT CURRENT_TIMESTAMP,
   return_date DEFAULT NULL,
   внешние ключи:
   ON DELETE CASCADE, ON UPDATE CASCADE для book_id,
   и ON UPDATE CASCADE для reader_id

6. **Readers**:
   name NOT NULL,
   surname NOT NULL

Кроме того, для интерактивного изменения информации о количестве книг в таблице **Books_count** в случае взятия книги на дом или, наоборот, возврата, добавлены триггеры, автоматические изменяющие информацию в таблице **Books_count**:

Триггер при новом взятии книги:

```
DELIMITER //
CREATE TRIGGER new_give
AFTER INSERT
ON Given_books
FOR EACH ROW
BEGIN
        UPDATE Books_count
    SET available = available - 1,
                given = given + 1
        WHERE Books_count.book_id = NEW.book_id;
END
//
```

Триггер при возврате книги:

```
CREATE TRIGGER new_return
BEFORE UPDATE
ON Given_books
FOR EACH ROW
BEGIN
        UPDATE Books_count
    SET available = available + 1,
                given = given - 1
        WHERE Books_count.book_id = NEW.book_id AND
ISNULL(OLD.return_date) AND NOT ISNULL(NEW.return_date);
END
//
DELIMITER ;
```

Изменять можно любую таблицу, но основные изменения происходят в таблице **Given_books** — изначально предполагается, что дата возврата книги неизвестна, поэтому изначально ей присваивает NULL, но при возврате значение обновляется, а вслед за ним идет обновление в таблице **Books_count**.

Задание на SQL к этой БД

Написать запросы к спроектированной БД, который вернет следующую информацию:

1. Топ 5 книг, которые были взяты наибольшее количество раз в прошлом месяце и количество раз, которое их брали.
2. ФИО читателей и количество просроченных (книги, которые не вернули в срок) ими разных (если один читатель брал книгу дважды и оба раза просрочил - нужно учитывать 1 раз) книг в жанре «Детектив"

Решение

Прежде всего покажу первоначально заполненные таблицы:

Authors

id	name	surname
1	Дарья	Донцова
2	Федор	Достоевский
3	Уолтер	Айзексон
4	Теодор	Драйзер
5	Александр	Пушкин

Genres

id	name
1	Детектив
2	Роман
3	Проза
4	Биография

Books

id	title	author_id	genre_id
1	Дезертир рая	1	1
2	Блоха на балу	1	1
3	Амур с гранатой	1	1
4	Преступление и нак…	2	2
5	Финансист	4	2
6	Американская траге…	4	2
7	Евгений Онегин	5	2
8	Стив Джобс	4	4

Books_count

book_id	available	given
1	11	3
2	20	1
3	4	4
4	31	4
5	23	2
6	25	1
7	52	2
8	12	0

Given_books

id	book_id	reader_id	date_of_start	deadline	return_date
1	1	1	2024-12-04 00:00:00	14	2024-12-08
2	1	1	2024-12-15 00:00:00	7	2024-12-25
3	1	1	2024-12-26 00:00:00	14	NULL
4	4	3	2024-12-04 00:00:00	14	2024-12-08
5	4	2	2024-12-12 00:00:00	14	2024-12-18
6	3	1	2024-12-04 00:00:00	14	2024-12-08
7	3	4	2024-12-05 00:00:00	7	2024-12-15
8	5	2	2024-12-01 00:00:00	14	2024-12-08
9	5	4	2024-12-04 00:00:00	14	2024-12-14
10	7	1	2024-12-04 00:00:00	14	2024-12-08
11	7	2	2024-12-05 00:00:00	7	2024-12-17
12	6	4	2024-12-04 00:00:00	14	2024-12-08
13	4	1	2025-01-14 00:00:00	7	NULL
14	4	2	2025-01-14 00:00:00	7	NULL
15	3	1	2025-01-03 00:00:00	7	2025-01-17
16	3	1	2025-01-04 00:00:00	7	2025-01-17
17	2	1	2025-01-04 00:00:00	7	2025-01-17

Readers

id	name	surname	phone	email
1	Никита	Волков	89257257209	nikita.volkov01@mail.ru
2	Алексей	Крыжовников	89157776655	alex.kryzh@gmail.com
3	Тимур	Юнусов	89777777777	timati@bs.com
4	Валерий	Сюткин	89264449243	black.cat@cloud.com

Решение

Запрос для вопроса 1:

```
SELECT title AS 'Название книги', count(*) AS 'Количество взятий'
FROM Given_books INNER JOIN Books_count ON Given_books.book_id = Books_count.book_id
                 INNER JOIN Books ON Books_count.book_id = Books.id
WHERE MONTH(date_of_start) = MONTH(SUBDATE(NOW(), INTERVAL 1 MONTH))
GROUP BY title
ORDER BY 2 DESC
LIMIT 5
```

Результат

Название книги	Количество взятий
Дезертир рая	3
Амур с гранатой	2
Преступление и наказание	2
Финансист	2
Евгений Онегин	2

Запрос для вопроса 2:

```
SELECT Readers.name as Name, Readers.surname as Surname, count(*) as 'Количество просроченных'
FROM
(
	SELECT DISTINCT book_id, reader_id
	FROM
	(
		SELECT book_id, reader_id, deadline,
		DATEDIFF(IF(ISNULL(return_date), CURDATE(), return_date), date_of_start) as start_finish_difference
		FROM Given_books
	) AS start_finish_difference_Table
	WHERE start_finish_difference > deadline
) AS FoundedBooksAndReadersTable
		INNER JOIN Readers ON FoundedBooksAndReadersTable.reader_id = Readers.id
		INNER JOIN Books_count ON FoundedBooksAndReadersTable.book_id =
		Books_count.book_id
		INNER JOIN Books ON Books_count.book_id = Books.id
		INNER JOIN Genres ON Books.genre_id = Genres.id
WHERE Genres.name = 'Детектив'
GROUP BY Name, Surname
```

Результат

Name	Surname	Количество просроченных
Никита	Волков	3
Валерий	Сюткин	1