

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЁТ о выполнении проекта по предмету “Базы данных”**

**Обучающегося: Столярова Антона Викторовича  
группы № 22201 курса 3**

**Тема проекта: "Информационная система библиотеки ВУЗа"**

*Новосибирск 2025*

## Текст задания

Библиотека включает в себя абонементы, читальные залы и справочную систему каталогов и картотек. Читателями библиотеки вуза имеют право быть: студенты всех форм обучения, профессорско-преподавательский состав университета, аспиранты, ассистенты и другие сотрудники подразделений вуза, слушатели подготовительного отделения (ПО), факультета повышения квалификации (ФПК), стажеры, абитуриенты. Различные категории читателей среди прочих обладают характеристиками, специфическими для своей категории: для студентов это название факультета, номер группы, для преподавателя - название кафедры, степень, звание и т.д. Слушатели ФПК, абитуриенты, стажеры - разовые читатели - имеют право пользоваться только читальными залами.

Читатели библиотеки имеют право получать книги и другие источники информации на всех пунктах выдачи библиотеки (абонементах и читальных залах), а также получать необходимые издания по межбиблиотечному абонементу, сделав предварительно заказ. Читатели, приходящие на пункт выдачи, обязаны иметь при себе читательский билет с отметками о записи и перерегистрации текущего года на данном пункте выдачи. При выбытии из вуза (отчисление, окончание обучения, увольнение) читатели обязаны вернуть числящиеся за ними издания и сдать читательские билеты.

За нарушение правил пользования библиотекой читатели лишаются права пользования всеми пунктами обслуживания библиотеки на установленные администрацией сроки (от 1 до 6 месяцев). В случае утери или порчи книг читатель обязан заменить их такими же или другими изданиями, признанными библиотекой равноценными, или же возместить их 10-кратную стоимость. В случае невозвращения в библиотеку книг в установленный срок, читатель обязан заплатить штраф.

Срок пользования литературой для различных категорий читателей и количество выдаваемых изданий на каждом абонементе определяется администрацией, исходя из вида литературы и категории читателя. Число книг, выдаваемых в читальных залах, не ограничивается.

При поступлении новых изданий в библиотеку они должны быть внесены в картотеку с указанием их количества для каждого абонемента и читального зала. Выдача книг, сроки, штрафы и т.п. собираются и обрабатываются администрацией.

## Инфологическая модель

### Сущности

№	Название сущности	Описание
1	Publication_issuing	Информация о выдаче публикаций (кому, когда, где, срок возврата и т.п.)
2	Pickup_point	Информация о пунктах выдачи (в зависимости от типа - количество мест, либо тип абонемента и т.п.)

3	Library_card	Информация о читательских билетах (кому, когда, где выдан и т.п.)
4	Pickup_point_publication	Связь между пунктом выдачи и публикацией (с указанием лимита публикаций на пункт)
5	Order	Информация о заказах публикаций (кому, что, где, когда и т.п.)
6	Reader	Информация о читателе (тип, имя, поля специфичные для каждого типа и т.п.)
7	Publication	Информация о публикации (название, автор, издание и т.п.)
8	Violation_instance	Информация о нарушениях (кто, когда, что нарушил, наказание и т.п.)
9	Teacher	Информация о преподавателях кафедры, звание, научная степень и т.д.
10	Publication_event	Информация о событиях, происходящих с публикацией (с какой публикацией что, когда произошло и т.п., за исключением того, когда )
11	Violation	Информация о типе нарушения (название и другие присущие признаки (если нужно))
12	Event_type	Информация о типе события с публикацией (название и другие присущие признаки (если нужно))
13	Library_card_reregistration	Логи перерегистрации читательского билета (когда, где, какой билет перерегистрировали)
14	Restriction	Информация об ограничениях для читателя (например: разовые читатели могут пользоваться только абонементом)
15	Applicant	Информация о типе читателя абитуриент (например: информация о факультете, на который собирается абитуриент)
16	University_employee	Информация о сотрудниках университета, которые не принадлежат к другим категориям (не нуждаются в атрибутах, присутствующих в других категориях читателей)
17	Student	Информация о студентах (группа, факультет и т.п.)

18	Science-worker	Информация о научных сотрудниках (принадлежность к кафедре, научная степень и т.п.)
19	Library_card_status	Информация о статусе читательского билета (активен, не активен и т.д.)
20	Publication_status	Информация о статусе публикации (Утеряна, в наличии и т.д.)

## Сущности и атрибуты

\*жирным выделены primary keys

Сущность	Атрибуты	Описание
Publication_issuing	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Issuing_date 6. Expiration_date 7. Return_date	2. ID читательского билета, которому выдавалась публикация 3. ID выдаваемой публикации 4. ID пункта выдачи, на котором была выдана публикация 5. Дата выдачи публикации 6. Дата, когда публикация должна быть возвращена 7. Дата возвращения публикации читателем
Pickup_point	1. <b>ID</b> 2. Subscription_type 3. Number_of_seats 4. Status 5. Name	2. Тип абонеента (межвузовый или обычный и т.п.), null, если пунктом выдачи является читальный зал 3. количество мест в читальном зале (null, если пунктом выдачи является абонемент) 4. Статус пункта выдачи (работает, ремонт и т.п.) 5. Название пункта выдачи (например "Читальный зал №3")
Library_card	1. <b>ID</b> 2. Issue date 3. Reader_ID 4. Expiration_date 5. Status_ID 6. Restriction_ID	2. Дата выдачи читательского билета 3. ID того, кому читательский билет был выдан 4. Дата, когда читательский билет заканчивается 5. Ссылка на статус билета (например: активен, приостановлен и т.п.) 6. Ограничения предъявленные читателю (например: разрешается использовать только читальные залы)
Pickup_point_publication	1. <b>Pickup_point_ID</b> 2. <b>Publication_ID</b> 3. Publication_limit	1. ID пункта выдачи 2. ID публикации 3. Предел количества выдаваемых публикаций на данный пункт выдачи

Order	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Ordering_date 6. Order_type_ID	2. ID читательского билета, который осуществлял заказ 3. ID публикации, которую читатель заказал 4. ID пункта выдачи, где заказ был осуществлен 5. Дата заказа 6. ID типа заказа (межбиблиотечный абонемент был использован или нет (edge case, когда один и тот же читатель заказывает одну и ту же публикацию по разным абонементам))
Reader	1. <b>ID</b> 2. Category 3. Surname 4. Status 5. Name 6. Middle_name	2. Категория читателя (например студент, преподаватель, абитуриент и т.п.) 3. Фамилия читателя 4. Текущий статус читателя (активный, ограничен в доступе и т.п.) 5. Имя читателя 6. Отчество читателя (если есть)
Publication	1. <b>ID</b> 2. Name 3. Author 4. Publishing_date 5. Publishing_office 6. Price 7. Status_ID	2. Название публикации 3. Автор(-ы) публикации 4. Дата выпуска публикации 5. Издание опубликовавшее публикацию (nullable) 6. Цена публикации 7. Текущий статус публикации (возвращена, не возвращена и т.п.)
Violation_instance	1. <b>Reader_ID</b> 2. <b>Violation_category_ID</b>	1. ID читателя, совершившего нарушение 2. ID типа нарушения, которое было совершено
Teacher	1. <b>ID</b> 2. Degree 3. Rank 4. Joining_date 5. Departure_date	2. Степень преподавателя 3. Звание преподавателя 4. Дата найма в университет 5. Дата увольнения из университета
Publication_event	1. <b>ID</b> 2. Publication_ID 3. Event_type 4. Event_date 5. Pickup_point_ID	2. ID публикации, с которой произошло событие 3. Тип события, случившегося с публикацией 4. Дата происшествия события 5. ID пункта выдачи, где событие произошло
Violation	1. <b>ID</b> 2. Name 3. Violation_date 4. Restriction_id	2. Описание нарушения 3. Дата, когда произошло нарушение 4. Ограничение (наказание) за нарушение
Event_type	1. <b>ID</b> 2. Name	2. Название категории читателя

Library_card _reregistrati on	1. ID 2. Reregistration_date 3. Library_card 4. Pickup_point	2. Дата перерегистрации 3. Читательский билет, который перерегистрировали 4. Пункт выдачи, на котором произошла перерегистрация
Restriction	1. ID 2. Description	2. Описание ограничения, наложенного на читателя
Applicant	1. ID 2. Faculty	2. Факультет, на который планирует подаваться абитуриент
University_e mployee	1. ID 2. Joining_date 3. Departure_date	2. Дата найма сотрудника университета 3. Дата увольнения сотрудника университета
Student	1. ID 2. Group 3. Faculty 4. Joining_date 5. Departure_date	2. Группа, в которой студент учится 3. Факультет, на котором студент учится 4. Дата зачисления в университет 5. Дата отчисления из университета
Science-wor ker	1. ID 2. Faculty 3. Department 4. Degree 5. Rank 6. Joining_date 7. Departure_date	2. Факультет научного сотрудника 3. Кафедра научного сотрудника 4. Степень научного сотрудника 5. Звание научного сотрудника 6. Дата зачисления/найма в университет 7. Дата отчисления/увольнения из университета
Library_card _status	1. ID 2. Info	2. Информация о статусе
Publication_ status	1. ID 2. Info	2. Информация о статусе

## Сущности и связи

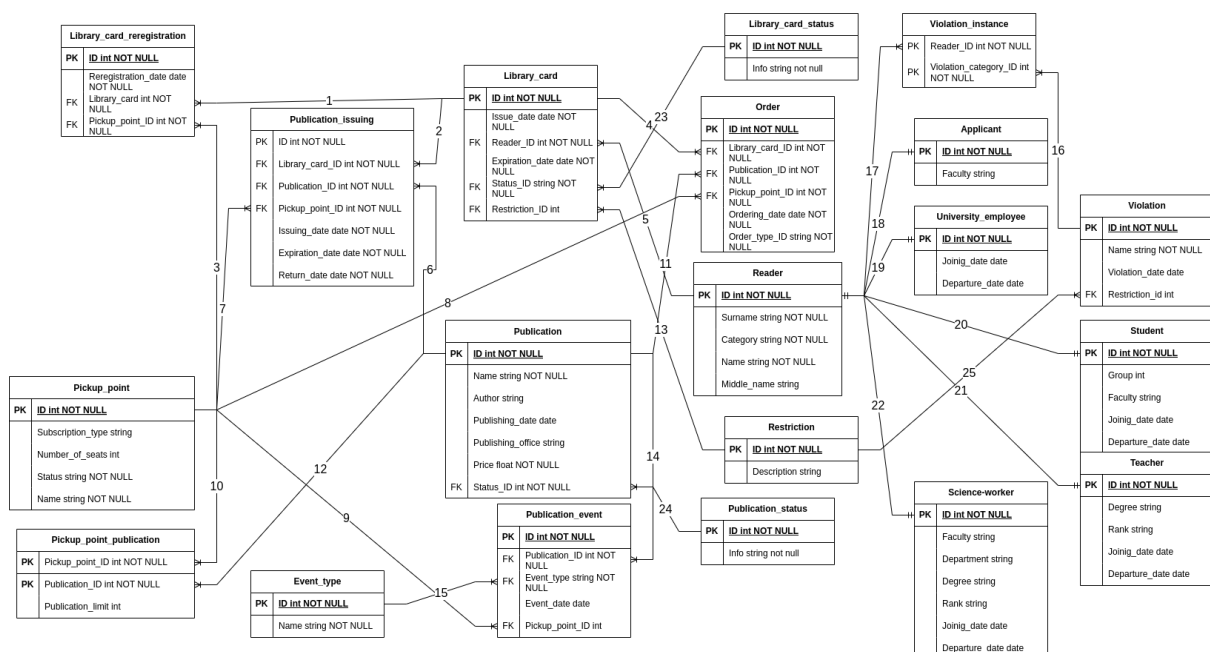
\*условная справа (слева) значит условная по отношению к таблице указанной справа (слева) во второй колонке

№	Связанные сущности	Тип связи	Свойства связи	Описание
2	Publication_issuing; Library_card	N:1	on update cascade; on delete set null	Указание на билет использованный при выдаче публикации
6	Publication_issuing; Publication	N:1	on update cascade; on delete cascade	Указание на публикацию, которую выдали
7	Publication_issuing; Pickup_point	N:1	on update cascade; on delete set null	Указание на пункт, на котором произошла

				выдача
1	Library_card_reregistration; Library_card	N:1	on update cascade; on delete cascade	Указание на читательский билет, для которого произошла выдача
8	Pickup_point; Order	1:N	on update cascade; on delete set null	Указание на пункт выдачи на котором произошёл заказ
9	Pickup_point; Publication_event	1:N, условная справа	on update cascade; on delete set null	Указание на пункт выдачи на котором произошло событие с публикацией
10	Pickup_point; Pickup_point_publication	1:N	on update cascade; on delete cascade	Связь пункта выдачи с таблицей ограничений на публикации
4	Library_card; Order	1:N	on update cascade; on delete set null	Указание на читательский билет, который использовался при заказе
5	Library_card; Reader	N:1	on update cascade; on delete set null	Указание на читателя, указанного в читательском билете
12	Pickup_point_publication; Publication	N:1	on update cascade; on delete cascade	Связь публикации с таблицей ограничений на публикации
11	Order; Publication	N:1	on update cascade; on delete cascade	Указание на заказываемую публикацию
14	Publication; Publication_event	1:N	on update cascade; on delete cascade	Указание на публикацию, с которой произошло событие
15	Publication_event; Event_type	N:1	on update cascade; on delete set null	Указание на тип событие которое произошло с публикацией
17	Reader; Violation_instance	1:N	on update cascade; on delete cascade	Указание на читателя, совершившего нарушение
3	Library_card_reregistration; Pickup_point	N:1	on update cascade; on delete set null	Указание на пункт выдачи, на котором произошла перерегистрация читательского билета
16	Violation_instance; Violation	N:1	on update cascade; on delete cascade	Указание на тип нарушения в таблице нарушений

13	<b>Library_card;</b> <b>Restriction</b>	N:1, условная слева	on update cascade; on delete set null	Указание на ограничение для читательского билета
18	<b>Reader;</b> <b>Applicant</b>	1:1	on update cascade; on delete cascade	Определение типа читателя - абитуриента
19	<b>Reader;</b> <b>University_employee</b>	1:1	on update cascade; on delete cascade	Определение типа читателя - не научного сотрудника университета
20	<b>Reader;</b> <b>Student</b>	1:1	on update cascade; on delete cascade	Определение типа читателя - студента
21	<b>Reader;</b> <b>Teacher</b>	1:1	on update cascade; on delete cascade	Определение типа читателя - преподавателя
22	<b>Reader;</b> <b>Science-worker</b>	1:1	on update cascade; on delete cascade	Определение типа читателя - научного сотрудника
23	<b>Library_card_status;</b> <b>Library_card</b>	1:N (условная справа)	on update cascade; on delete set null	Определение статуса читательского билета
24	<b>Publication_status;</b> <b>Publication</b>	1:N (условная справа)	on update cascade; on delete set null	Определение статуса публикации
25	<b>Violation;</b> <b>Restriction</b>	N:1 (условная слева)	on update cascade; on delete set null	Определение ограничения (наказание) за нарушение

## ER-модель





## Пользователи

**Администратор** - пользователь с полным доступом ко всем таблицам, нужен для разрешения конфликтных ситуаций в базе данных, помощи другим пользователям в исполнении того, к чему у них доступа нет.

**Библиотекарь** - пользователь имеющий возможность добавлять, изменять данные таблиц публикаций, читательских билетов, заказов и т.д. (см. таблицу прав). Нужен для сотрудников библиотеки, работающих на пунктах выдачи и занимающихся приемом посетителей библиотеки, организацией заказов и выдачей публикаций.

**Сотрудник отдела комплектования и учёта** - пользователь, имеющий доступ на значительный доступ к таблицам пунктов выдачи, публикаций, событий публикаций и т.д. (см. таблицу прав). Подразумевается, что будет использоваться сотрудниками библиотеки, занимающимися структурированием, доставкой и организацией публикаций в библиотеке и на пунктах выдачи (контакта с посетителями не имеют).

## Таблица прав пользователей

Сущность\Пользователь	Администратор	Библиотекарь	Сотрудник отдела комплектования и учета
Publication_issuing	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	Чтение
Pickup_point	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	Изменение, чтение
Library_card	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	Чтение
Pickup_point_publication	Добавление, изменение, чтение, удаление	Чтение	Добавление, изменение, чтение

<b>Order</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	Чтение
<b>Reader</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Publication</b>	Добавление, изменение, чтение, удаление	Чтение	Добавление, изменение, чтение
<b>Violation_instance</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Teacher</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Publication_event</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	Добавление, изменение, чтение
<b>Violation</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Event_type</b>	Добавление, изменение, чтение, удаление	Чтение	Чтение
<b>Library_card_registration</b>	Добавление, изменение, чтение, удаление	Добавление, чтение	-
<b>Restriction</b>	Добавление, изменение, чтение, удаление	Чтение	-
<b>Applicant</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>University_employee</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-

<b>Student</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Science-worker</b>	Добавление, изменение, чтение, удаление	Добавление, изменение, чтение	-
<b>Library_card_status</b>	Добавление, изменение, чтение, удаление	Чтение	-
<b>Publication_status</b>	Добавление, изменение, чтение, удаление	Чтение	Чтение

## Логическая модель

Таблицы данной базы данных были созданы со следующими соображениями:

- избыточность сведена к минимуму
- введены ограничения на все отношения
- выведены первичные ключи и они минимальны
- выведены внешние ключи (без зависимостей между атрибутами, где ни один из них не главный ключ)
- созданы триггеры, необходимые для обеспечения целостности базы данных

Таким образом выполняется четвертая нормальная форма:

1. Первая нормальная форма
  - все атрибуты атомарны
  - каждое отношение имеет первичный ключ
2. Вторая нормальная форма
  - находится в первой нормальной форме
  - нет атрибутов, зависящих от части составного ключа
3. Третья нормальная форма
  - находится во второй нормальной форме
  - все атрибуты зависят только от первичного ключа (нет транзитивной зависимости в рамках одной таблицы)
4. Четвёртая нормальная форма
  - находится в третьей нормальной форме
  - отсутствуют многозначные зависимости: каждая потенциальная многозначная зависимость выделена в отдельное отношение
  - каждая таблица описывает только одну фактическую многозначную связь или одну сущность, поэтому для любого ключа не существует двух независимых множественных атрибутов в рамках одной таблицы

## Реляционная схема

\***жирным** выделены primary keys; внешние ключи перечислены индексами атрибутов этой же таблицы

Сущность	Атрибуты	Тип	Ограничения	Внешний ключ
Publication_issuing	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Issuing_date 6. Expiration_date 7. Return_date	1. bigint 2. bigint 3. bigint 4. shortint 5. date 6. date 7. date	1. not null 3. not null 5. not null 6. not null	2, 3, 4
Pickup_point	1. <b>ID</b> 2. Subscription_type 3. Number_of_seats 4. Status 5. Name	1. shortint 2. char 3. shortint 4. char 5. varchar	1. not null 3. >0 4. not null	-
Library_card	1. <b>ID</b> 2. Issue date 3. Reader_ID 4. Expiration_date 5. Status_ID 6. Restriction_ID	1. bigint 2. date 3. bigint 4. date 5. shortint 6. shortint	1. not null 2. not null 3. not null	3, 5, 6
Pickup_point_publication	1. <b>Pickup_point_ID</b> 2. <b>Publication_ID</b> 3. Publication_limit	1. shortint 2. bigint 3. shortint	1. not null 2. not null 3. >0	1, 2
Order	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Ordering_date 6. Order_type_ID	1. bigint 2. bigint 3. bigint 4. shortint 5. date 6. char	1. not null 3. not null 5. not null 6. not null	2, 3, 4
Reader	1. <b>ID</b> 2. Category 3. Surname 4. Name 5. Middle_name	1. bigint 2. char 3. varchar 4. varchar 5. varchar	1. not null 2. not null 3. not null 4. not null	1
Publication	1. <b>ID</b> 2. Name 3. Author 4. Publishing_date 5. Publishing_office 6. Price 7. Status_ID	1. bigint 2. varchar 3. varchar 4. date 5. varchar 6. numeric 7. shortint	1. not null 2. not null 6. not null, >0	7

Violation_instance	1. <b>Reader_ID</b> 2. <b>Violation_category_ID</b>	1. bigint 2. shortint	1. not null 2. not null	1, 2
Teacher	1. <b>ID</b> 2. Degree 3. Rank 4. Joining_date 5. Departure_date	1. int 2. char 3. char 4. date 5. date	1. not null 4. not null	1
Publication_event	1. <b>ID</b> 2. Publication_ID 3. Event_type 4. Event_date 5. Pickup_point_ID	1. bigint 2. bigint 3. shortint 4. date 5. shortint	1. not null 2. not null 3. not null	2, 3, 5
Violation	1. <b>ID</b> 2. Name 3. Violation_date 4. Restriction_id	1. smalltint 2. char 3. date 4. smallint	1. not null 2. not null	4
Event_type	1. <b>ID</b> 2. Name	1. shortint 2. char	1. not null 2. not null, unique	-
Library_card_reregistration	1. <b>ID</b> 2. Reregistration_date 3. Library_card 4. Pickup_point	1. bigint 2. date 3. bigint 4. shortint	1. not null 2. not null 3. not null 4. not null	3, 4
Restriction	1. <b>ID</b> 2. Description	1. shortint 2. char	1. not null 2. not null, unique	-
Applicant	1. <b>ID</b> 2. Faculty	1. int 2. char	1. not null 2. not null	1
University_employee	1. <b>ID</b> 2. Joining_date 3. Departure_date	1. int 2. date 3. date	1. not null 2. not null	1
Student	1. <b>ID</b> 2. Group 3. Faculty 4. Joining_date 5. Departure_date	1. bigint 2. shortint 3. char 4. date 5. date	1. not null 2. not null, >0 3. not null 4. not null	1
Science-worker	1. <b>ID</b> 2. Faculty 3. Department 4. Degree 5. Rank 6. Joining_date 7. Departure_date	1. int 2. char 3. char 4. char 5. char 6. date 7. date	1. not null 4. not null 6. not null	1
Library_card	1. <b>ID</b>	1. shortint	1. not null	-

_status	2. Info	2. varchar	2. not null, unique	
Publication_ status	1. <b>ID</b> 2. Info	1. shortint 2. varchar	1. not null 2. not null, unique	-

### Таблица значений по умолчанию

Сущность	Атрибуты	Значения по умолчанию
Publication_issuing	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Issuing_date 6. Expiration_date 7. Return_date	-
Pickup_point	1. <b>ID</b> 2. Subscription_type 3. Number_of_seats 4. Status 5. Name	4. "Inactive"
Library_card	1. <b>ID</b> 2. Issue date 3. Reader_ID 4. Expiration_date 5. Status_ID	-
Pickup_point_publication	1. <b>Pickup_point_ID</b> 2. <b>Publication_ID</b> 3. Publication_limit	-
Order	1. <b>ID</b> 2. Library_card_ID 3. Publication_ID 4. Pickup_point_ID 5. Ordering_date 6. Order_type_ID	-
Reader	1. <b>ID</b> 2. Category 3. Surname 4. Name 5. Middle_name	-
Publication	1. <b>ID</b> 2. Name	-

	3. Author 4. Publishing_date 5. Publishing_office 6. Price 7. Status_ID	
Violation_instance	1. <b>Reader_ID</b> 2. <b>Violation_category_ID</b>	-
Teacher	1. <b>ID</b> 2. Degree 3. Rank 4. Joining_date 5. Departure_date	-
Publication_event	1. <b>ID</b> 2. Publication_ID 3. Event_type 4. Event_date 5. Pickup_point_ID	-
Violation	1. <b>ID</b> 2. Name 3. Violation_date 4. Penalty 5. Penalty_duration_until	-
Event_type	1. <b>ID</b> 2. Name	-
Library_card_reregistration	1. <b>ID</b> 2. Reregistration_date 3. Library_card 4. Pickup_point	-
Restriction	1. <b>ID</b> 2. Description	-
Applicant	1. <b>ID</b> 2. Faculty	-
University_employee	1. <b>ID</b> 2. Joining_date 3. Departure_date	-
Student	1. <b>ID</b> 2. Group 3. Faculty 4. Joining_date 5. Departure_date	-
Science-worker	1. <b>ID</b> 2. Faculty 3. Department 4. Degree	-

	5. Rank 6. Joining_date 7. Departure_date	
Library_card_status	1. ID 2. Info	-
Publication_status	1. ID 2. Info	-

## Триггеры

№	Название	Условия включения	Таблица	Момент срабатывания	Действие, выполняемое при срабатывании триггера
1	Publication returned update	On update (when return date has been changed)	Publication_issuing	After update	Изменения статуса публикации на "Present" в таблице Publication, когда публикация была возвращена
2	Publication order update	On insert	Order	After insert	Изменения статуса публикации на "Ordered" в таблице Publication, когда публикация была заказана
3	Status update on Library card when departing	On update	University_employee	After update	Изменение статуса читательского билета в "Inactive" из-за отбытия из университета
4	Status update on Library_card when departing	On update	Student	After update	Изменение статуса читательского билета в "Inactive" из-за отбытия из университета
5	Status update on Library_card when departing	On update	Teacher	After update	Изменение статуса читателя в "Inactive" из-за отбытия из университета
6	Status update on Library_card when departing	On update	Science-worker	After update	Изменение статуса читателя в "Inactive" из-за отбытия из университета

## Процедуры



№	Название	Описание
1	issue_library_card	Добавляет новый читательский билет
2	register_library_card	Заносит лог о перерегистрации читательского билета
3	record_reader_violation	Записывает лог о нарушении читателя
4	add_new_publication	Добавляет новую публикацию
5	add_new_student	Добавляет нового студента
6	borrow_publication	Записывает лог о выдаче публикации
7	return_publication	Запись лога о возвращении публикации

## Скрипты создания базы

### База данных и роли

```
create role admin
with
    login password 'admin_password';

create database university_library owner admin;

create user librarian
with
    password 'librarian_password';

create user catalog_manager
with
    password 'catalog_manager_password';
```

### Таблицы и представления

```
create table IF NOT EXISTS
    pickup_point
(
    id          smallserial primary key,
    subscription_type char(40),
    number_of_seats  smallint check (number_of_seats > 0),
    status_        char(200) not null DEFAULT 'Inactive',
    name_         varchar(350)
);

create table IF NOT EXISTS
    event_type
(
    id  smallserial primary key,
    name_ char(300) not null unique
);

create table IF NOT EXISTS
    restriction
```

```

(
    id          smallserial primary key,
    description_ char(500) not null unique
);

create table IF NOT EXISTS
    library_card_status
(
    id          smallserial primary key,
    info        varchar(100) not null unique
);

create table IF NOT EXISTS
    violation
(
    id          smallserial primary key,
    name_       char(350) not null,
    violation_date date,
    restriction_id smallint references restriction (id) on update cascade on delete set null
);

create table IF NOT EXISTS
    publication_status
(
    id          smallserial primary key,
    info        varchar(100) not null unique
);

create table IF NOT EXISTS
    publication
(
    id          bigserial primary key,
    name_       varchar(500) not null,
    author      varchar(400),
    publishing_date date,
    publishing_office varchar(400),
    price       money not null check (price > 0.00::money),
    status_id   smallint references publication_status (id) on update cascade on delete set null
);

create table IF NOT EXISTS publication_event
(
    id          bigserial primary key,
    publication_id bigint not null references publication (id) on update cascade on delete cascade,
    event_type_id smallint not null references event_type (id) on update cascade on delete set null,
    event_date   date,
    pickup_point_id smallint references pickup_point (id) on update cascade on delete set null
);

create table IF NOT EXISTS pickup_point_publication
(
    pickup_point_id smallint not null references pickup_point (id) on update cascade on delete cascade,
    publication_id   bigint not null references publication (id) on update cascade on delete cascade,
    publication_limit smallint check (publication_limit > 0),
    primary key (pickup_point_id, publication_id)
);

create table IF NOT EXISTS
    reader
(
    id          bigserial primary key,
    category    char(50) not null,
    surname     varchar(100) not null,
    name_       varchar(100) not null,
    middle_name varchar(100) not null
);

create table IF NOT EXISTS
    applicant
(
    id          serial primary key references reader (id) on update cascade on delete cascade,
    faculty     char(250) not null
);

create table IF NOT EXISTS

```

```

university_employee
(
    id          serial primary key references reader (id) on update cascade on delete cascade,
    joining_date date not null,
    departure_date date
);

create table IF NOT EXISTS
    student
(
    id          bigserial primary key references reader (id) on update cascade on delete cascade,
    group_      char(20) not null,
    faculty     char(250) not null,
    course      smallint, -- Added course column here
    joining_date date not null,
    departure_date date
);

create table IF NOT EXISTS
    teacher
(
    id          serial primary key references reader (id) on update cascade on delete cascade,
    degree      char(200),
    rank        char(200),
    joining_date date not null,
    departure_date date
);

create table IF NOT EXISTS
    science_worker
(
    id          serial primary key references reader (id) on update cascade on delete cascade,
    faculty     char(250),
    department   char(250),
    degree_     char(200) not null,
    rank        char(200),
    joining_date date not null,
    departure_date date
);

create table IF NOT EXISTS violation_instance
(
    reader_id    bigint references reader (id) on update cascade on delete cascade,
    violation_category_id smallint references violation (id) on update cascade on delete cascade,
    primary key (reader_id, violation_category_id)
);

create table IF NOT EXISTS
    library_card
(
    id          bigserial primary key,
    issue_date  date not null,
    reader_id   bigint not null references reader (id) on update cascade on delete set null,
    expiration_date date,
    status_id   smallint references library_card_status (id) on update cascade on delete set null,
    restriction_id smallint references restriction (id) on update cascade on delete set null
);

create table IF NOT EXISTS
    publication_issuing
(
    id          bigserial primary key,
    library_card_id bigint references library_card (id) on update cascade on delete set null,
    publication_id bigint not null references publication (id) on update cascade on delete cascade,
    pickup_point_id smallint references pickup_point (id) on update cascade on delete set null,
    issuing_date date not null,
    expiration_date date not null,
    return_date date
);

create table IF NOT EXISTS order_
(
    id          bigserial primary key,
    library_card_id bigint references library_card (id) on update cascade on delete set null,
    publication_id bigint not null references publication (id) on update cascade on delete cascade,

```

```

pickup_point_id smallint references pickup_point (id) on update cascade on delete set null,
ordering_date date not null,
order_type char(300) not null
);

create table IF NOT EXISTS library_card_reregistration
(
    id bigserial primary key,
    reregistration_date date not null,
    library_card_id bigint not null references library_card (id) on update cascade on delete cascade,
    pickup_point_id smallint not null references pickup_point (id) on update cascade on delete set null
);

DO
$$
BEGIN
    IF NOT EXISTS (SELECT 1
        FROM information_schema.columns
        WHERE table_name = 'student'
        AND column_name = 'course') THEN
        ALTER TABLE student
        ADD COLUMN course smallint;
    END IF;
END
$$;

DROP VIEW IF EXISTS debtors_view;
DROP VIEW IF EXISTS book_orders_stats;
DROP VIEW IF EXISTS reader_full_info;

CREATE OR REPLACE VIEW reader_full_info AS
SELECT r.id,
    r.category,
    r.surname,
    r.name_,
    r.middle_name,
    CASE
        WHEN s.id IS NOT NULL THEN 'student'
        WHEN t.id IS NOT NULL THEN 'teacher'
        WHEN sw.id IS NOT NULL THEN 'science_worker'
        WHEN ue.id IS NOT NULL THEN 'university_employee'
        WHEN a.id IS NOT NULL THEN 'applicant'
        ELSE 'other'
    END as reader_type,
    COALESCE(s.faculty, sw.faculty, a.faculty, 'N/A') as faculty,
    COALESCE(s.group_, 'N/A') as group_,
    COALESCE(s.course, 0) as course,
    COALESCE(sw.department, 'N/A') as department,
    COALESCE(t.degree, sw.degree_, 'N/A') as degree,
    COALESCE(t.rank, sw.rank, 'N/A') as rank_
FROM reader r
LEFT JOIN student s ON r.id = s.id
LEFT JOIN teacher t ON r.id = t.id
LEFT JOIN science_worker sw ON r.id = sw.id
LEFT JOIN university_employee ue ON r.id = ue.id
LEFT JOIN applicant a ON r.id = a.id;

CREATE OR REPLACE VIEW debtors_view AS
SELECT rfi.*,
    pi.id as issuing_id,
    pi.issuing_date,
    pi.expiration_date,
    pi.return_date,
    pi.pickup_point_id,
    pp.name_ as pickup_point_name,
    p.name_ as publication_name,
    p.author,
    (CASE
        WHEN pi.return_date IS NULL AND pi.expiration_date < CURRENT_DATE
        THEN (CURRENT_DATE - pi.expiration_date)
        ELSE NULL END) as days_overdue
FROM reader_full_info rfi
JOIN library_card lc ON rfi.id = lc.reader_id
JOIN publication_issuing pi ON lc.id = pi.library_card_id
JOIN pickup_point pp ON pi.pickup_point_id = pp.id

```

```

JOIN publication p ON pi.publication_id = p.id
WHERE pi.return_date IS NULL
AND pi.expiration_date < CURRENT_DATE;

CREATE OR REPLACE VIEW book_orders_stats AS
SELECT p.id      as publication_id,
       p.name_   as publication_name,
       p.author,
       pp.id     as pickup_point_id,
       pp.name_  as pickup_point_name,
       rfi.faculty,
       COUNT(o.id) as order_count
FROM publication p
      JOIN order_ o ON p.id = o.publication_id
      JOIN pickup_point pp ON o.pickup_point_id = pp.id
      JOIN library_card lc ON o.library_card_id = lc.id
      JOIN reader_full_info rfi ON lc.reader_id = rfi.id
GROUP BY p.id, p.name_, p.author, pp.id, pp.name_, rfi.faculty;

```

## Разрешения пользователей

```

grant select, update, insert on publication_issuing to librarian;
grant select, update, insert on pickup_point to librarian;
grant select, update, insert on library_card to librarian;
grant select on pickup_point_publication to librarian;
grant select, update, insert on order_ to librarian;
grant select, update, insert on reader to librarian;
grant select on publication to librarian;
grant select, update, insert on violation_instance to librarian;
grant select, update, insert on teacher to librarian;
grant select, update, insert on publication_event to librarian;
grant select, update, insert on violation to librarian;
grant select on event_type to librarian;
grant select, insert on library_card_reregistration to librarian;
grant select on restriction to librarian;
grant select, update, insert on applicant to librarian;
grant select, update, insert on university_employee to librarian;
grant select, update, insert on student to librarian;
grant select, update, insert on science_worker to librarian;
grant select on library_card_status to librarian;
grant select on publication_status to librarian;

grant select on publication_issuing to catalog_manager;
grant select, update on pickup_point to catalog_manager;
grant select on library_card to catalog_manager;
grant select, update, insert on pickup_point_publication to catalog_manager;
grant select on order_ to catalog_manager;
grant select, update, insert on publication to catalog_manager;
grant select, update, insert on publication_event to catalog_manager;
grant select on event_type to catalog_manager;
grant select on publication_status to catalog_manager;

GRANT SELECT ON reader_full_info TO librarian;
GRANT SELECT ON reader_full_info TO catalog_manager;

GRANT SELECT ON debtors_view TO librarian;
GRANT SELECT ON debtors_view TO catalog_manager;

GRANT SELECT ON book_orders_stats TO librarian;
GRANT SELECT ON book_orders_stats TO catalog_manager;

```

## Триггеры

```

create or replace function publication_returned_update()
returns trigger as $$

```

```

begin
if (old.return_date != new.return_date) and new.return_date is not null then
    update publication
    set status_id = (
        select id from publication_status where info = 'present'
    )
    where id = new.publication_id;
end if;
return new;
end;
$$ language plpgsql;

create trigger publication_returned
after update on publication_issuing
for each row
execute function publication_returned_update();

create or replace function publication_order_update()
returns trigger as $$
begin
    update publication
    set status_id = (
        select id from publication_status where info = 'ordered'
    )
    where id = new.publication_id;
    return new;
end;
$$ language plpgsql;

create trigger publication_order
after insert on order_
for each row
execute function publication_order_update();

create or replace function library_card_inactivate()
returns trigger as $$
declare inactive_id int;
begin
if old.departure_date is null and new.departure_date is not null then
    select id into inactive_id
    from library_card_status
    where info = 'inactive';
    update library_card
    set status_id = inactive_id
    where reader_id = new.id;
end if;
return new;
end;
$$ language plpgsql;

create trigger emp_departure
after update on university_employee
for each row
execute function library_card_inactivate();

create trigger student_departure
after update on student
for each row
execute function library_card_inactivate();

create trigger teacher_departure
after update on teacher
for each row
execute function library_card_inactivate();

create trigger science_worker_departure
after update on science_worker
for each row
execute function library_card_inactivate();

```

## Процедуры

```

create or replace procedure issue_library_card(

```

```

    p_reader_id bigint,
    p_issue_date date,
    p_expiration_date date,
    p_status_id smallint,
    p_restriction_id smallint
)
language plpgsql
as $$
begin
    insert into library_card (reader_id, issue_date, expiration_date, status_id, restriction_id)
    values (p_reader_id, p_issue_date, p_expiration_date, p_status_id, p_restriction_id);
end;
$$;

create or replace procedure register_library_card(
    p_library_card_id bigint,
    p_reregistration_date date,
    p_pickup_point_id smallint
)
language plpgsql
as $$
begin
    insert into library_card_reregistration (library_card_id, reregistration_date, pickup_point_id)
    values (p_library_card_id, p_reregistration_date, p_pickup_point_id);
end;
$$;

create or replace procedure record_reader_violation(
    p_reader_id bigint,
    p_violation_category_id smallint
)
language plpgsql
as $$
begin
    insert into violation_instance (reader_id, violation_category_id)
    values (p_reader_id, p_violation_category_id);
end;
$$;

create or replace procedure add_new_publication(
    p_name varchar,
    p_author varchar,
    p_publishing_date date,
    p_publishing_office varchar,
    p_price money,
    p_status_id smallint
)
language plpgsql
as $$
begin
    insert into publication (name_, author, publishing_date, publishing_office, price, status_id)
    values (p_name, p_author, p_publishing_date, p_publishing_office, p_price, p_status_id);
end;
$$;

create or replace procedure add_new_student(
    p_surname varchar,
    p_name varchar,
    p_middle_name varchar,
    p_group char,
    p_faculty char,
    p_joining_date date,
    p_departure_date date
)
language plpgsql
as $$
declare
    new_reader_id bigint;
begin
    insert into reader (category, surname, name_, middle_name)
    values ('student', p_surname, p_name, p_middle_name)
    returning id into new_reader_id;

    insert into student (id, group_, faculty, joining_date, departure_date)
    values (new_reader_id, p_group, p_faculty, p_joining_date, p_departure_date);
end;

```

```

end;
$$;

create or replace procedure borrow_publication(
    p_library_card_id bigint,
    p_publication_id bigint,
    p_pickup_point_id smallint,
    p_issuing_date date,
    p_expiration_date date
)
language plpgsql
as $$
begin
    insert into publication_issuing (library_card_id, publication_id, pickup_point_id, issuing_date, expiration_date)
    values (p_library_card_id, p_publication_id, p_pickup_point_id, p_issuing_date, p_expiration_date);
end;
$$;

create or replace procedure return_publication(
    p_publication_issuing_id bigint,
    p_return_date date
)
language plpgsql
as $$
begin
    update publication_issuing
    set return_date = p_return_issuing_id
    where id = p_publication_issuing_id;
end;
$$;

```

## Вставка данных

```

INSERT INTO pickup_point (id, subscription_type, number_of_seats, status_, name_)
VALUES (1, 'Абонемент', NULL, 'Active', 'Главный абонемент'),
(2, 'Читальный зал', 100, 'Active', 'Читальный зал №1 (Гуманитарные науки)'),
(3, 'Абонемент', NULL, 'Maintenance', 'Абонемент для преподавателей'),
(4, 'Читальный зал', 75, 'Active', 'Читальный зал №2 (Технические науки)'),
(5, 'Межбиблиотечный абонемент', NULL, 'Active', 'Отдел МБА'),
(6, 'Абонемент', NULL, 'Inactive', 'Абонемент редких книг'),
(7, 'Читальный зал', 60, 'Active', 'Читальный зал периодики')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO event_type (id, name_)
VALUES (1, 'Поступление нового издания'),
(2, 'Списание издания'),
(3, 'Утеря издания читателем'),
(4, 'Замена утерянного издания'),
(5, 'Передача в другой пункт выдачи'),
(6, 'Возврат из другого пункта выдачи'),
(7, 'Инвентаризация - найдено'),
(8, 'Инвентаризация - не найдено')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO restriction (id, description_)
VALUES (1, 'Лишение права пользования на 1 месяц'),
(2, 'Лишение права пользования на 2 месяца'),
(3, 'Лишение права пользования на 3 месяца'),
(4, 'Лишение права пользования на 6 месяцев'),
(5, 'Только читальный зал (разовый читатель)'),
(6, 'Обязательство возместить утерянную книгу'),
(7, 'Запрет на выдачу до погашения штрафа'),
(8, 'Ограничение на количество одновременно взятых книг')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO library_card_status (id, info)
VALUES (1, 'Активен'),
(2, 'Просрочен'),
(3, 'Заблокирован (нарушение)'),
(4, 'Сдан (выбытие)'),
(5, 'Утерян'),
(6, 'Ожидает перерегистрации'),
(7, 'Аннулирован'),

```



```
(8, 'Первичная регистрация')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO publication_status (id, info)
VALUES (1, 'В наличии'),
(2, 'Выдано'),
(3, 'Заказано'),
(4, 'Списано'),
(5, 'Утеряно'),
(6, 'В ремонте'),
(7, 'На бронеполке'),
(8, 'Передано в другой отдел')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO reader (id, category, surname, name_, middle_name)
VALUES (1, 'Абитуриент', 'Иванов', 'Петр', 'Сергеевич'),
(2, 'Абитуриент', 'Петрова', 'Анна', 'Викторовна'),
(3, 'Абитуриент', 'Сидоров', 'Алексей', 'Игоревич'),
(4, 'Абитуриент', 'Кузнецова', 'Елена', 'Дмитриевна'),
(5, 'Абитуриент', 'Михайлов', 'Иван', 'Александрович'),
(6, 'Абитуриент', 'Васильева', 'Ольга', 'Николаевна'),
(7, 'Абитуриент', 'Новиков', 'Дмитрий', 'Федорович'),
(8, 'Абитуриент', 'Морозова', 'Светлана', 'Юрьевна')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO reader (id, category, surname, name_, middle_name)
VALUES (9, 'Сотрудник ВУЗа', 'Смирнов', 'Андрей', 'Владимирович'),
(10, 'Сотрудник ВУЗа', 'Попова', 'Мария', 'Алексеевна'),
(11, 'Сотрудник ВУЗа', 'Волков', 'Сергей', 'Михайлович'),
(12, 'Сотрудник ВУЗа', 'Федорова', 'Екатерина', 'Ивановна'),
(13, 'Сотрудник ВУЗа', 'Романов', 'Олег', 'Петрович'),
(14, 'Сотрудник ВУЗа', 'Захарова', 'Татьяна', 'Борисовна'),
(15, 'Сотрудник ВУЗа', 'Павлов', 'Денис', 'Григорьевич'),
(16, 'Сотрудник ВУЗа', 'Козлова', 'Ирина', 'Леонидовна')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO reader (id, category, surname, name_, middle_name)
VALUES (17, 'Студент', 'Алексеев', 'Максим', 'Андреевич'),
(18, 'Студент', 'Соколова', 'Виктория', 'Сергеевна'),
(19, 'Студент', 'Лебедев', 'Артем', 'Олегович'),
(20, 'Студент', 'Орлова', 'Дарья', 'Константиновна'),
(21, 'Студент', 'Егоров', 'Никита', 'Евгеньевич'),
(22, 'Студент', 'Степанова', 'Анастасия', 'Романовна'),
(23, 'Студент', 'Ковалев', 'Владислав', 'Денисович'),
(24, 'Студент', 'Ильина', 'Полина', 'Максимовна')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO reader (id, category, surname, name_, middle_name)
VALUES (25, 'Преподаватель', 'Борисов', 'Владимир', 'Анатольевич'),
(26, 'Преподаватель', 'Антонова', 'Людмила', 'Геннадьевна'),
(27, 'Преподаватель', 'Григорьев', 'Игорь', 'Степанович'),
(28, 'Преподаватель', 'Макарова', 'Нина', 'Васильевна'),
(29, 'Преподаватель', 'Тихонов', 'Юрий', 'Эдуардович'),
(30, 'Преподаватель', 'Белова', 'Александра', 'Павловна'),
(31, 'Преподаватель', 'Семенов', 'Константин', 'Яковлевич'),
(32, 'Преподаватель', 'Виноградова', 'Зоя', 'Феликсовна')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO reader (id, category, surname, name_, middle_name)
VALUES (33, 'Научный сотрудник', 'Медведев', 'Аркадий', 'Ильич'),
(34, 'Научный сотрудник', 'Ершова', 'Валентина', 'Захаровна'),
(35, 'Научный сотрудник', 'Фомин', 'Геннадий', 'Тимофеевич'),
(36, 'Научный сотрудник', 'Лазарева', 'Клавдия', 'Наумовна'),
(37, 'Научный сотрудник', 'Давыдов', 'Платон', 'Эльдарович'),
(38, 'Научный сотрудник', 'Архипова', 'Регина', 'Станиславовна'),
(39, 'Научный сотрудник', 'Жуков', 'Руслан', 'Харитонович'),
(40, 'Научный сотрудник', 'Соловьева', 'Эльвира', 'Чеславовна')
ON CONFLICT (id) DO NOTHING;
```

```
INSERT INTO applicant (id, faculty)
VALUES (1, 'Факультет информационных технологий'),
(2, 'Экономический факультет'),
(3, 'Юридический факультет'),
(4, 'Факультет журналистики'),
(5, 'Строительный факультет');
```

```

(6, 'Медицинский факультет'),
(7, 'Факультет иностранных языков'),
(8, 'Физико-математический факультет')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO university_employee (id, joining_date, departure_date)
VALUES (9, '2010-09-01', NULL),
(10, '2015-03-10', NULL),
(11, '2005-08-15', '2023-07-31'),
(12, '2018-11-20', NULL),
(13, '2012-01-10', NULL),
(14, '2019-06-01', '2024-05-10'),
(15, '2008-07-07', NULL),
(16, '2021-02-15', NULL)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO student (id, group_, faculty, course, joining_date, departure_date)
VALUES (17, 'ИСТ-201', 'Факультет информационных технологий', 3, '2020-09-01', NULL),
(18, 'ЭК-302', 'Экономический факультет', 2, '2019-09-01', NULL),
(19, 'ЮР-101', 'Юридический факультет', 1, '2021-09-01', '2023-06-30'),
(20, 'ЖУР-405', 'Факультет журналистики', 4, '2018-09-01', NULL),
(21, 'СТР-210', 'Строительный факультет', 3, '2020-09-01', NULL),
(22, 'МЕД-503', 'Медицинский факультет', 5, '2017-09-01', '2023-07-15'),
(23, 'ИНЯЗ-315', 'Факультет иностранных языков', 2, '2019-09-01', NULL),
(24, 'ФМ-111', 'Физико-математический факультет', 1, '2021-09-01', NULL)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO teacher (id, degree, rank, joining_date, departure_date)
VALUES (25, 'Кандидат технических наук', 'Доцент', '2002-09-01', NULL),
(26, 'Доктор экономических наук', 'Профессор', '1995-03-10', NULL),
(27, 'Кандидат юридических наук', 'Старший преподаватель', '2010-08-15', '2024-01-10'),
(28, 'Кандидат филологических наук', 'Доцент', '2008-11-20', NULL),
(29, NULL, 'Ассистент', '2019-09-01', NULL),
(30, 'Доктор медицинских наук', 'Профессор', '2000-06-01', NULL),
(31, 'Кандидат педагогических наук', 'Доцент', '2015-07-07', NULL),
(32, 'Доктор физико-математических наук', 'Заведующий кафедрой', '1998-02-15', '2023-12-31')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO science_worker (id, faculty, department, degree_, rank, joining_date, departure_date)
VALUES (33, 'Физико-математический факультет', 'Кафедра теоретической физики', 'Доктор физико-математических наук',
'Ведущий научный сотрудник', '2005-09-01', NULL),
(34, 'Химический факультет', 'Лаборатория органического синтеза', 'Кандидат химических наук',
'Старший научный сотрудник', '2012-03-15', NULL),
(35, 'Биологический факультет', 'Кафедра генетики', 'Кандидат биологических наук', 'Научный сотрудник',
'2018-11-01', '2024-04-30'),
(36, 'Исторический факультет', 'Отдел археологии', 'Доктор исторических наук', 'Главный научный сотрудник',
'2000-01-20', NULL),
(37, 'Факультет информационных технологий', 'Лаборатория искусственного интеллекта', 'Кандидат технических наук',
'Младший научный сотрудник', '2021-09-01', NULL),
(38, 'Географический факультет', 'Кафедра картографии', 'Кандидат географических наук',
'Старший научный сотрудник', '2010-07-10', NULL),
(39, 'Экономический факультет', 'Центр экономических исследований', 'Доктор экономических наук',
'Ведущий научный сотрудник', '2008-05-05', NULL),
(40, 'Филологический факультет', 'Отдел славяноведения', 'Кандидат филологических наук', 'Научный сотрудник',
'2016-02-20', '2023-10-10')
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO violation (id, name_, violation_date, restriction_id)
VALUES (1, 'Утеря книги "Основы программирования"', '2023-05-15', 6),
(2, 'Невозврат книги "Высшая математика" в срок', '2023-11-10', 1),
(3, 'Порча книги "История Древнего Мира"', '2024-01-20', 6),
(4, 'Повторное нарушение сроков возврата', '2024-02-01', 2),
(5, 'Курение в читальном зале', '2023-09-05', 1),
(6, 'Передача читательского билета другому лицу', '2024-03-12', 3),
(7, 'Попытка выноса книги из читального зала без записи', '2023-12-22', 1),
(8, 'Невозврат 3+ книг в установленный срок', '2024-04-30', 4)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO publication (id, name_, author, publishing_date, publishing_office, price, status_id)
VALUES (1, 'Алгоритмы: построение и анализ', 'Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн',
'2019-01-15', 'Вильямс', 2500.00::money, 1),
(2, 'Структура и интерпретация компьютерных программ', 'Харольд Абельсон, Джеральд Джей Сассман, Джули Сассман',
'2017-05-20', 'Добросвет', 1800.50::money, 2),
(3, 'Искусственный интеллект: современный подход', 'Стьюарт Рассел, Питер Норвиг', '2021-08-10', 'Питер',
3200.00::money, 1),

```

```

(4, 'Физика для любознательных. Том 1: Механика', 'Л.Д. Ландау, А.И. Китайгородский', '2015-03-01', 'Наука',
1200.75::money, 4),
(5, 'Краткая история времени', 'Стивен Хокинг', '2018-11-25', 'АСТ', 950.00::money, 1),
(6, 'Война и мир. Том 1', 'Лев Толстой', '2010-06-01', 'Эксмо', 750.25::money, 3),
(7, 'Социология: Основы общей теории', 'Никлас Луман', '2009-09-09', 'РГГУ', 1500.00::money, 5),
(8, 'История государства и права зарубежных стран. Часть 1', 'Крашенинникова Н.А., Жидков О.А.', '2020-02-10',
'Норма', 2100.90::money, 1)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO publication_event (id, publication_id, event_type_id, event_date, pickup_point_id)
VALUES (1, 1, 1, '2023-01-10', 1), -- Поступление
(2, 2, 2, '2023-02-15', 2), -- Списание
(3, 3, 1, '2023-03-20', 1), -- Поступление
(4, 4, 3, '2023-04-05', 4), -- Утеря (привязана к пункту выдачи)
(5, 5, 1, '2023-05-12', 2), -- Поступление
(6, 6, 5, '2023-06-18', 1), -- Передача
(7, 6, 6, '2023-06-19', 3), -- Возврат
(8, 7, 1, '2023-07-25', 7), -- Поступление
(9, 1, 3, '2024-01-01', NULL), -- Утеря (не привязана к пункту выдачи)
(10, 3, 1, '2024-02-01', 1), -- Поступление (новый год)
(11, 5, 3, '2024-03-01', NULL), -- Утеря (новый год)
(12, 8, 1, '2024-04-01', 2) -- Поступление (новый год)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO pickup_point_publication (pickup_point_id, publication_id, publication_limit)
VALUES (1, 1, 10), -- Алгоритмы в Главном абонементе
(2, 1, 5), -- Алгоритмы в Читальном зале №1
(1, 3, 8), -- Искусственный интеллект в Главном абонементе
(4, 3, 4), -- Искусственный интеллект в Читальном зале №2
(1, 5, 12), -- Краткая история времени в Главном абонементе
(7, 5, 6), -- Краткая история времени в Читальном зале периодики
(1, 8, 7), -- История государства в Главном абонементе
(2, 8, 3), -- История государства в Читальном зале №1
(3, 1, 3), -- Алгоритмы в Абонементе для преподавателей
(1, 2, 5), -- Структура и интерпретация в Главном абонементе
(5, 6, 2) -- Война и мир в Отделе МБА
ON CONFLICT (pickup_point_id, publication_id) DO NOTHING;

```

```

INSERT INTO violation_instance (reader_id, violation_category_id)
VALUES (17, 2), -- Студент Алексеев, нарушение 2 (невозврат)
(25, 1), -- Преподаватель Борисов, нарушение 1 (утеря)
(19, 3), -- Студент Лебедев, нарушение 3 (порча)
(9, 4), -- Сотрудник Смирнов, нарушение 4 (повторное)
(33, 5), -- Научный сотрудник Медведев, нарушение 5 (курение)
(2, 6), -- Абитуриент Петрова, нарушение 6 (передача билета)
(21, 7), -- Студент Егоров, нарушение 7 (попытка выноса)
(28, 8), -- Преподаватель Макарова, нарушение 8 (невозврат 3+ книг)
(17, 1), -- Алексеев, еще одно нарушение (для total_violations_count)
(18, 2) -- Соколова, нарушение
ON CONFLICT (reader_id, violation_category_id) DO NOTHING;

```

```

INSERT INTO library_card (id, issue_date, reader_id, expiration_date, status_id, restriction_id)
VALUES (1, '2022-09-01', 17, '2026-08-31', 1, NULL), -- Активен, Главный абонемент
(2, '2021-09-01', 18, '2025-08-31', 1, NULL), -- Активен, Читальный зал №1
(3, '2020-09-01', 25, '2025-08-31', 1, NULL), -- Активен, Главный абонемент
(4, '2019-03-10', 10, '2024-03-09', 2, NULL), -- Просрочен, Отдел МБА
(5, '2023-01-15', 1, '2023-07-15', 5, 5), -- Утерян, Только читальный зал
(6, '2022-11-01', 33, '2027-10-31', 1, NULL), -- Активен, Читальный зал периодики
(7, '2021-08-20', 19, '2023-06-30', 4, 3), -- Сдан (выбытие), Заблокирован на 3 месяца
(8, '2023-05-01', 28, '2028-04-30', 3, 4), -- Заблокирован (нарушение), Лишен на 6 месяцев
(9, '2024-04-01', 20, '2028-03-31', 1, NULL), -- Новый читатель, Читальный зал №2
(10, '2024-05-15', 21, '2028-05-14', 1, NULL), -- Новый читатель, Главный абонемент
(11, '2022-01-01', 11, '2024-01-01', 4, NULL), -- Выбывший читатель (ушел в 2024)
(12, '2023-01-01', 12, '2024-05-01', 4, NULL), -- Выбывший читатель (ушел в 2024)
(13, '2024-01-01', 34, '2025-01-01', 1, NULL), -- Научный сотрудник
(14, '2024-01-01', 35, '2025-01-01', 1, NULL) -- Научный сотрудник
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO publication_issuing (id, library_card_id, publication_id, pickup_point_id, issuing_date, expiration_date,
return_date)
VALUES (1, 1, 1, 1, '2024-03-01', '2024-09-01', NULL), -- На руках у Алексеева (студент 17)

```

```

(2, 2, 3, 2, '2024-02-15', '2024-08-15', NULL),      -- На руках у Соколовой (студент 18)
(3, 3, 5, 1, '2024-01-10', '2024-07-10', '2024-05-01'), -- Возвращена Борисовым (преподаватель 25)
(4, 6, 8, 7, '2024-04-05', '2024-10-05', NULL),      -- На руках у Медведева (науч. сотр. 33)
(5, 1, 2, 1, '2023-10-01', '2024-04-01', NULL),      -- Просрочена у Алексеева (студент 17) - 2 месяца просрочки
(6, 4, 6, 3, '2023-11-01', '2024-05-01', '2024-02-28'), -- Возвращена Поповой (сотрудник 10)
(7, 2, 1, 2, '2023-09-15', '2024-03-15', '2024-01-20'), -- Возвращена Соколовой (студент 18)
(8, 3, 8, 1, '2024-05-02', '2024-11-02', NULL),      -- На руках у Борисова (преподаватель 25)
(9, 1, 4, 1, '2024-05-10', '2024-08-10', NULL),      -- На руках у Алексеева (студент 17)
(10, 2, 5, 2, '2024-05-12', '2024-08-12', NULL),     -- На руках у Соколовой (студент 18)
(11, 1, 3, 1, '2024-05-15', '2024-08-15', NULL),     -- На руках у Алексеева (студент 17) - для задачи 12
(12, 8, 3, 1, '2024-05-16', '2024-08-16',
NULL)      -- На руках у Борисова (преподаватель 25) - для задачи 12, сдать позже
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO order_ (id, library_card_id, publication_id, pickup_point_id, ordering_date, order_type)
VALUES (1, 1, 6, 1, '2024-05-01', 'Обычный заказ на абонемент'),      -- Алексеев, Война и мир
      (2, 3, 2, 5, '2024-04-20', 'Межбиблиотечный абонемент'),      -- Борисов, Структура и интерпретация (МБА)
      (3, 6, 7, 5, '2024-03-10', 'Заказ на утерянную книгу (для ознакомления)'), -- Медведев, Социология (МБА)
      (4, 2, 4, 2, '2024-05-05', 'Бронирование для читального зала'), -- Соколова, Физика
      (5, 8, 1, 1, '2024-02-11', 'Обычный заказ на абонемент'),      -- Макарова, Алгоритмы
      (6, 1, 3, 1, '2024-01-25', 'Обычный заказ на абонемент'),      -- Алексеев, Искусственный интеллект
      (7, 3, 5, 5, '2023-12-15', 'Межбиблиотечный абонемент'),      -- Борисов, Краткая история времени (МБА)
      (8, 6, 1, 7, '2024-04-28', 'Бронирование для читального зала'), -- Медведев, Алгоритмы
      (9, 1, 8, 1, '2024-05-20', 'Обычный заказ на абонемент'),      -- Алексеев, История государства
      (10, 2, 6, 2, '2024-05-22', 'Обычный заказ на абонемент'),     -- Соколова, Война и мир
      (11, 3, 1, 1, '2024-05-25', 'Обычный заказ на абонемент'),     -- Борисов, Алгоритмы (для топ-20)
      (12, 1, 1, 1, '2024-05-26', 'Обычный заказ на абонемент'),     -- Алексеев, Алгоритмы (для топ-20)
      (13, 2, 1, 2, '2024-05-27', 'Обычный заказ на абонемент'),     -- Соколова, Алгоритмы (для топ-20)
      (14, 4, 1, 5, '2024-05-28', 'Межбиблиотечный абонемент'),      -- Попова, Алгоритмы (МБА)
      (15, 6, 1, 7, '2024-05-29', 'Бронирование для читального зала') -- Медведев, Алгоритмы (для топ-20)
ON CONFLICT (id) DO NOTHING;

```

```

INSERT INTO library_card_reregistration (id, reregistration_date, library_card_id, pickup_point_id)
VALUES (1, '2023-09-05', 1, 1),
      (2, '2023-09-10', 2, 2),
      (3, '2023-09-02', 3, 1),
      (4, '2022-09-01', 6, 7),
      (5, '2024-01-15', 1, 1),
      (6, '2024-01-20', 3, 3),
      (7, '2023-08-25', 2, 2),
      (8, '2023-09-11', 6, 7),
      (9, '2024-05-01', 9, 4), -- Перерегистрация нового читателя
      (10, '2024-05-10', 10, 1) -- Перерегистрация нового читателя
ON CONFLICT (id) DO NOTHING;

```

## Запросы

```

-- 1) query
SELECT COALESCE(rfi.department, 'N/A data')      AS department,
       COALESCE(rfi.faculty, 'N/A data')          AS faculty,
       COALESCE(rfi.course::text, 'N/A data')     AS course, -- rfi.course is numeric, cast for display
       COALESCE(rfi.group, 'N/A data')            AS group_identifier,
       COUNT(DISTINCT rfi.id)                    AS total_readers,
       STRING_AGG(DISTINCT rfi.surname || ' ' || rfi.name_ || COALESCE(' ' || rfi.middle_name, ''), ' ') AS readers_list
FROM reader_full_info rfi
WHERE (
  %(pickup_point_id) s IS NULL OR
  EXISTS ( -- check if reader has any issued books from this pickup point
    SELECT 1
    FROM publication_issuing pi_filter
    JOIN library_card_ic_filter ON pi_filter.library_card_id = ic_filter.id
    WHERE ic_filter.reader_id = rfi.id
    AND pi_filter.pickup_point_id = %(pickup_point_id) s) OR
  EXISTS ( -- check if reader has any orders from this pickup point
    SELECT 1
    FROM order_o_filter
    JOIN library_card_ic_filter ON o_filter.library_card_id = ic_filter.id
    WHERE ic_filter.reader_id = rfi.id
    AND o_filter.pickup_point_id = %(pickup_point_id) s)
)
AND ( %(department_name) s IS NULL OR rfi.department ILIKE %(department_name) s)
AND ( %(faculty_name) s IS NULL OR rfi.faculty ILIKE %(faculty_name) s)
AND ( %(course_name) s IS NULL OR rfi.course::text ILIKE %(course_name) s) -- Compare course as text
AND ( %(group_name) s IS NULL OR rfi.group_ ILIKE %(group_name) s)
GROUP BY COALESCE(rfi.department, 'N/A data'),
         COALESCE(rfi.faculty, 'N/A data'),
         COALESCE(rfi.course::text, 'N/A data'),
         COALESCE(rfi.group_, 'N/A data')
ORDER BY department, faculty, course, group_identifier;

-- 2) query
SELECT COALESCE(dv.department, 'N/A data')      AS department,
       COALESCE(dv.faculty, 'N/A data')          AS faculty,
       COALESCE(dv.course::text, 'N/A data')     AS course,
       COALESCE(dv.group_, 'N/A data')            AS group_identifier,
       COALESCE(dv.category, 'N/A data')          AS reader_category,
       COUNT(DISTINCT dv.id)                    AS total_overdue_readers,
       STRING_AGG(DISTINCT dv.surname || ' ' || dv.name_ || COALESCE(' ' || dv.middle_name, '')) || ' (Просрочено: ' || dv.days_overdue || ' дней, Квирра: ' || dv.publication_name || ')', ' ') AS overdue_readers_list
FROM debtors_view dv
WHERE ( %(pickup_point_id) s IS NULL OR dv.pickup_point_id = %(pickup_point_id) s)

```

```

AND (%(min_days_overdue) s IS NULL OR dv.days_overdue > %(min_days_overdue) s)
AND (%(department_name) s IS NULL OR dv.department ILIKE %(department_name) s)
AND (%(faculty_name) s IS NULL OR dv.faculty ILIKE %(faculty_name) s)
AND (%(course_name) s IS NULL OR dv.course:text ILIKE %(course_name) s)
AND (%(group_name) s IS NULL OR dv.group ILIKE %(group_name) s)
AND (%(reader_category_name) s IS NULL OR dv.category ILIKE %(reader_category_name) s)
GROUP BY COALESCE(dv.department, 'N/A data'),
          COALESCE(dv.faculty, 'N/A data'),
          COALESCE(dv.course:text, 'N/A data'),
          COALESCE(dv.group_, 'N/A data'),
          COALESCE(dv.category, 'N/A data')
ORDER BY total_overdue_readers DESC, department, faculty, course, group_identifier, reader_category;

-- 3) query
SELECT bos.publication_name,
       bos.author,
       bos.pickup_point_name AS ordering_location,
       bos.faculty          AS reader_faculty,
       bos.order_count
FROM book_orders_state bos
WHERE (%(pickup_point_id) s IS NULL OR bos.pickup_point_id = %(pickup_point_id) s)
AND (%(faculty_name) s IS NULL OR bos.faculty ILIKE %(faculty_name) s)
ORDER BY bos.order_count DESC LIMIT 20;

-- 4) query (received)
WITH ReceivedEvents AS (SELECT pe.publication_id,
                                p.name_      AS publication_name,
                                p.author,
                                p.publishing_date,
                                pe.event_date AS receipt_date,
                                pe.pickup_point_id,
                                pp.name_     AS pickup_point_name
                        FROM publication_event pe
                        JOIN publication p ON pe.publication_id = p.id
                        JOIN event_type et ON pe.event_type_id = et.id
                        LEFT JOIN pickup_point pp ON pe.pickup_point_id = pp.id
                        WHERE et.name_ ILIKE %(event_type_reception) s
                        AND pe.event_date >= (CURRENT_DATE - INTERVAL '1 year')
                        AND pe.event_date <= CURRENT_DATE
                        )
SELECT COALESCE(re.pickup_point_name, 'N/A или центральный фонд') AS location_received,
       re.author,
       EXTRACT(YEAR FROM re.publishing_date)::TEXT AS publication_year, EXTRACT(YEAR FROM re.receipt_date)::TEXT AS receipt_year_in_library, COUNT(re.publication_id) AS total_books_received,
       STRING_AGG(DISTINCT re.publication_name || ' (Получено: ' || TO_CHAR(re.receipt_date, 'YYYY-MM-DD') || '); ', '') AS received_books_list
FROM ReceivedEvents re
WHERE (%(pickup_point_id) s IS NULL OR re.pickup_point_id = %(pickup_point_id) s)
AND (%(author_name) s IS NULL OR re.author ILIKE %(author_name) s)
AND (%(publication_year_filter) s IS NULL OR
      EXTRACT (YEAR FROM re.publishing_date) = %(publication_year_filter) s)
AND (%(receipt_year_filter) s IS NULL OR EXTRACT (YEAR FROM re.receipt_date) = %(receipt_year_filter) s)
GROUP BY COALESCE(re.pickup_point_name, 'N/A или центральный фонд'),
          re.author,
          EXTRACT(YEAR FROM re.publishing_date)::TEXT,
          EXTRACT(YEAR FROM re.receipt_date)::TEXT
ORDER BY location_received, author, publication_year, receipt_year_in_library;

-- 4) query (lost)
WITH LostEvents AS (SELECT pe.publication_id,
                           p.name_      AS publication_name,
                           p.author,
                           p.publishing_date,
                           pe.event_date AS lost_date,
                           pe.pickup_point_id AS last_known_pickup_point_id,
                           pp.name_     AS last_known_pickup_point_name
                   FROM publication_event pe
                   JOIN publication p ON pe.publication_id = p.id
                   JOIN event_type et ON pe.event_type_id = et.id
                   LEFT JOIN pickup_point pp ON pe.pickup_point_id = pp.id
                   WHERE et.name_ ILIKE %(event_type_lost) s -- Parameterized
                   AND pe.event_date >= (CURRENT_DATE - INTERVAL '1 year')
                   AND pe.event_date <= CURRENT_DATE
                   )
SELECT COALESCE(le.last_known_pickup_point_name, 'N/A или центральный фонд') AS location_lost_from,
       le.author,
       EXTRACT(YEAR FROM le.publishing_date)::TEXT AS publication_year, EXTRACT(YEAR FROM le.lost_date)::TEXT AS year_lost, COUNT(le.publication_id) AS total_books_lost,
       STRING_AGG(DISTINCT le.publication_name || ' (Утеряно: ' || TO_CHAR(le.lost_date, 'YYYY-MM-DD') || '); ', '') AS lost_books_list
FROM LostEvents le
WHERE (%(pickup_point_id) s IS NULL OR le.last_known_pickup_point_id = %(pickup_point_id) s)
AND (%(author_name) s IS NULL OR le.author ILIKE %(author_name) s)
AND (%(publication_year_filter) s IS NULL OR
      EXTRACT (YEAR FROM le.publishing_date) = %(publication_year_filter) s)
AND (%(lost_year_filter) s IS NULL OR EXTRACT (YEAR FROM le.lost_date) = %(lost_year_filter) s)
GROUP BY COALESCE(le.last_known_pickup_point_name, 'N/A или центральный фонд'),
          le.author,
          EXTRACT(YEAR FROM le.publishing_date)::TEXT,
          EXTRACT(YEAR FROM le.lost_date)::TEXT
ORDER BY location_lost_from, author, publication_year, year_lost;

-- 5) query (regular readers)
SELECT pp.name_          AS pickup_point_name,
       COUNT(DISTINCT r_assoc.reader_id) AS total_readers
FROM pickup_point pp
LEFT JOIN (
-- Readers who ordered from this pickup point
SELECT o.pickup_point_id AS pid, lc.reader_id
FROM order_o
JOIN library_card lc ON o.library_card_id = lc.id
WHERE o.pickup_point_id IS NOT NULL
UNION
-- Readers who had books issued from this pickup point
SELECT pi.pickup_point_id AS pid, lc.reader_id
FROM publication_issuing pi
JOIN library_card lc ON pi.library_card_id = lc.id
WHERE pi.pickup_point_id IS NOT NULL) r_assoc ON pp.id = r_assoc.pid
GROUP BY pp.id, pp.name_
ORDER BY total_readers %(sort_direction)s;

-- 5) query (debtors)
SELECT dv.pickup_point_name,
       COUNT(DISTINCT dv.id) AS total_overdue_readers
FROM debtors_view dv
WHERE dv.pickup_point_name IS NOT NULL
GROUP BY dv.pickup_point_name
ORDER BY total_overdue_readers %(sort_direction)s;

-- 5) query (biggest overdue sum)
SELECT pp.name_          AS pickup_point_name,
       SUM(COALESCE(p.price, 0::money)) AS total_overdue_value -- Summing price of overdue books as 'fine_amount' is not in schema
FROM pickup_point pp
JOIN publication_issuing pi ON pp.id = pi.pickup_point_id
JOIN debtors_view dv ON pi.id = dv.issuing_id -- Ensures only currently overdue items
JOIN publication p ON pi.publication_id = p.id -- To get the price
GROUP BY pp.name_
ORDER BY total_overdue_value %(sort_direction)s;

-- 6) query
SELECT p.name_          AS publication_name,
       p.author,
       COUNT(o.id)          AS total_orders,
       STRING_AGG(DISTINCT rf.surname || ' ' || rf.name_ || ' (Заказано: ' || TO_CHAR(o.ordering_date, 'YYYY-MM-DD') || '); ', '') AS ordering_readers_list
FROM order_o
JOIN publication p ON o.publication_id = p.id

```

```

JOIN pickup_point pp ON o.pickup_point_id = pp.id
JOIN library_card lc ON o.library_card_id = lc.id
JOIN reader_full_info rfi ON lc.reader_id = rfi.id
WHERE pp.subscription_type ILIKE %(interlibrary_loan_identifier)s
AND o.ordering_date >= (CURRENT_DATE - %(time_interval)s:: INTERVAL)
AND o.ordering_date <= CURRENT_DATE
GROUP BY p.name_, p.author
ORDER BY total_orders DESC, p.name_;
```

-- 7) query

```

WITH CopiesData AS (SELECT p.id AS publication_id,
p.name_ AS publication_name,
p.author,
ppp.pickup_point_id,
ppp.publication_limit,
(SELECT COUNT(*)
FROM publication_issuing pi_count
WHERE pi_count.publication_id = p.id
AND pi_count.pickup_point_id = ppp.pickup_point_id
AND pi_count.return_date IS NULL) AS issued_count
FROM publication p
JOIN pickup_point_publication ppp ON p.id = ppp.publication_id
WHERE %(publication_name_filter)s IS NULL OR p.name_ ILIKE
%(publication_name_filter)s
AND %(author_filter)s IS NULL OR p.author ILIKE %(author_filter)s
AND %(publication_id_filter)s IS NULL OR p.id = %(publication_id_filter)s)
SELECT CASE
WHEN %(pickup_point_id_filter)s IS NOT NULL AND %(pickup_point_id_filter)s != -1 THEN COALESCE (pp.name_, 'Неизвестный пункт')
ELSE 'По всей библиотеке'
END
AS scope,
cd.publication_name,
cd.author,
SUM(GREATEST(0, cd.publication_limit - cd.issued_count)) AS number_of_available_copies
FROM CopiesData cd
LEFT JOIN pickup_point pp ON cd.pickup_point_id = pp.id
WHERE %(pickup_point_id_filter)s IS NULL OR %(pickup_point_id_filter)s = -1 OR
cd.pickup_point_id = %(pickup_point_id_filter)s
GROUP BY scope, cd.publication_name, cd.author,
(CASE
WHEN %(pickup_point_id_filter)s IS NOT NULL AND %(pickup_point_id_filter)s != -1 THEN pp.name_
ELSE NULL END)
ORDER BY scope, cd.publication_name;
```

-- 8) query

```

SELECT COALESCE(rfi.department, 'N/A data') AS department,
COALESCE(rfi.faculty, 'N/A data') AS faculty,
COALESCE(rfi.course::text, 'N/A data') AS course,
COALESCE(rfi.group_, 'N/A data') AS group_identifier,
COALESCE(rfi.category, 'N/A data') AS reader_category,
COUNT(DISTINCT rfi.id) AS total_restricted_readers,
STRING_AGG(DISTINCT rfi.surname || ' ' || rfi.name_ ||
' (Нарушение от: ' || TO_CHAR(v.violation_date, 'YYYY-MM-DD') ||
', Причина: ' || res.description || '); ' ) AS restricted_readers_list
FROM reader_full_info rfi
JOIN violation_instance vi ON rfi.id = vi.reader_id
JOIN violation v ON vi.violation_category_id = v.id
JOIN restriction res ON v.restriction_id = res.id
WHERE res.description ILIKE %(restriction_deprived_pattern)s -- Parameterized
AND v.violation_date <= (CURRENT_DATE - INTERVAL '2 months')
AND %(department_name)s IS NULL
OR rfi.department ILIKE %(department_name)s
AND %(faculty_name)s IS NULL
OR rfi.faculty ILIKE %(faculty_name)s
AND %(course_name)s IS NULL
OR rfi.course::text ILIKE %(course_name)s
AND %(group_name)s IS NULL
OR rfi.group_ ILIKE %(group_name)s
AND %(reader_category_name)s IS NULL
OR rfi.category ILIKE %(reader_category_name)s
GROUP BY
COALESCE (rfi.department, 'N/A data'),
COALESCE (rfi.faculty, 'N/A data'),
COALESCE (rfi.course::text, 'N/A data'),
COALESCE (rfi.group_, 'N/A data'),
COALESCE (rfi.category, 'N/A data')
ORDER BY total_restricted_readers DESC, department, faculty, course, group_identifier, reader_category;
```

-- 9) query (new readers)

```

SELECT COALESCE(rfi.department, 'N/A data') AS department,
COALESCE(rfi.faculty, 'N/A data') AS faculty,
COALESCE(rfi.course::text, 'N/A data') AS course,
COALESCE(rfi.group_, 'N/A data') AS group_identifier,
COALESCE(rfi.category, 'N/A data') AS reader_category,
COUNT(DISTINCT rfi.id) AS total_new_readers,
STRING_AGG(DISTINCT rfi.surname || ' ' || rfi.name_ || ' (Карта выдана: ' || TO_CHAR(lc.issue_date, 'YYYY-MM-DD') || '); ' ) AS new_readers_list
FROM reader_full_info rfi
JOIN library_card lc ON rfi.id = lc.reader_id
WHERE lc.issue_date =
(SELECT MIN(lc_inner.issue_date) FROM library_card lc_inner WHERE lc_inner.reader_id = rfi.id)
AND lc.issue_date >= (CURRENT_DATE - %(time_interval)s:: INTERVAL)
AND lc.issue_date <= CURRENT_DATE
AND %(department_name)s IS NULL OR rfi.department ILIKE %(department_name)s
AND %(faculty_name)s IS NULL OR rfi.faculty ILIKE %(faculty_name)s
AND %(course_name)s IS NULL OR rfi.course::text ILIKE %(course_name)s
AND %(group_name)s IS NULL OR rfi.group_ ILIKE %(group_name)s
AND %(reader_category_name)s IS NULL OR rfi.category ILIKE %(reader_category_name)s
GROUP BY COALESCE(rfi.department, 'N/A data'),
COALESCE(rfi.faculty, 'N/A data'),
COALESCE(rfi.course::text, 'N/A data'),
COALESCE(rfi.group_, 'N/A data'),
COALESCE(rfi.category, 'N/A data')
ORDER BY total_new_readers DESC, department, faculty, course, group_identifier, reader_category;
```

-- 9) query (departed)

```

WITH ReaderDepartureDates AS (SELECT r.id as reader_id_key, s.departure_date
FROM reader r
JOIN student s ON r.id = s.id
WHERE s.departure_date IS NOT NULL
UNION ALL
SELECT r.id as reader_id_key, t.departure_date
FROM teacher t
JOIN teacher t ON r.id = t.id
WHERE t.departure_date IS NOT NULL
UNION ALL
SELECT r.id as reader_id_key, ue.departure_date
FROM reader r
JOIN university_employee ue ON r.id = ue.id
WHERE ue.departure_date IS NOT NULL
UNION ALL
SELECT r.id as reader_id_key, sw.departure_date
FROM reader r
JOIN science_worker sw ON r.id = sw.id
WHERE sw.departure_date IS NOT NULL),
ActualDeparture AS (SELECT reader_id_key, MAX(departure_date) as departure_date
FROM ReaderDepartureDates
GROUP BY reader_id_key)
SELECT COALESCE(rfi.department, 'N/A data') AS department,
COALESCE(rfi.faculty, 'N/A data') AS faculty,
COALESCE(rfi.course::text, 'N/A data') AS course,
COALESCE(rfi.group_, 'N/A data') AS group_identifier,
COALESCE(rfi.category, 'N/A data') AS reader_category,
```

```

COUNT(DISTINCT rf.id) AS total_departed_readers,
STRING_AGG(DISTINCT rf.surname || ' ' || rf.name_ || ' (Был: ' || TO_CHAR(ad.departure_date, 'YYYY-MM-DD') || '); ' || rf.name_ || ' AS departed_readers_list
FROM reader_full_info rf
JOIN ActualDeparture ad ON rf.id = ad.reader_id_key
WHERE ad.departure_date >= (CURRENT_DATE - %(time_interval) s)::INTERVAL
AND ad.departure_date <= CURRENT_DATE
AND (%(department_name) s IS NULL OR rf.department ILIKE %(department_name) s)
AND (%(faculty_name) s IS NULL OR rf.faculty ILIKE %(faculty_name) s)
AND (%(course_name) s IS NULL OR rf.course ILIKE %(course_name) s)
AND (%(group_name) s IS NULL OR rf.group ILIKE %(group_name) s)
AND (%(reader_category_name) s IS NULL OR rf.category ILIKE %(reader_category_name) s)
GROUP BY COALESCE(rf.department, 'N/A data'),
COALESCE(rf.faculty, 'N/A data'),
COALESCE(rf.course, 'N/A data'),
COALESCE(rf.group, 'N/A data'),
COALESCE(rf.category, 'N/A data')
ORDER BY total_departed_readers DESC, department, faculty, course, group_identifier, reader_category;

-- 10) query (ordered)
WITH ReaderOrders AS (SELECT p.name_ AS publication_name,
p.author,
o.ordering_date,
o.order_type,
pp.name_ AS ordered_at_pickup_point
FROM order_o
JOIN publication p ON o.publication_id = p.id
JOIN library_card lc ON o.library_card_id = lc.id
LEFT JOIN pickup_point pp ON o.pickup_point_id = pp.id
WHERE lc.reader_id = %(reader_id) s
AND o.ordering_date >= (CURRENT_DATE - %(time_interval) s)::INTERVAL
AND o.ordering_date <= CURRENT_DATE)
SELECT COUNT(*) AS total_ordered_books,
COALESCE(STRING_AGG(publication_name || ' aar: ' || COALESCE(author, 'N/A') ||
' (Заказано: ' ||
TO_CHAR(ordering_date, 'YYYY-MM-DD') || -- Corrected from order_date
', Тин заказа: ' || COALESCE(order_type, 'N/A') || '); ' ||
'Her sawase) AS ordered_books_list
FROM ReaderOrders;

-- 10) query (in possession)
WITH ReaderPossessions AS (SELECT p.name_ AS publication_name,
p.author,
pi.issuing_date,
pi.expiration_date,
pp.name_ AS issued_at_pickup_point,
(CASE
WHEN pi.expiration_date < CURRENT_DATE
THEN (CURRENT_DATE - pi.expiration_date)
ELSE 0
END) as days_overdue
FROM publication_issuing pi
JOIN library_card lc ON pi.library_card_id = lc.id
JOIN publication p ON pi.publication_id = p.id
LEFT JOIN pickup_point pp ON pi.pickup_point_id = pp.id
WHERE lc.reader_id = %(reader_id) s
AND pi.return_date IS NULL)
SELECT COUNT(*) AS total_books_on_hand,
COALESCE(STRING_AGG(publication_name || ' aar: ' || COALESCE(author, 'N/A') ||
' (Срок: ' || TO_CHAR(expiration_date, 'YYYY-MM-DD') ||
CASE
WHEN days_overdue > 0 THEN ' Просрочено: ' || days_overdue || ' дней'
ELSE ' END | '); ' ||
'Her wwr na pyxar') AS books_on_hand_list
FROM ReaderPossessions;

-- 11) query
SELECT pp.name_ AS subscriber_point_name,
p.name_ AS publication_name,
p.author,
SUM(GREATEST(0, COALESCE(ppp.publication_limit, 0) -
(SELECT COUNT(*)
FROM publication_issuing pi_inner
WHERE pi_inner.publication_id = p.id
AND pi_inner.pickup_point_id = pp.id
AND pi_inner.return_date IS NULL))) AS available_copies
FROM publication p
JOIN pickup_point_publication ppp ON p.id = ppp.publication_id
JOIN pickup_point pp ON ppp.pickup_point_id = pp.id
WHERE (%(publication_name_filter) s IS NULL OR p.name_ ILIKE %(publication_name_filter) s)
AND (%(author_filter) s IS NULL OR p.author ILIKE %(author_filter) s)
AND (%(publication_id_filter) s IS NULL OR p.id = %(publication_id_filter) s)
AND pp.subscription_type ILIKE %(subscription_type_pattern)s -- Parameterized
GROUP BY pp.id, pp.name_, p.id, p.name_, p.author
HAVING SUM (GREATEST(0
, COALESCE (ppp.publication_limit
0) -
(SELECT COUNT (*)
FROM publication_issuing pi_inner
WHERE pi_inner.publication_id = p.id
AND pi_inner.pickup_point_id = pp.id
AND pi_inner.return_date IS NULL)))
> 0
ORDER BY subscriber_point_name, publication_name;

-- 12) query
WITH BookHolders AS (SELECT rf.id AS reader_id,
rf.surname,
rf.name_,
rf.middle_name,
p.name_ AS publication_name,
p.author AS publication_author,
pi.expiration_date,
pp.name_ AS issued_from_pickup_point
FROM publication_issuing pi
JOIN library_card lc ON pi.library_card_id = lc.id
JOIN reader_full_info rf ON lc.reader_id = rf.id
JOIN publication p ON pi.publication_id = p.id
LEFT JOIN pickup_point pp ON pi.pickup_point_id = pp.id
WHERE pi.return_date IS NULL
AND (
(%(publication_name_filter) s IS NOT NULL AND p.name_ ILIKE
%(publication_name_filter) s) OR
(%(publication_id_filter) s IS NOT NULL AND p.id = %(publication_id_filter) s)
)
AND (%(author_filter) s IS NULL OR p.author ILIKE %(author_filter) s))
SELECT (SELECT COUNT(*) FROM BookHolders) AS total_holders,
COALESCE(
STRING_AGG(DISTINCT surname || ' ' || name_ || COALESCE(' ' || middle_name, '') || ' (Срок: ' || TO_CHAR(expiration_date, 'YYYY-MM-DD') || '); ' ||
'Her читателей с этой книгой) AS readers_with_book_list,
COALESCE(
(SELECT bh_inner.surname || ' ' || bh_inner.name_ ||
COALESCE(' ' || bh_inner.middle_name, '') ||
' (Срок: ' || TO_CHAR(bh_inner.expiration_date, 'YYYY-MM-DD') || '); '
FROM BookHolders bh_inner
ORDER BY bh_inner.expiration_date ASC
LIMIT 1),
'N/A') AS earliest_returner
FROM BookHolders
GROUP BY CASE WHEN (SELECT COUNT(*) FROM BookHolders) > 0 THEN 1 ELSE NULL END;

-- 13) query
WITH ReaderDetails AS (SELECT rf.id AS reader_id,

```

```

rfi.surname,
rfi.name_.,
rfi.middle_name,
rfi.category,
rfi.reader_type,
rfi.department,
rfi.faculty,
rfi.course,
rfi.group_.,
(SELECT MIN(lc_reg.issue_date)
FROM library_card lc_reg
WHERE lc_reg.reader_id = rfi.id) AS registration_date,
(SELECT MAX(ad_inner.departure_date)
FROM (SELECT s.id as reader_id_key, s.departure_date
FROM student s
WHERE s.id = rfi.id
AND s.departure_date IS NOT NULL
UNION ALL
SELECT t.id, t.departure_date
FROM teacher t
WHERE t.id = rfi.id
AND t.departure_date IS NOT NULL
UNION ALL
SELECT ue.id, ue.departure_date
FROM university_employee ue
WHERE ue.id = rfi.id
AND ue.departure_date IS NOT NULL
UNION ALL
SELECT sw.id, sw.departure_date
FROM science_worker sw
WHERE sw.id = rfi.id
AND sw.departure_date IS NOT NULL) ad_inner) AS departure_date,
lc.id AS library_card_id,
lc.issue_date AS lc_issue_date,
lc.expiration_date AS lc_expiration_date,
lcs.info AS lc_status
FROM reader_full_info rfi
LEFT JOIN library_card lc ON rfi.id = lc.reader_id AND lc.issue_date =
(SELECT MAX(lc_max.issue_date)
FROM library_card lc_max
WHERE lc_max.reader_id = rfi.id)
LEFT JOIN library_card_status lcs ON lc.status_id = lcs.id
WHERE rfi.surname
ILIKE %(surname_filter) s
AND %(name_filter) s IS NULL OR rfi.name_ ILIKE %(name_filter) s)
AND %(middle_name_filter) s IS NULL OR rfi.middle_name ILIKE %(middle_name_filter) s)
)
, ReaderViolations AS (
SELECT
vi.reader_id, v.name_ AS violation_name, v.violation_date AS violation_event_date, res.description_ AS restriction_imposed
FROM violation_instance vi
JOIN violation v
ON vi.violation_category_id = vid
LEFT JOIN restriction res ON v.restriction_id = res.id
WHERE vi.reader_id IN (SELECT reader_id FROM ReaderDetails)
)
, LostBookViolations AS (
SELECT
vi_lost.reader_id, v_lost.name_ AS lost_book_violation_name, v_lost.violation_date AS date_marked_lost
FROM violation_instance vi_lost
JOIN violation v_lost
ON vi_lost.violation_category_id = v_lost.id
WHERE vi_lost.reader_id IN (SELECT reader_id FROM ReaderDetails)
AND v_lost.name_ ILIKE %(lost_book_pattern) s -- Parameterized
)
SELECT rd.reader_id,
rd.surname,
rd.name_.,
rd.middle_name,
rd.category AS reader_base_category,
rd.reader_type,
rd.department,
rd.faculty,
rd.course,
rd.group_.,
TO_CHAR(rd.registration_date, 'YYYY-MM-DD') AS registration_date,
TO_CHAR(rd.departure_date, 'YYYY-MM-DD') AS departure_date,
rd.library_card_id,
TO_CHAR(rd.lc_issue_date, 'YYYY-MM-DD') AS lc_issue_date,
TO_CHAR(rd.lc_expiration_date, 'YYYY-MM-DD') AS lc_expiration_date,
rd.lc_status,
(SELECT COUNT(*)
FROM ReaderViolations rvio
WHERE rvio.reader_id = rd.reader_id) AS total_violations,
COALESCE(
(SELECT STRING_AGG(DISTINCT rvio.violation_name ||
(Дата: ' || TO_CHAR(rvio.violation_event_date, 'YYYY-MM-DD') ||
COALESCE(' Ограничение: ' || rvio.restriction_imposed, '') || '); ', ''))
FROM ReaderViolations rvio
WHERE rvio.reader_id = rd.reader_id),
'Нет нарушений') AS violations_list,
COALESCE(
(SELECT STRING_AGG(DISTINCT lb.lost_book_violation_name ||
(Зафиксировано: ' || TO_CHAR(lb.date_marked_lost, 'YYYY-MM-DD') || '); ', ''))
FROM LostBookViolations lb
WHERE lb.reader_id = rd.reader_id),
'Нет зафиксированных утерь книг (по типу нарушения)') AS lost_books_violations_list
FROM ReaderDetails rd;

```

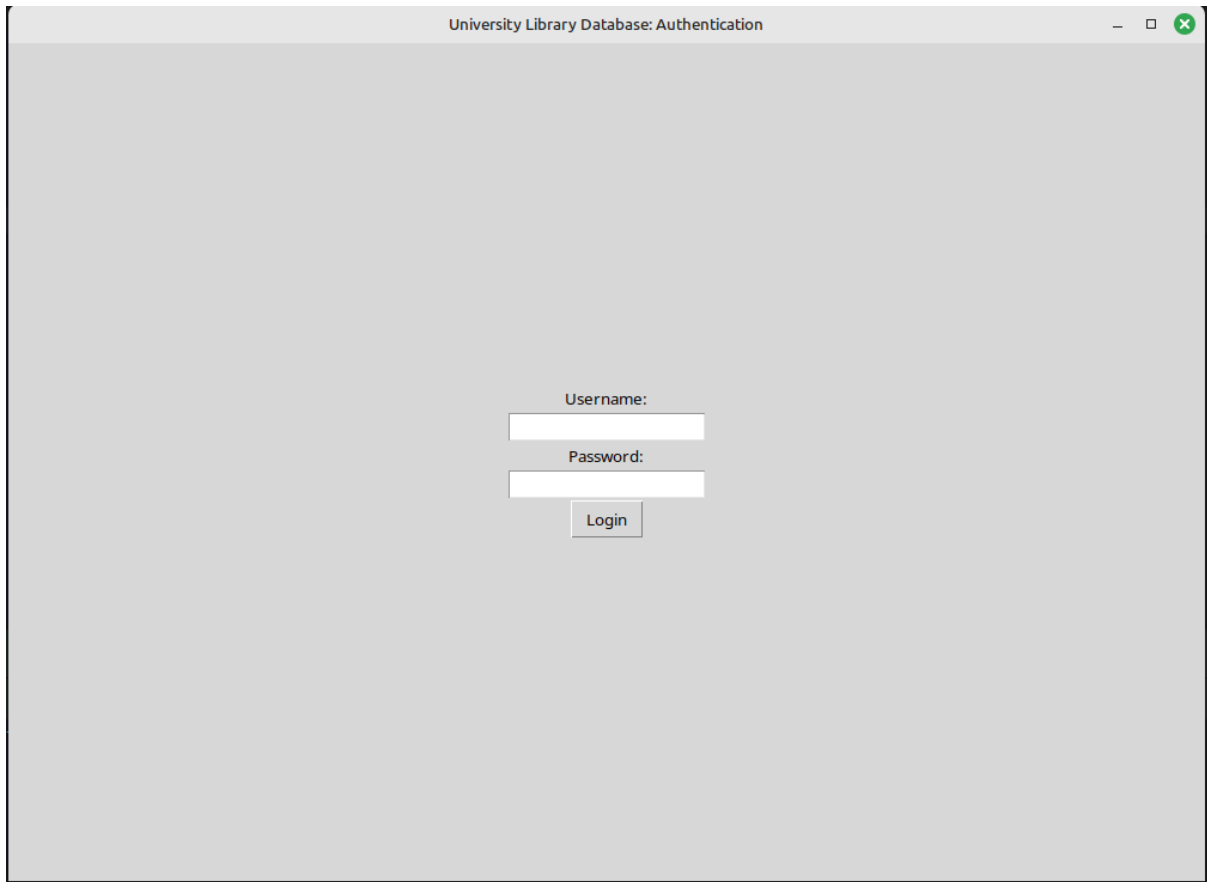
## Архитектура приложения

Приложение написано по архитектуре MVC (Model, View, Controller), где View отвечает за GUI компоненту приложения, а непосредственное взаимодействие с базой данных PostgreSQL осуществляется через Model.

## Аутентификация



Аутентификация в приложении проводится непосредственно через подключение к базе данных. После входа в приложение сменить пользователя и сессию, как следствие, нельзя. Чтобы сменить пользователя нужно заново запустить приложение.



University Library Database: Authentication

Username:

Password:

Login

## Разрешение гонки данных

Для разрешения конкурентного доступа двух запросов и подобных ситуаций гонки данных, сессия с базой данных создается с параметром `serializable`.

## Формы

В приложении присутствуют формы для заполнения таблиц данными, изменения данных в таблицах и форма с готовыми запросами. Также в каждом окне с формами есть возможность отправить запрос напрямую в базу данных.

University Library Database: Menu

Insert Form

Update Form

Search Form (Queries)

University Library Database: Insert Form

Select table: order\_

Or enter raw SQL query (INSERT/UPDATE/DELETE):

Execute Raw Query

Column	Type	Nullable	Value
id	bigint	NO	
library_card_id	bigint	YES	
publication_id	bigint	NO	
pickup_point_id	smallint	YES	
ordering_date	date	NO	
order_type	character	NO	

Submit

Back to Menu

University Library Database: Update Form

Select table:

Or enter raw SQL query (INSERT/UPDATE/DELETE):

Execute Raw Query

WHERE condition (e.g., id = %s):

Column	Type	New Value
id	bigint	<input type="text"/>
category	character	<input type="text"/>
surname	character varying	<input type="text"/>
name_	character varying	<input type="text"/>
middle_name	character varying	<input type="text"/>

Submit

Back to Menu

University Library Database: Search/Select Form

Enter SQL SELECT query:

Execute

Or choose a ready-made query:

1. Список и общее число читателей
2. Читатели-задолжники
3. Топ-20 наиболее часто заказываемых книг
- 4а. Книги, поступившие за последний год
- 4б. Книги, утерянные за последний год
- 5а. Пункты выдачи по числу читателей
- 5б. Пункты выдачи по числу читателей-задолжников
- 5в. Пункты выдачи по сумме задолженности (стоимости книг)
6. Книги, заказанные по МБА
7. Количество экземпляров книги (доступных)
8. Читатели, лишенные права пользования (> 2 мес. с даты нарушения)
- 9а. Новые читатели (по дате выдачи первой карты)
- 9б. Выбывшие читатели (по дате убытия)
- 10а. Книги, заказанные читателем
- 10б. Книги на руках у читателя
11. Наличие книги на абонементах (доступные экз.)
12. Кто имеет книгу и кто сдал раньше всех
13. Полная информация о читателе по фамилии

Execute Selected Ready Query

Back to Menu

Query Results				
id	category	surname	name_	middle_name
1	Абитуриент	Иванов	Петр	Сергеевич
2	Абитуриент	Петрова	Анна	Викторовна
3	Абитуриент	Сидоров	Алексей	Игоревич
4	Абитуриент	Кузнецова	Елена	Дмитриевна
5	Абитуриент	Михайлов	Иван	Александрович
6	Абитуриент	Васильева	Ольга	Николаевна
7	Абитуриент	Новиков	Дмитрий	Федорович
8	Абитуриент	Морозова	Светлана	Юрьевна
9	Сотрудник ВУЗа	Смирнов	Андрей	Владимирович
10	Сотрудник ВУЗа	Попова	Мария	Алексеевна
11	Сотрудник ВУЗа	Волков	Сергей	Михайлович
12	Сотрудник ВУЗа	Федорова	Екатерина	Ивановна
13	Сотрудник ВУЗа	Романов	Олег	Петрович
14	Сотрудник ВУЗа	Захарова	Татьяна	Борисовна
15	Сотрудник ВУЗа	Павлов	Денис	Григорьевич
16	Сотрудник ВУЗа	Козлова	Ирина	Леонидовна
17	Студент	Алексеев	Максим	Андреевич
18	Студент	Соколова	Виктория	Сергеевна
19	Студент	Лебедев	Артем	Олегович
20	Студент	Орлова	Дарья	Константиновна
21	Студент	Егоров	Никита	Евгеньевич
22	Студент	Степанова	Анастасия	Романовна
23	Студент	Ковалев	Владислав	Денисович
24	Студент	Ильина	Полина	Максимовна
25	Преподаватель	Борисов	Владимир	Анатолевич
26	Преподаватель	Антонова	Людмила	Геннадьевна
27	Преподаватель	Григорьев	Игорь	Степанович

## Степень готовности

Дата	Готовность
18.05.2025	Первая версия GUI готова (на данный момент с заглушками в бэкенде): формы вставки и редактирования пока что содержат поля только для прямых sql запросов, но форма для поиска данных уже имеет реализованный лист готовых запросов (указанных в задании)
26.05.2025	GUI имеет базовую функциональность; формы готовы (корректно вставляют, изменяют и отображают информацию), поля для запросов напрямую тоже готовы, начальные варианты готовых запросов написаны.
31.05.2025	Запросы завершены, архитектура описана в отчете, текст скриптов также в отчёте.

## Ссылки

1. <https://github.com/hellttek/databases>