Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №5**

по дисциплине «Базы данных»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

Кандидат тех. наук., доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

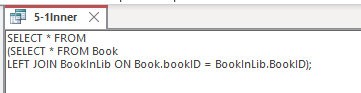
Санкт-Петербург, 2025

Оглавление

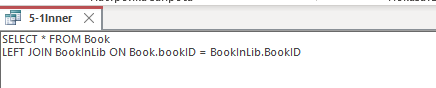
[Яндекс практикум 15](#_Toc190869791)

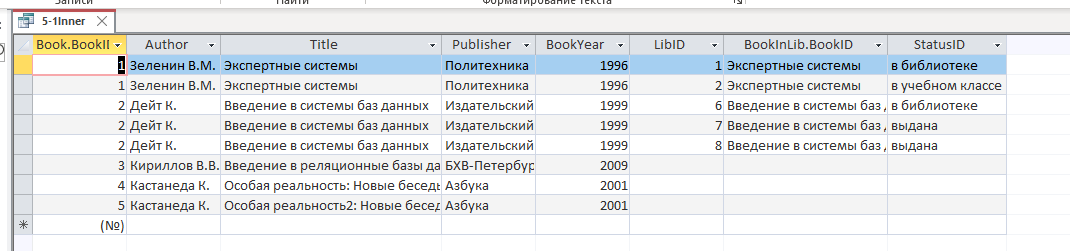
[Авиаперевозки 25](#_Toc190869792)

Задание. Используя базу из второй лабораторной работы, напишите запрос, который будет содержать выражение (подзапрос) SELECT в секции FROM. Проверьте его работу. Можно ли аналогичный запрос записать по-другому?

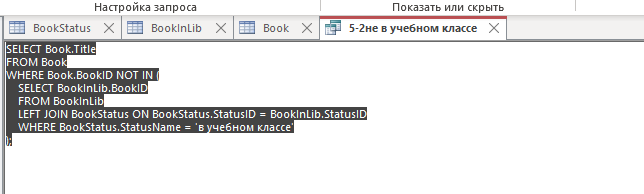


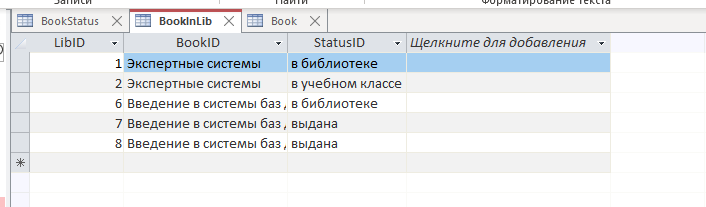




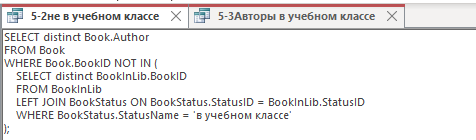


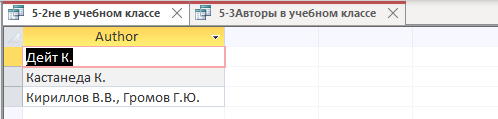
Задание. Напишите запрос, который выведет названия книг, ни одного экземпляра которых нет в учебном классе (т.е. ни у одного экземпляра текущий статус не равен «в учебном классе»). Обратите внимание, что в библиотеке, может быть, несколько экземпляров 2 одного издания с разными статусами, и ваш запрос должен корректно обрабатывать эту ситуацию.





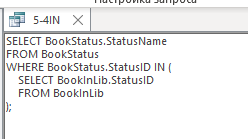
Задание. Измените предыдущий запрос. Пусть теперь он выводит без повторений авторов книг, таких что ни одной книги этого автора нет в учебном классе. При написании запроса надо учитывать, что у автора может быть несколько книг, каждая из которых, может быть, в нескольких экземплярах с разными статусами, поэтому запрос должен проверять что ни у одного экземпляра, ни одной книги данного автора нет указанного статуса.





Задание. Напишите аналогичный по смыслу запрос, используя выражение IN. Какой запрос и почему в подобных случаях будет эффективнее с точки зрения времени выполнения? (Если ответ на этот вопрос вызывает затруднение, прочитайте в разделе 7.7 про коррелированную и некоррелированную обработку подзапросов).

SELECT BookStatus.StatusName FROM BookStatus WHERE EXISTS (SELECT \* FROM BookInLib WHERE BookInLib.StatusID=BookStatus.StatusID);

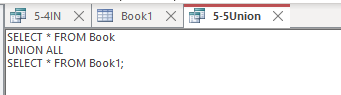


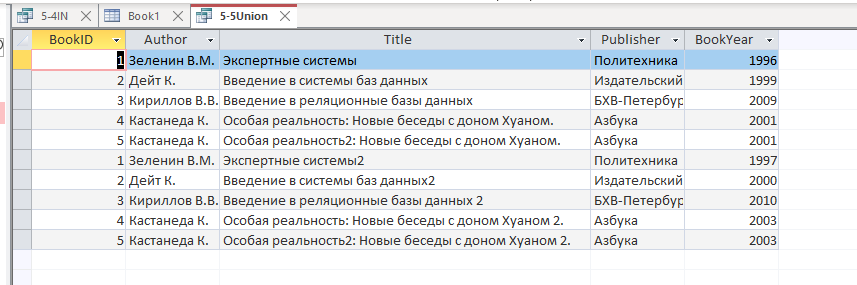
запрос с EXISTS использует коррелированный подзапрос. Это значит, что подзапрос выполняется для каждой строки таблицы BookStatus, проверяя, существует ли соответствующая запись в BookInLib. EXISTS останавливает выполнение подзапроса, как только находит первое совпадение, что может быть эффективнее. Если таблица BookStatus большая, то подзапрос будет выполняться многократно, что может замедлить выполнение.

Запрос с IN использует некоррелированный подзапрос. Подзапрос выполняется один раз, и его результат кэшируется. Подзапрос выполняется только один раз, что может быть быстрее, если таблица BookInLib небольшая. Если результат подзапроса большой (например, много StatusID), то операция IN может быть медленной, так как СУБД нужно сравнить каждое значение из BookStatus с большим списком.

* UNION (объединение);
* INTERSECT (пересечение) – не поддерживается Access;
* EXCEPT (вычитание) – не поддерживается Access.

Задание. В базе данных скопируйте таблицу Book и сохраните ее под именем Book1. Измените набор записей таблицы Book1, удалив из нее часть старых записей и добавив новых. Напишите запрос, возвращающий результат объединения множеств строк из этих таблиц. Замените UNION на UNION ALL объясните разницу между этими операторами.

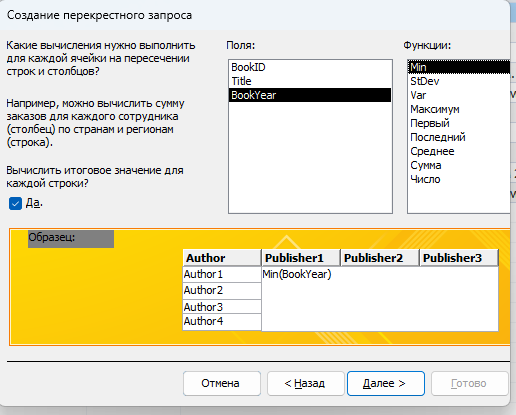


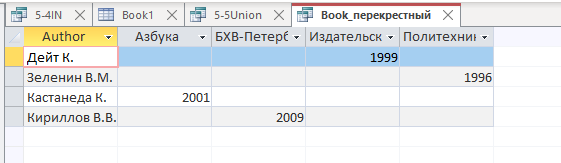


UNION Объединяет результаты двух или более запросов в один набор, удаляя дубликаты.

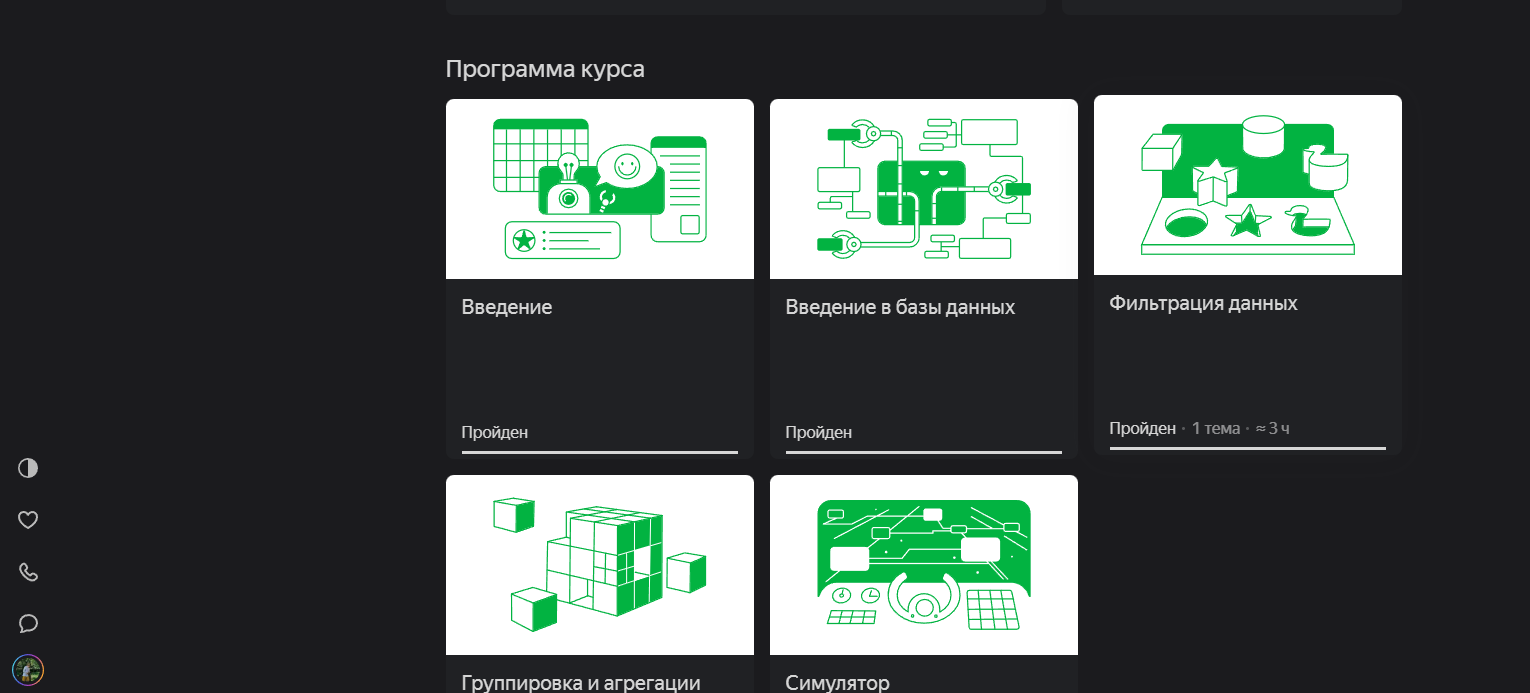
UNION ALL не удаляет дубликаты

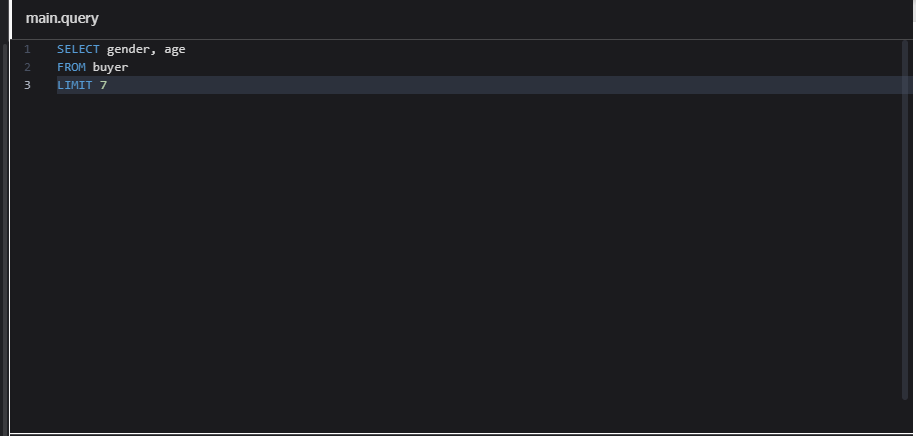
Задание. С помощью мастера постройте запрос, выводящий для каждого автора и издательства первый год, когда в этом издательстве вышла книга этого автора. Издательства выводите в заголовках столбцов, итоговое значение по строкам не отображайте. После проверки работы запроса, откройте его в режиме SQL. По справке ознакомьтесь с синтаксическими конструкциями, используемыми для создания перекрестного запроса.

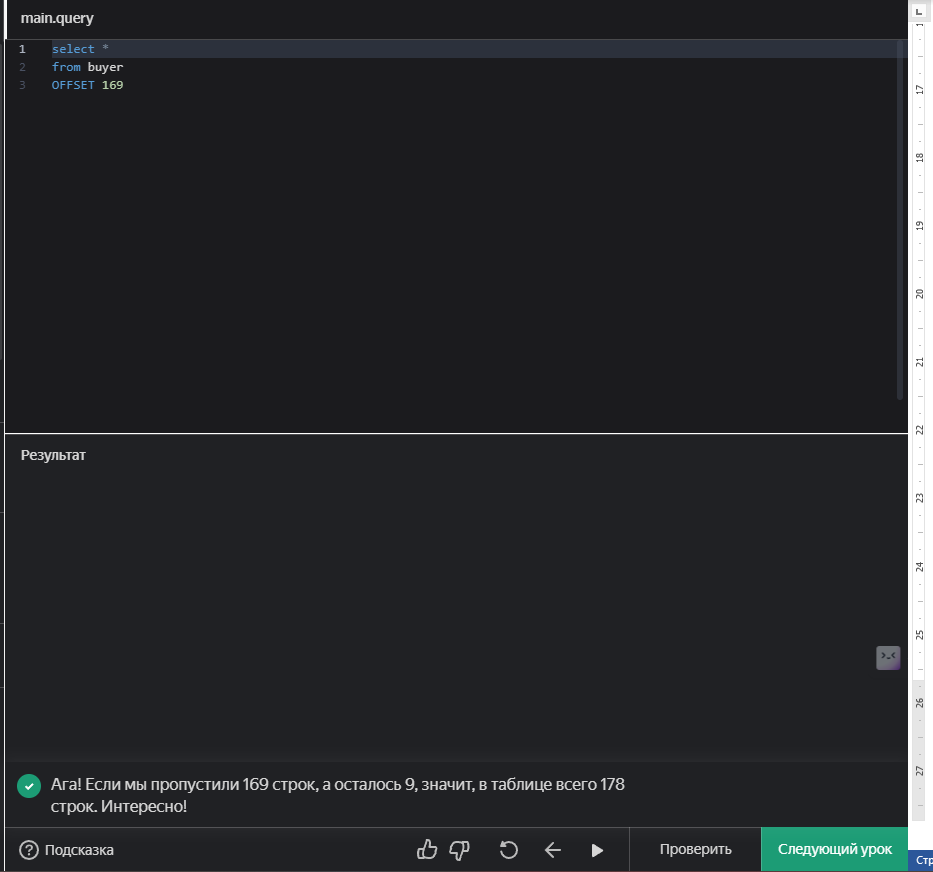


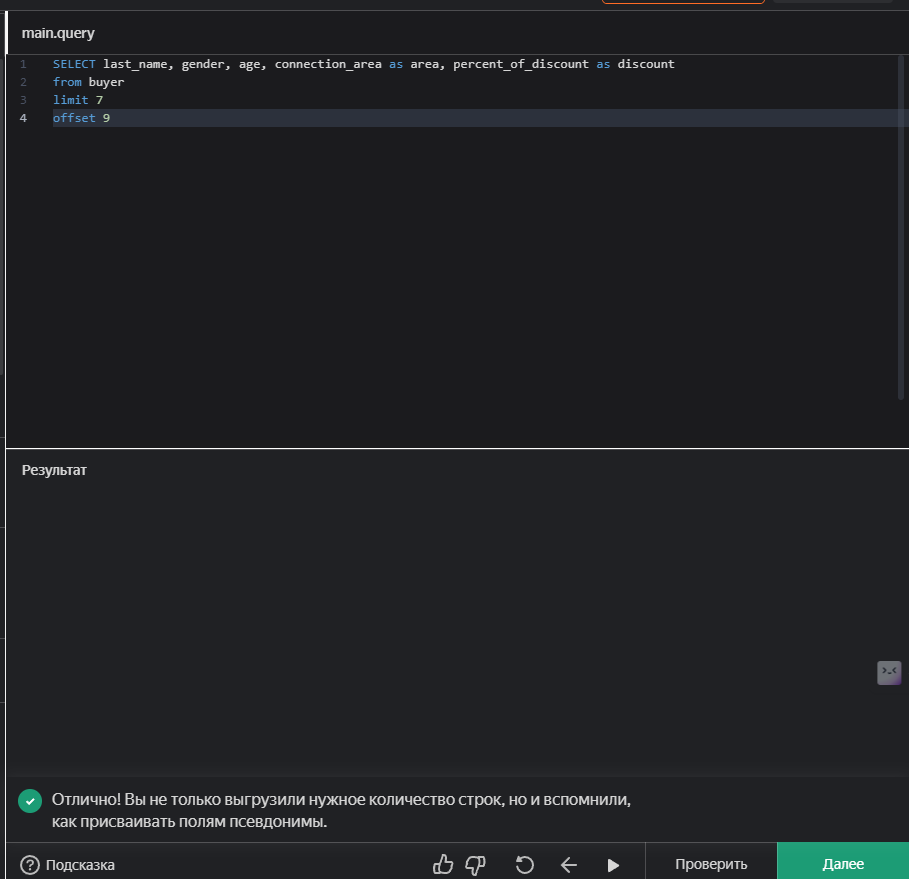


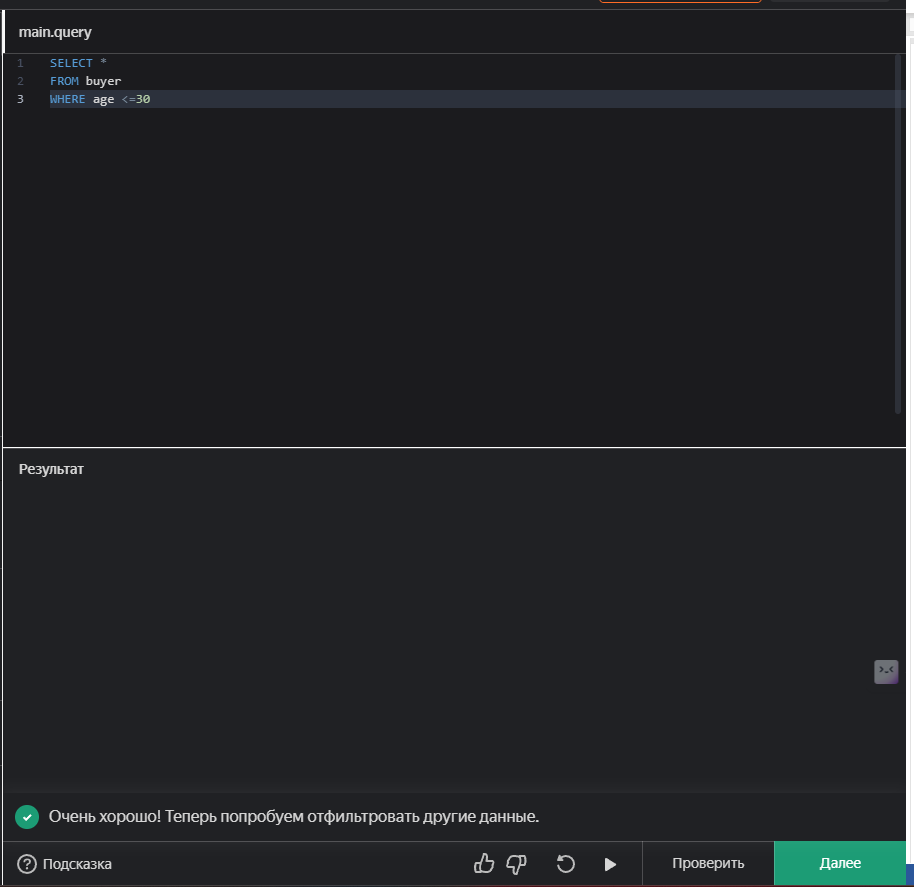
Яндекс практикум

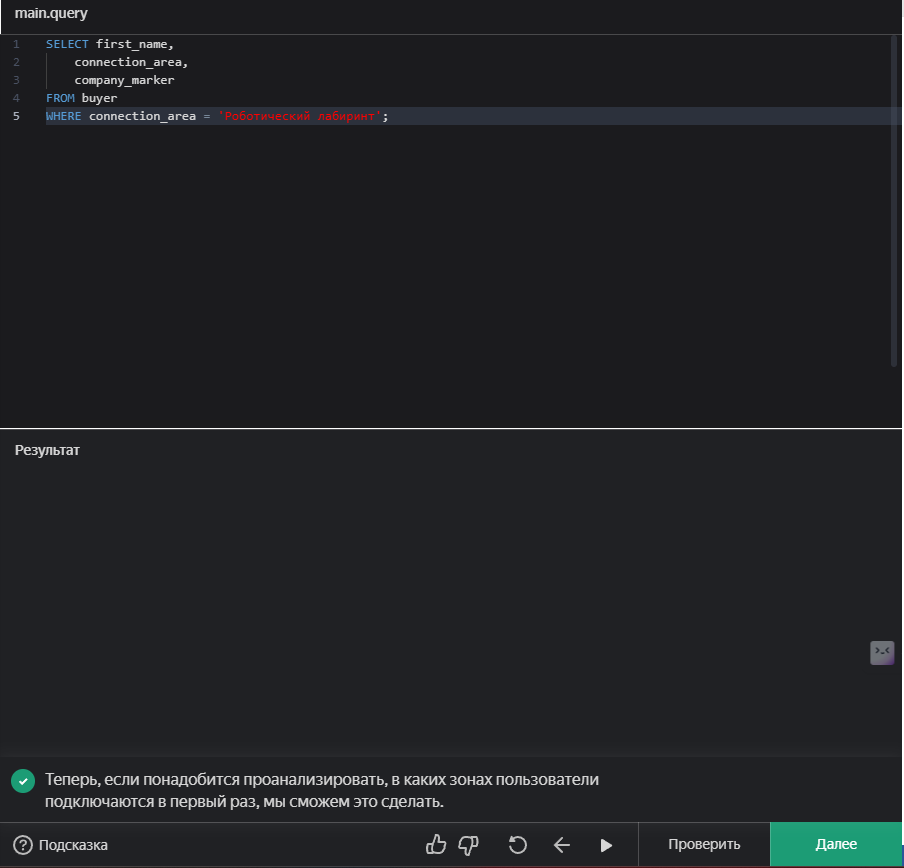


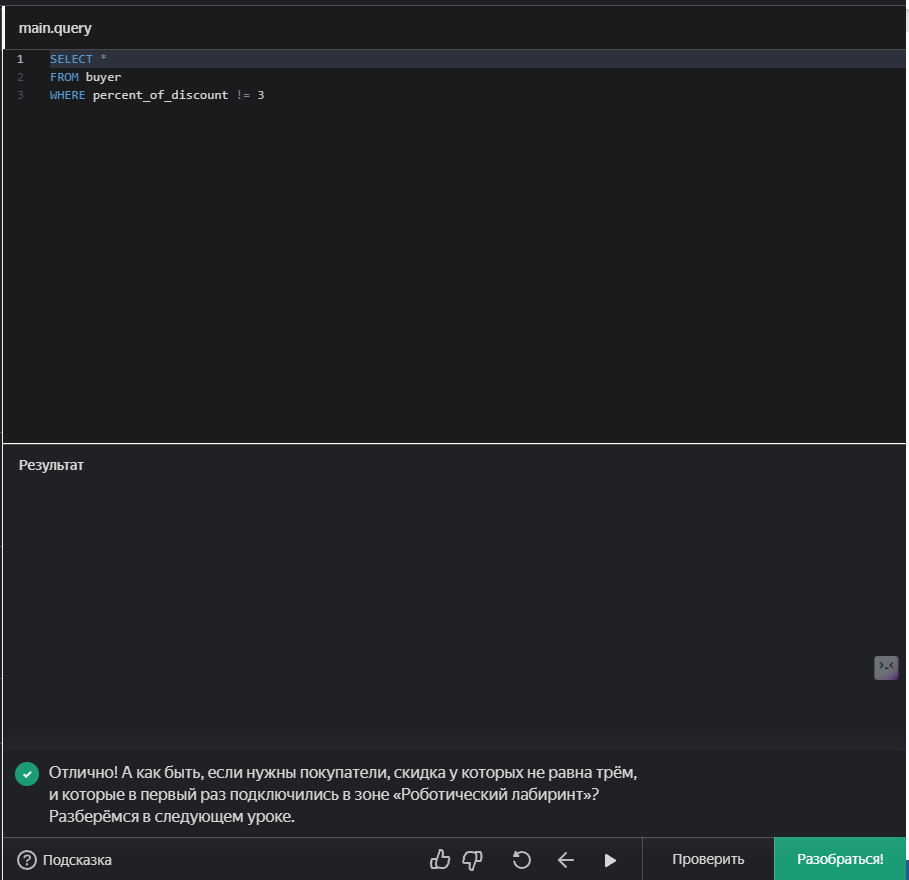












Пример разработки ER-модели

Рассмотрим применение ER-диаграмм на практике. При разработке ER-моделей мы должны получить следующую информацию о предметной области:

Список сущностей предметной области.

Список атрибутов сущностей.

Описание взаимосвязей между сущностями.

Процесс разработки ER-диаграммы является итерационным. Первый вариант диаграммы может быть не полным и ставить вопросы для уточнения с экспертами предметной области. Результатом уточнений является следующий вариант ER-диаграммы и т.д. пока не будут сняты все вопросы и уточнены все детали. ER-диаграммы в тоже время будут выполнять роль документации. Список сущностей как правило определяется всегда в первую очередь. Атрибуты и сущности могут определятся и дополнятся в любом порядке, в зависимости от доступности информации.

В качестве примера рассмотрим процесс разработки ER-диаграммы для проектирования информационной системы оптовой фирмы. В первую очередь мы должны изучить предметную область. Опросим сотрудников фирмы, изучим документацию, формы заказов, накладных и т.п.

Например, в ходе беседы с менеджером по продажам, выяснилось, что он (менеджер) считает, что проектируемая система должна выполнять следующие действия:

Хранить информацию о покупателях.

Печатать накладные на отпущенные товары.

Следить за наличием товаров на складе.

Мы помним, что сущности и атрибуты именуются существительными. Таким образом, чтобы выделить потенциальных кандидатов на роль сущностей и атрибутов, выпишем все существительные из заметок по итогам общения с менеджером. Непонятные термины будем выделять знаком вопроса.

Покупатель - явный кандидат на сущность.

Накладная - явный кандидат на сущность.

Товар - явный кандидат на сущность.

(?)Склад - а вообще, сколько складов имеет фирма? Если несколько, то это будет кандидатом на новую сущность.

(?)Наличие товара — это, скорее всего, атрибут, но атрибут какой сущности?

Сразу возникает очевидная связь между сущностями - “покупатели могут покупать много товаров” и “товары могут продаваться многим покупателям”.

Для уточнения поставленных вопросов мы вновь общаемся с менеджером и выясняем, что фирма имеет несколько складов. Причем, каждый товар может храниться на нескольких складах и быть проданным с любого склада.

Куда поместить сущности “Накладная” и “Склад” и с чем их связать? Спросим себя, как связаны эти сущности между собой и с сущностями “Покупатель” и “Товар”? Покупатели покупают товары, получая при этом накладные, в которые внесены данные о количестве и цене купленного товара. Каждый покупатель может получить несколько накладных. Каждая накладная обязана выписываться на одного покупателя. Каждая накладная обязана содержать несколько товаров (не бывает пустых накладных). Каждый товар, в свою очередь, может быть продан нескольким покупателям через несколько накладных. Кроме того, каждая накладная должна быть выписана с определенного склада, и с любого склада может быть выписано много накладных.

На данном этапе можно считать, что мы определили сущности и связи между ними. У нас не осталось вопросов, относительно них. Пока не осталось. На следующей итерации необходимо уточнить атрибуты сущностей. Беседуя с сотрудниками фирмы, мы выяснили следующее:

Каждый покупатель является юридическим лицом и имеет наименование, адрес, банковские реквизиты.

Каждый товар имеет наименование, цену, а также характеризуется единицами измерения.

Каждая накладная имеет уникальный номер, дату выписывания, список товаров с количествами и ценами, а также общую сумму накладной.

Накладная выписывается с определенного склада и на определенного покупателя.

Каждый склад имеет свое наименование.

Снова выпишем все существительные, которые будут потенциальными атрибутами, и проанализируем их:

Юридическое лицо - термин риторический, мы не работаем с физическими лицами. Не обращаем внимания.

Наименование покупателя - явная характеристика покупателя.

Адрес - явная характеристика покупателя.

Банковские реквизиты - явная характеристика покупателя.

Наименование товара - явная характеристика товара.

(?)Цена товара - похоже, что это характеристика товара. Отличается ли эта характеристика от цены в накладной?

Единица измерения - явная характеристика товара.

Номер накладной - явная уникальная характеристика накладной.

Дата накладной - явная характеристика накладной.

(?)Список товаров в накладной - список не может быть атрибутом. Вероятно, нужно выделить этот список в отдельную сущность.

(?)Количество товара в накладной — это явная характеристика, но характеристика чего? Это характеристика не просто “товара”, а “товара в накладной”.

(?)Цена товара в накладной - опять же это должна быть не просто характеристика товара, а характеристика товара в накладной. Но цена товара уже встречалась выше — это одно и то же?

Сумма накладной - явная характеристика накладной. Эта характеристика не является независимой. Сумма накладной равна сумме стоимостей всех товаров, входящих в накладную.

Наименование склада - явная характеристика склада.

В ходе дополнительной беседы с менеджером удалось прояснить различные понятия цен. Оказалось, что каждый товар имеет некоторую текущую цену. Эта цена, по которой товар продается в данный момент. Естественно, что эта цена может меняться со временем. Цена одного и того же товара в разных накладных, выписанных в разное время, может быть различной. Таким образом, имеется две цены - цена товара в накладной и текущая цена товара.

С возникающим понятием “Список товаров в накладной” все довольно ясно. Сущности “Накладная” и “Товар” связаны друг с другом отношением типа много-ко-многим. Такая связь, должна быть расщеплена на две связи типа один-ко-многим. Для этого требуется дополнительная сущность. Этой сущностью и будет сущность “Список товаров в накладной”. Связь ее с сущностями “Накладная” и “Товар” характеризуется следующими фразами - “каждая накладная обязана иметь несколько записей из списка товаров в накладной”, “каждая запись из списка товаров в накладной обязана включаться ровно в одну накладную”, “каждый товар может включаться в несколько записей из списка товаров в накладной”, “каждая запись из списка товаров в накладной обязана быть связана ровно с одним товаром”. Атрибуты “Количество товара в накладной” и “Цена товара в накладной” являются атрибутами сущности “Список товаров в накладной”.

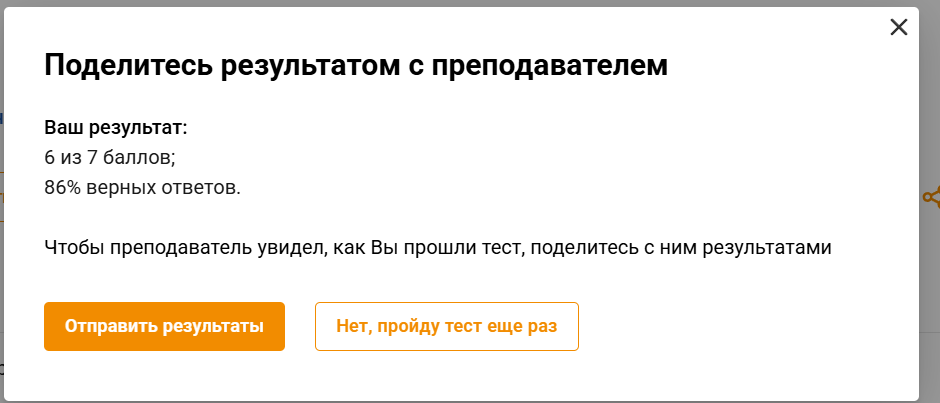
Точно также поступим со связью, соединяющей сущности “Склад” и “Товар”. Введем дополнительную сущность “Товар на складе”. Атрибутом этой сущности будет “Количество товара на складе”. Таким образом, товар будет числиться на любом складе и количество его на каждом складе будет свое.

Мы учли все атрибуты, у нас не осталось связей “много-ко-многим”, а следовательно можно считать что на данном этапе ER-диаграмма разработана в соответствие с требованиями. В последующих версиях системы или при возникновении новых требований, эту диаграмму можно также дополнять или модифицировать в рамках подготовки новых версий разрабатываемой информационной системы.

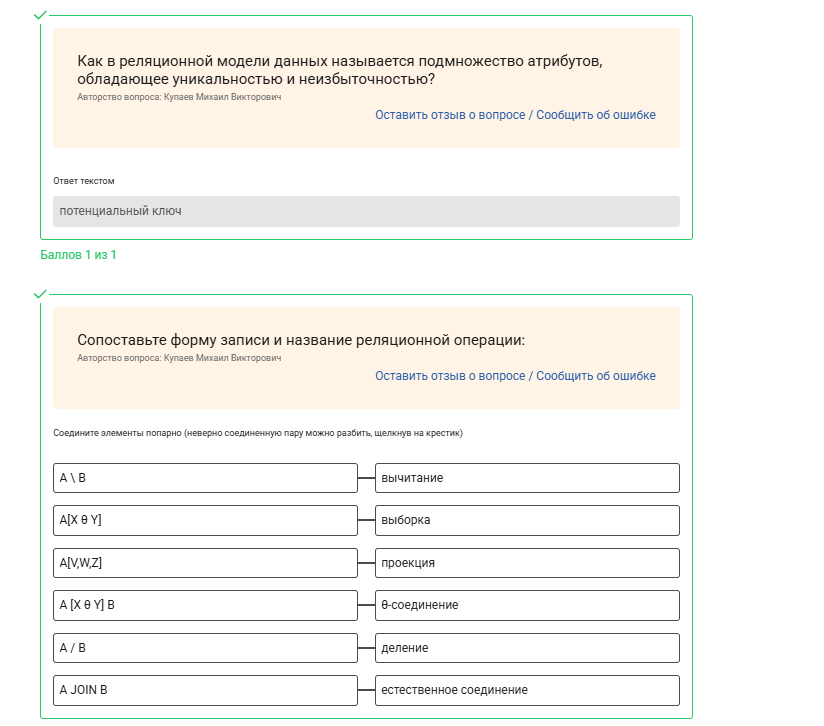
Концептуальные и физические ER-модели

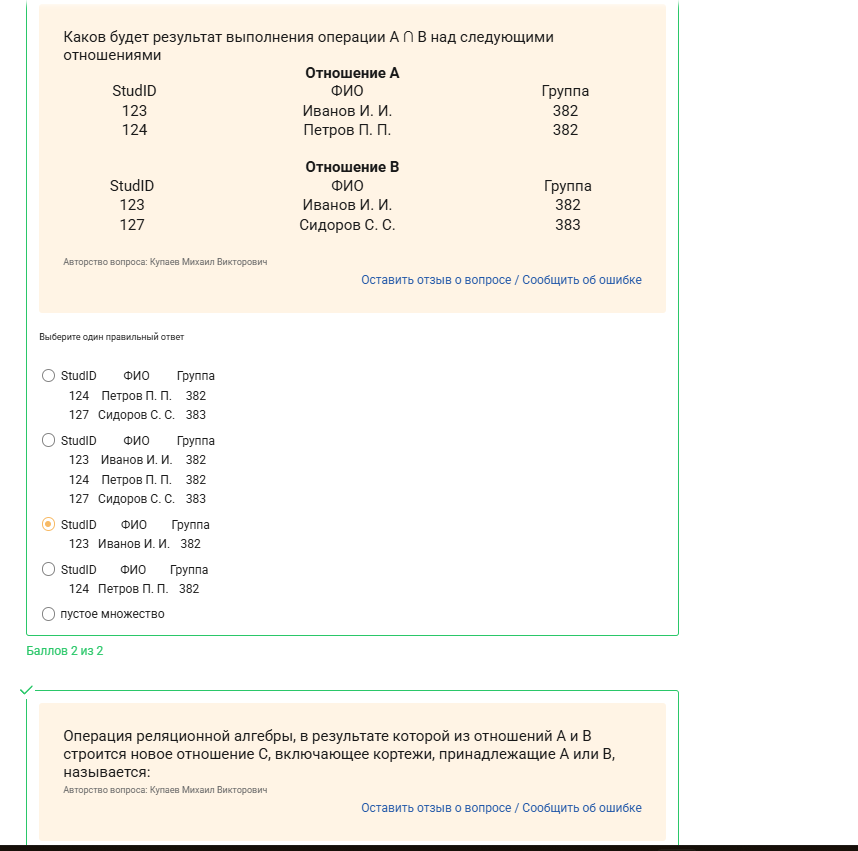
Разработанный выше пример ER-диаграммы является примером концептуальной диаграммы. Это означает, что диаграмма не учитывает особенности конкретной СУБД. По данной концептуальной диаграмме можно построить физическую диаграмму, которая уже будут учитываться такие особенности СУБД, как допустимые типы и наименования полей и таблиц, ограничения целостности и т.п. Физический вариант диаграммы может выглядеть, например, следующим образом:

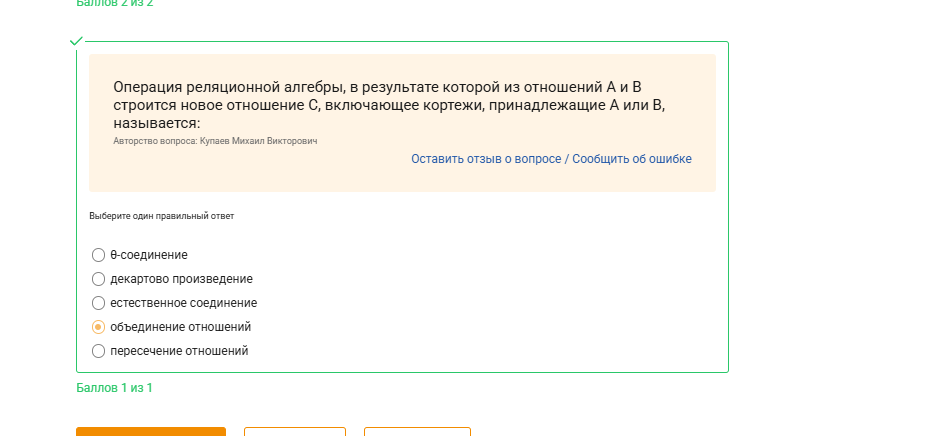
На данной диаграмме каждая сущность представляет собой таблицу базы данных, каждый атрибут становится колонкой соответствующей таблицы. Обращаем внимание на то, что во многих таблицах, например, “CUST\_DETAIL” и “PROD\_IN\_SKLAD”, соответствующих сущностям “Запись списка накладной” и “Товар на складе”, появились новые атрибуты, которых не было в концептуальной модели - это ключевые атрибуты родительских таблиц, мигрировавших в дочерние таблицы для того, чтобы обеспечить связь между таблицами посредством внешних ключей.





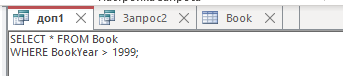


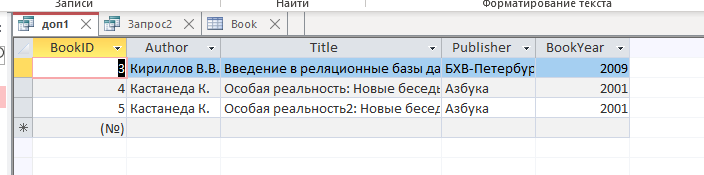


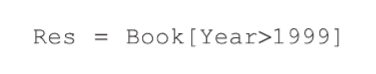


Дополнительные задания

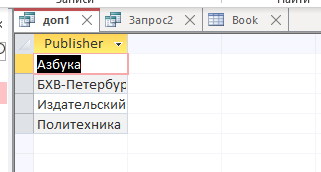
1. Найти книги, изданные после 1999 года.

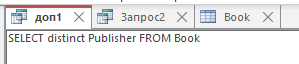






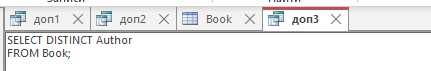
1. Перечень всех издательств, о которых имеется упоминание в нашей БД.

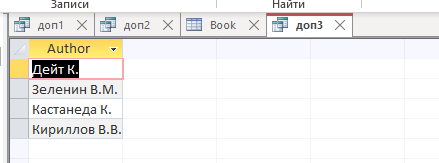


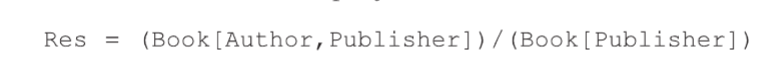




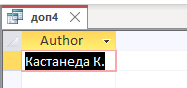
1. Список авторов, издававшихся во всех известных нам издательствах.

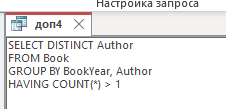


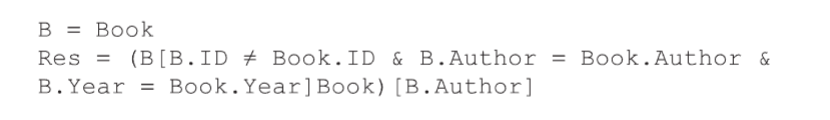




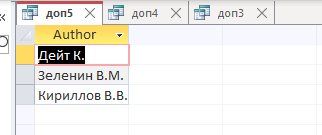
1. Список авторов, чьи книги более одного раза издавались в одном и том же году.

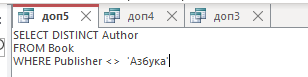


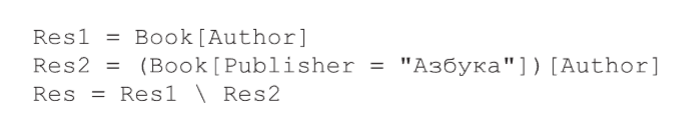


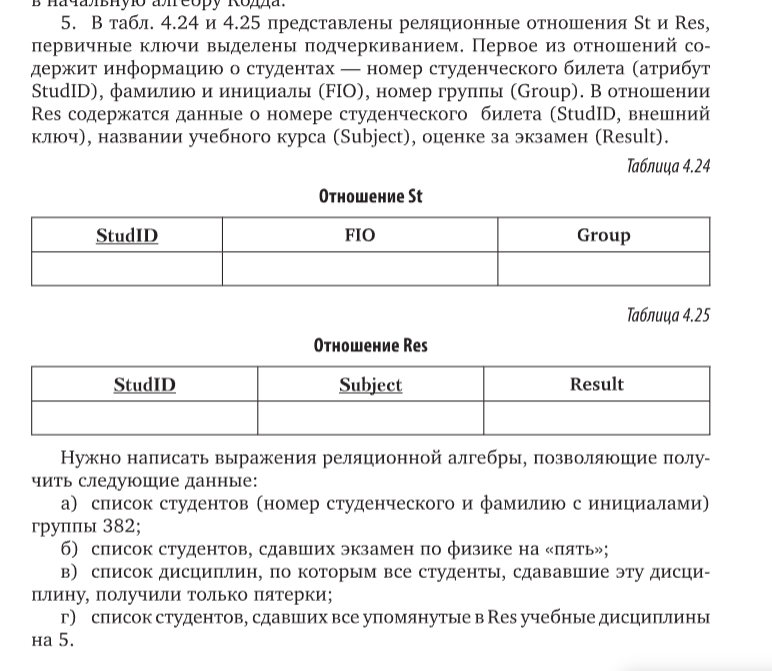


1. Список авторов, чьи книги ни разу не издавались в издательстве «Азбука».









1. SELECT St.StudID, FIO

FROM St

WHERE Group = 382

Изображение выглядит как Шрифт, типография, текст, каллиграфия

Автоматически созданное описание

1. SELECT DISTINCT StudID, FIO

FROM St

INNER JOIN Res

ON St.StudID = Res.StudID

WHERE Subject = ‘физика’ AND Result = 5



1. SELECT Subject

FROM Res

GROUP BY Subject

Having MIN(Result) = 5 AND MAX(Result) = 5;

Изображение выглядит как Шрифт, текст, типография, каллиграфия

Автоматически созданное описание

1. SELECT St.StudID, FIO

FROM St

INNER JOIN Res ON St.StudID = Res.StudID

GROUP BY St.StudID, FIO

HAVING MIN(result) = 5 AND MAX(Result) = 5;

Изображение выглядит как Шрифт, текст, типография, каллиграфия

Автоматически созданное описание

ПРАКТИЧЕСКИЕ ЗАДАНИЯ ЧЕТВЁРТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ»

1. **Определить адреса клиентов, заказывавших игры с доставкой**

Ответ:



Пояснение:

* 1. Соединяем таблицы К и З по ключу Ид клиента, чтобы в одной таблице были и заказы, и информация о клиентах.
  2. Фильтруем заказы, оставляя только те, у которых Получение = 'Доставка'.
  3. Проецируем (выбираем) только поле Адрес.

1. **Определить название и производителя игры (игр), в которую можно играть самой большой компанией.**

Ответ:

Мы хотим выявить игры, у которых есть другие игры с большим значением "Макс игроков".

Для этого делаем самосоединение таблицы игр И:



Где:

* И и И' — это одна и та же таблица, но рассматриваемая дважды (первый раз — как оригинальная, второй — как копия для сравнения).
* И.Макс игроков < И'.Макс игроков — условие, означающее, что у И' есть игра с большим числом игроков, чем у И.
* Затем выбираем только названия и производителей игр, которые уступают по числу игроков хотя бы одной другой игре.

Теперь, чтобы найти игру (игры) с наибольшим числом игроков, мы просто:

1. Берём все игры (И).
2. Убираем из них те, что оказались в R\_1 (то есть у которых есть конкурент с большим числом игроков).

Это выражается так:



1. **Определить табельный номер сотрудника, назначенного ответственным только за один заказ (на момент выполнения запроса).**

Ответ:

Для начала необходимо найти сотрудников, назначенных более чем на один заказ.

Для поиска сотрудников, назначенных на несколько заказов, используется самосоединение отношения 3 с самим собой. Условие соединения:

* Табельные номера сотрудников должны совпадать:



* Номера заказов должны различаться:



Результат можно выразить следующим образом:



Где R1​ — множество табельных номеров сотрудников, назначенных более чем на один заказ.

Из полного списка сотрудников, имеющих назначенные заказы (ΠТаб номер(3)), исключаются сотрудники из множества R1​, поскольку они ответственны за более чем один заказ.

Операция выражается следующим образом:

Изображение выглядит как Шрифт, текст, типография, каллиграфия

Автоматически созданное описание

1. **Определить имена и e-mail клиентов, заказавших игры с названием Диксит от всех производителей (не обязательно за один раз). Решить двумя способами.**

Ответ:

**СПОСОБ 1:**

Первым делом определяем, какие производители выпускают "Диксит".

Запрос:



Пояснения:

1. 

* Оператор σ (селекция) выбирает только те строки из отношения И (Игры), в которых Название = "Диксит".
* Это оставляет только строки, где встречается "Диксит" с его производителями.

1. 

* Оператор π (проекция) выбирает только два столбца: Название и Производитель.
* Убираются ненужные столбцы, такие как номер игры и другие характеристики.

Теперь определяем кто и у каких производителей заказывал игры. Для этого нужно соединить три отношения:

* К — содержит клиентов (имя, e-mail).
* 3 — связывает клиентов с заказами (номер заказа).
* ПЗ — связывает заказы с играми (название, производитель).

Запрос:



Пояснения:

1. 

* Это соединение (⋈) трёх отношений:
  + К (Клиенты) — содержит информацию о клиентах (Имя, Почта, Номер заказа).
  + 3 (отношение "Клиент-заказ") связывает клиентов с их заказами по номеру заказа.
  + ПЗ (отношение "Заказ-игры") показывает, какие игры входят в какой заказ.
* Связь осуществляется по номеру заказа, поскольку именно через него можно узнать, какие игры клиент заказал.

Теперь необходимо найти клиентов, которые заказывали "Диксит" от всех производителей. Это делается с помощью операции деления:

1. 

Оператор П выбирает только нужные столбцы:

* Имя и Почта (кто заказал)
* Название игры и Производитель (какую игру заказали и от какого производителя).

Теперь необходимо найти клиентов, которые заказывали "Диксит" от всех производителей. Это делается с помощью операции деления:



Где:

* R2​ содержит пары (клиент, игра, производитель).
* R1​ содержит всех производителей игры "Диксит".
* Деление выбирает только тех клиентов, у которых есть "Диксит" от всех производителей из R1​.

**СПОСОБ 2**

Шаг 1 и шаг 2 такие же, как и в способе 1:

* Шаг 1: определяем список всех игр "Диксит" с их производителями (R₁).
* Шаг 2: определяем клиентов и их заказы (R₂).

Далее для определения клиентов, которые должны были заказать "Диксит" у всех производителей, создаём всевозможные пары клиентов с играми:



Пояснение:

* Операция декартова произведения × создаёт все возможные сочетания клиентов с каждым возможным производителем игры "Диксит".
* Это означает, что для каждого клиента мы получаем строки, в которых он как будто бы заказал все варианты "Диксит".

Результат (R₃) — потенциальные заказы клиентов:

Далее определяем заказы, **которых не хватает клиентам** для выполнения условия задачи:



Пояснения:

* Операция вычитания ∖ исключает из списка потенциальных заказов (R₄) те строки, которые реально существуют в R₂ (фактические заказы клиентов).
* В итоге в R5​ останутся те заказы, которых не хватает у клиентов, чтобы соответствовать условию задачи (заказ "Диксита" от всех производителей).

Результат (R₅) — отсутствующие заказы

Теперь оставляем только клиентов, которые не попали в R₅ (то есть заказали "Диксит" у всех производителей):

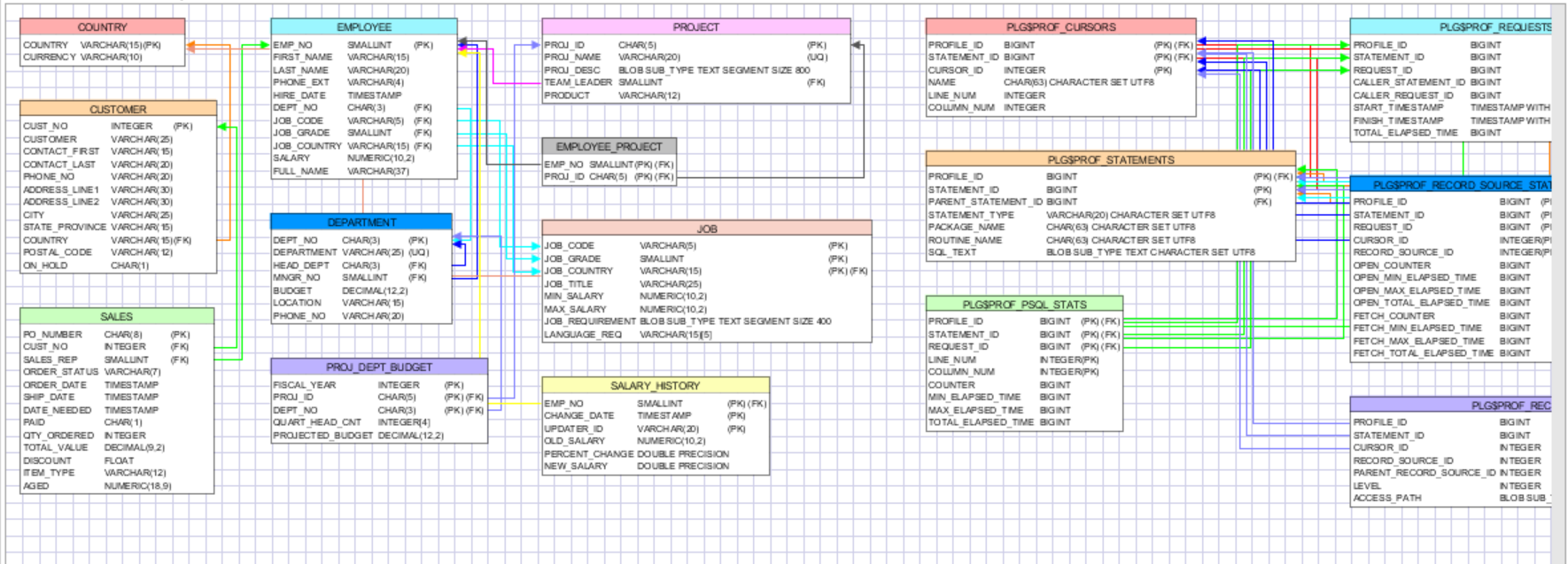


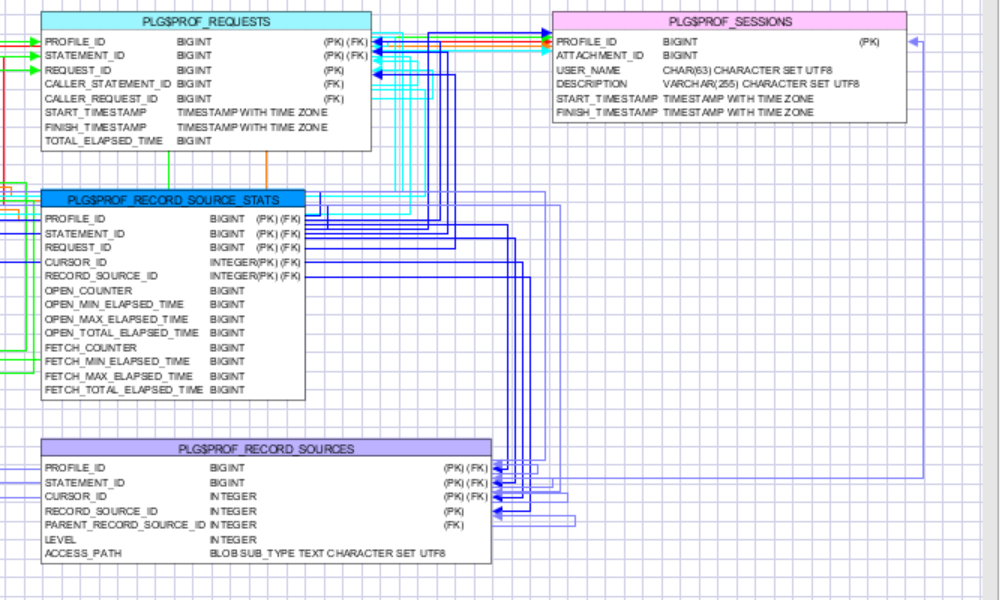
Пояснения:

* В R5​ находятся клиенты, которые не выполнили условия задачи.
* Операция проекции ΠИмя, Почта ​ в R6 исключает этих клиентов, оставляя только тех, кто соответствует требованиям (т.е. заказывал "Диксит" у всех производителей).

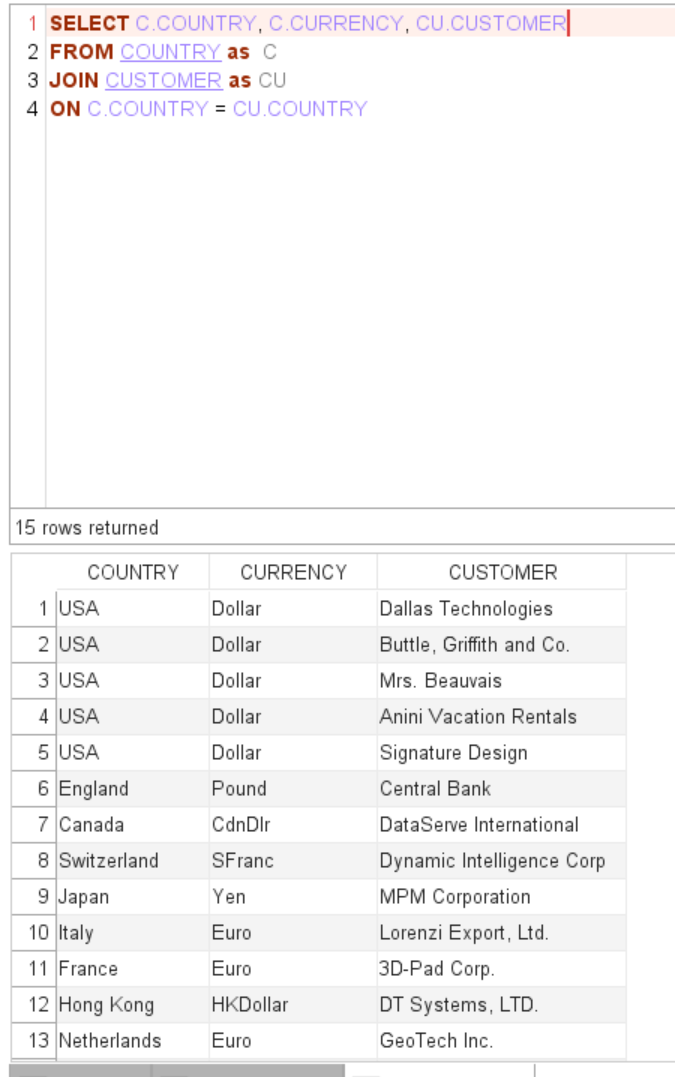
Результат (R₆) — окончательный список клиентов

1. Выполните аналогичные действия и добавьте диаграмму в свой отчет. При закрытии диаграммы вам ее предложат сохранить во внутреннем формате Red Expert или в одном из графических форматов.

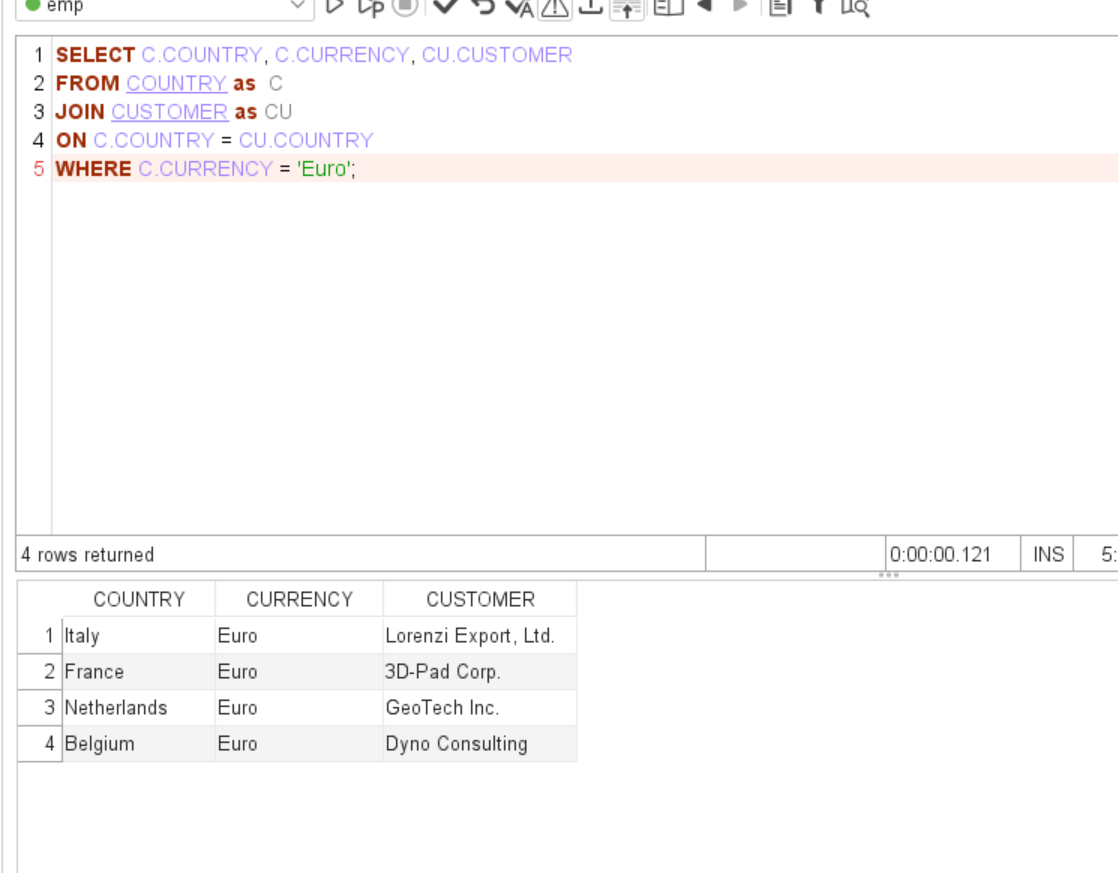




2. Напишите запрос, который выведет названия и валюту стран, в которых есть заказчики, и названия заказчиков из этих стран. Чтобы компактнее записывать соединения, в секции FROM для таблиц рекомендуется задавать короткие псевдонимы из одной-двух букв.

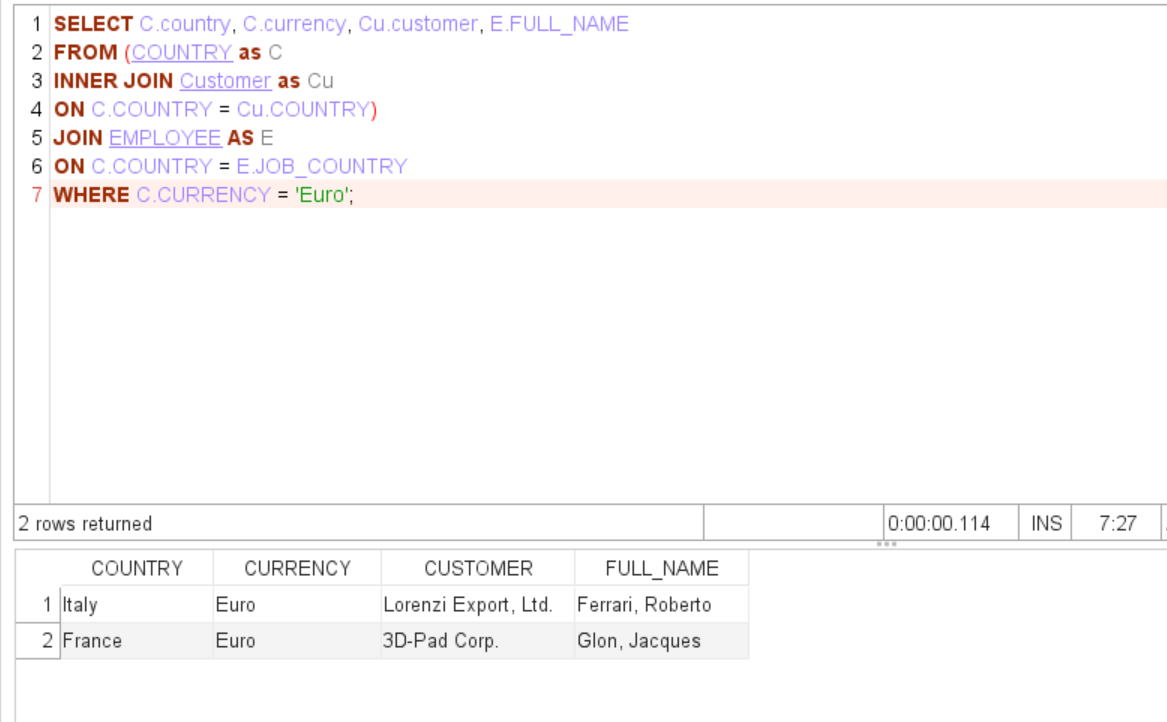


3. Измените предыдущий запрос таким образом, чтобы выводились валюта и названия всех стран, где валюта называется ’Euro’, и название заказчиков из этих стран, если в стране заказчики есть (если нет - выводим пустое поле). В каких из выведенных стран нет заказчиков?



4. Теперь обратите внимание на таблицу EMPLOYEE, где есть внешний ключ JOB\_COUNTRY, ссылающийся на таблицу COUNTRY.

Напишите запрос, выводящий страну, валюту, заказчика и полное имя сотрудника (FULL\_NAME) для тех стран с валютой Euro, в которых есть заказчики и работает кто-то из сотрудников. Что это за страны?



5. Задействуем еще таблицу с продажами (заказами) SALES.

Выведите для каждого заказа информацию о номере заказа (PO\_NUMBER), объеме заказа (TOTAL\_VALUE), названии заказчика, стране заказчика (обозначим столбец CUSTOMER COUNTRY, что надо сделать, если псевдоним с пробелом?), полном имени сотрудника,

стране работы сотрудника (обозначим EMPLOYEE COUNTRY) для таких заказов, где заказчик и обслуживающий заказ сотрудник были из разных стран.

