Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №8

За 7 семестр

По дисциплине «Компьютерное моделирование и анализ данных»

Выполнил: студент 4 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Чичурин А.В.

Брест 2022

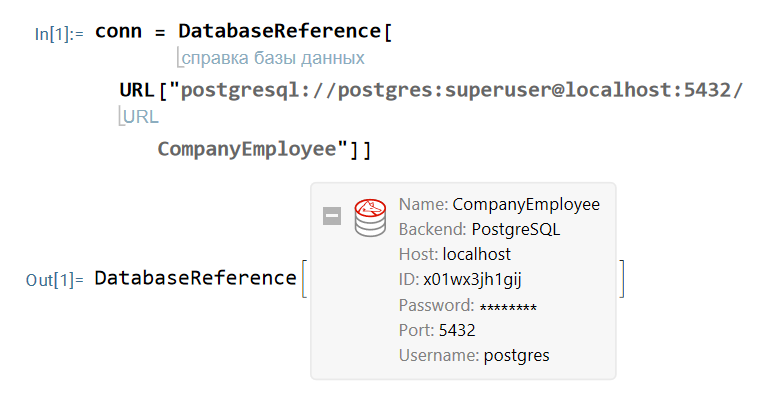
**Лабораторная работа №8**

**Работа с базами данных**

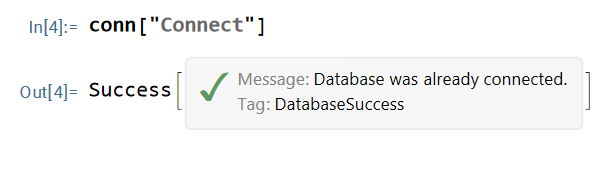
Прямая интеграция реляционных баз данных в свою работу может усложниться по ряду причин. Во-первых, многие эксперты не обладают знанием SQL, необходимыми для непосредственного использования реляционных баз данных. Во-вторых, существуют различия между диалектами SQL и поддерживаемыми наборами функций для различных серверных частей баз данных. По этим причинам тесная интеграция реляционных баз данных непосредственно в язык очень полезна. Это позволяет пользователям сосредоточиться на решении своих проблем, а не на предоставлении технической инфраструктуры.

В Wolfram, для огромного количества случаев, когда нужно запрашивать базу данных, можно взаимодействовать с базой данных через высокоуровневые запросы Entity Framework, построенные с использованием языка функциональных запросов Entity framework. Затем платформа отвечает за перевод запроса на соответствующий диалект SQL, выполнение его на стороне базы данных и возврат результатов обратно пользователю в соответствующем формате. Entity Framework для Wolfram в контексте реляционных баз данных во многом похож на то, что представляют собой объектно-реляционные отображения (ORM) для других языков. Несколько упрощая, сущности сопоставляются с отдельными строками таблиц базы данных, классы сущностей сопоставляются с таблицами базы данных, а свойства сущностей сопоставляются со столбцами базы данных.

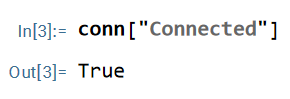
Чтобы начать работать с реляционной базой данных, нужно сначала создать ссылку для соединения с ней. Используем для этого DatabaseReference.



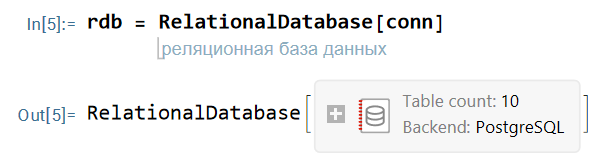
С помощью функции DatabaseConnect или [“Connect”] попробуем установить соединение с базой данных по ранее созданной ссылке.



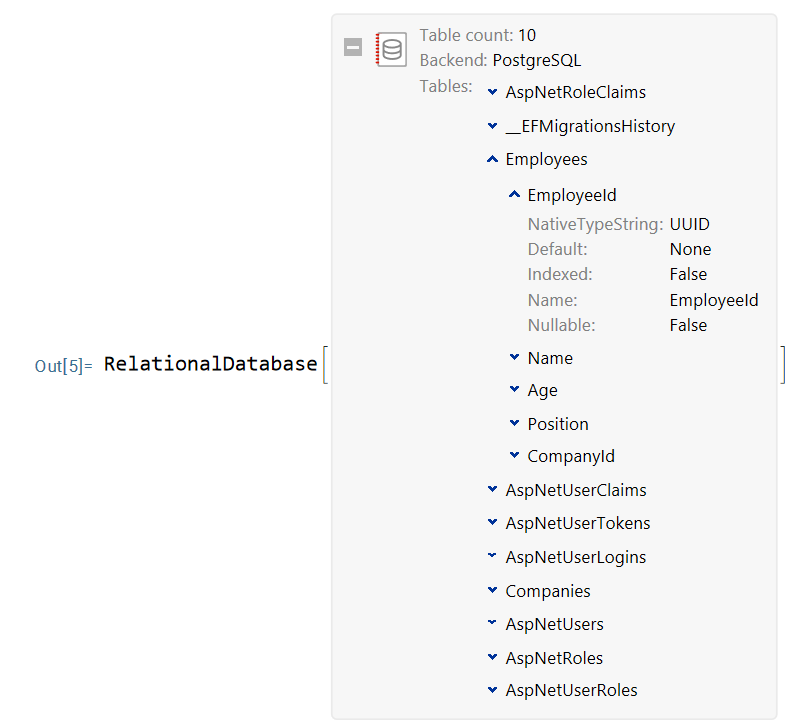
Проверим, действительно ли прошло соединение с базой данных.



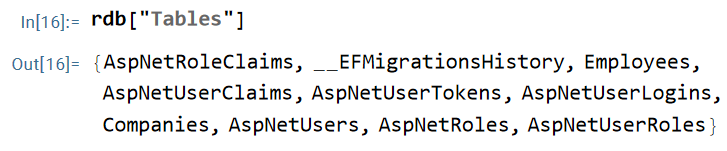
Создадим с помощью функции RelationalDatabase объект RelationalDatabase. Он содержит полную структуру базы данных, на которую мы ссылаемся (таблицы, колонки, ограничения и так далее) и он может быть использован как для визуального осмотра, так и для программного вытягивания этой информации, которая нам интересна.



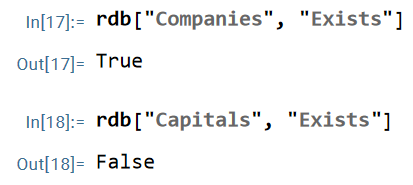
Посмотрим структуру базу данных более детально.



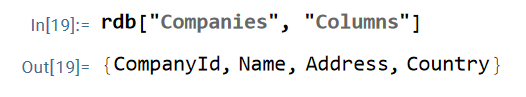
Вытянем просмотренную выше информацию программным способом. Задав параметр “Tables” можно получить список все таблиц в базе данных.



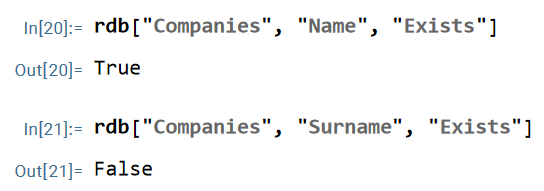
Также можно проверить, существует ли указанная таблица в базе данных. Для этого нужно указать название предполагаемой таблицы и “Exists”.



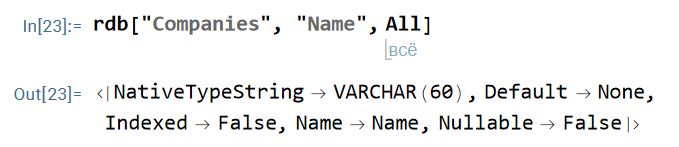
Задав как параметр название существующей таблицы и “Columns”, можно получить список всех столбцов данной таблицы.



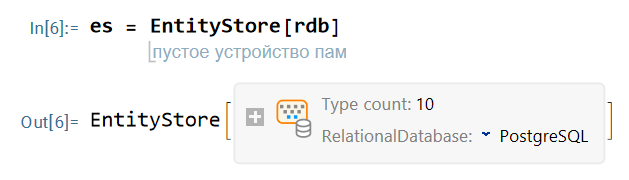
Аналогично проверке на наличие таблицы, можно сделать проверку на наличие определенного столбца.



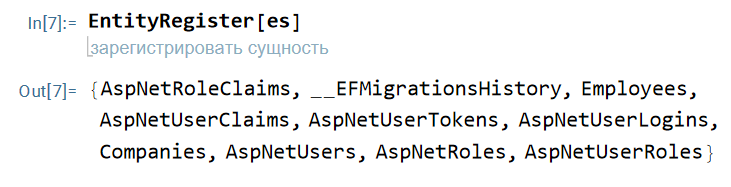
Напоследок, можно получить список всех свойств определенной колонки таблицы, а также их значения.



Создадим поддерживаемой базой данных объект EntityStore из объекта RelationalDatabase. В нем будут находится все сущности, которые находятся в нужной нам базе данных.

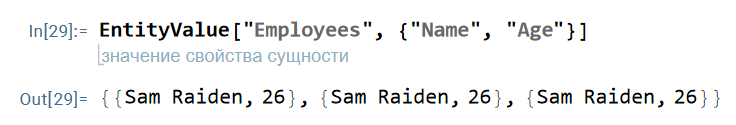


Затем зарегистрируем сущности в объекте EntityStore, чтобы можно было получить к ним доступ напрямую через Entity.

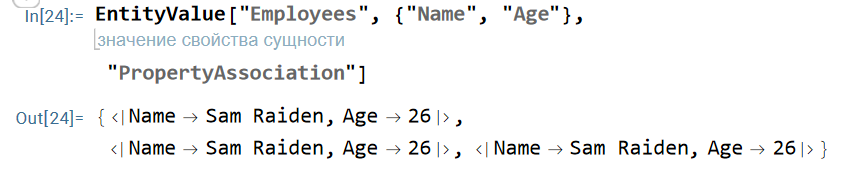


Теперь можно начинать работу с самими таблицами в базе данных.

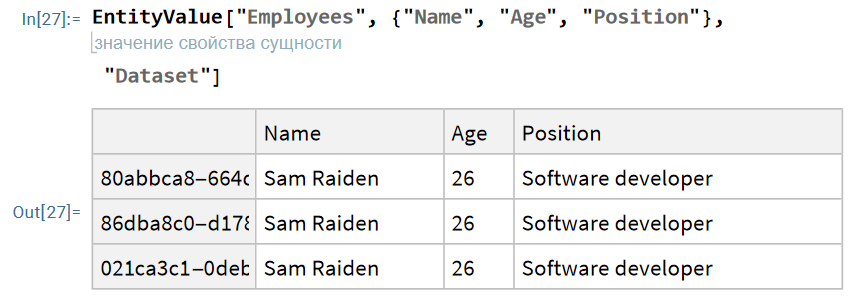
Для примера извлечем из таблицы Employees имя и фамилию работника, а также его возраст. Для этого используется функция EntityValue. Он используется для выполнения запроса и возвращает результат в различных формах. Первым параметром укажем сущность (таблицу) Employees, которая недавно была зарегистрирована, а вторым параметром – список извлекаемых свойств (полей) сущности.



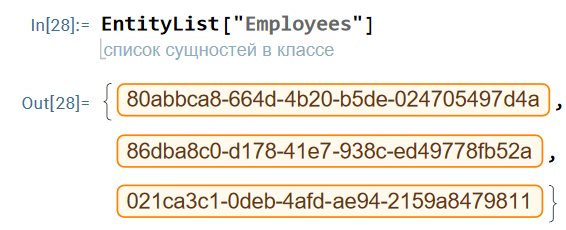
Добавим параметр “PropertyAssociation”, чтобы к возвращаемым значениям еще добавлялись названия свойств.



Получим результат в виде таблицы значений.



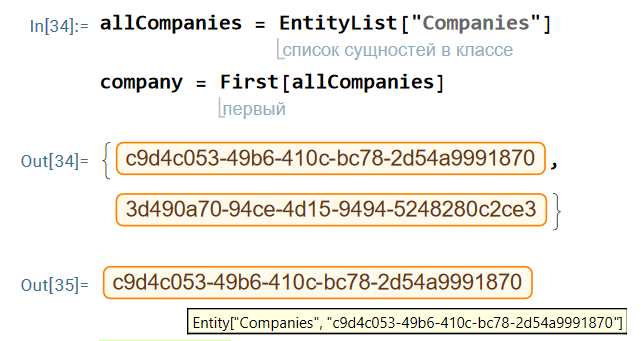
В некоторых случаях может понадобиться получить список сущностей, содержащихся в типе или классе. Это можно сделать через EntityList.



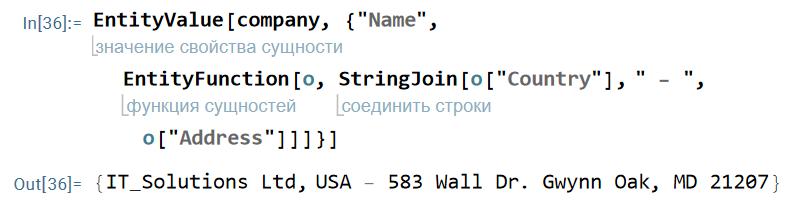
Кроме извлечения существующих свойств, можно также извлекать вычисляемые свойства – те свойства, которые вычисляются на лету. Такие свойства должны быть выражены через функцию EntityFunction. Пример: если вывести данные таблицы Companies, то получатся Название, Страна и Адрес. Логичнее было бы объединить Страну и Адрес в одно поле.



Entity Framework позволяет работать с одиночными сущностями. Так, например, получим из списка сущностей таблицы Companies первую сущность.



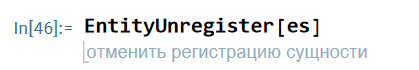
Затем можно вывести детальную информацию о ней.



При необходимости, выводимые поля можно отсортировать. Отсортируем поля таблицы Companies по их названию.



Если по какой-либо причине ранее зарегистрированные сущности нам будут не нужно, то их можно просто-напросто удалить. Для этого используется функция EntityUnregister.



После завершения работы с базой данных, желательно разорвать соединение с ней. Функция DatabaseDisconnect используется для это, а в качестве параметра передается ссылка соединения с ней, созданная в самом начале.

