TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Запросы на языке SQL. Начальный курс

Учебный материал

Лектор: Жанна Грачёва

Содержание

Основные понятия, классификация 3
Классификация СУБД 4
Реляционная база данных 5
Свойства реляционной таблицы 6
Связи в реляционной базе данных 7
Ключи 7
Схема базы данных Northwind 8
Задание 8
Язык SQL 9
Основные категории команд языка SQL: 9
Учебная база данных 11
Удаление избыточных данных 12
Оператор ORDER BY 13
Выборка с использованием фразы WHERE 13
Сравнение 14
Диапазон 14
Принадлежность множеству 14
Соответствие шаблону 15
Значение NULL 15
Ключевое слово TOP 15
Внутреннее соединение (INNER JOIN) 17
Внутреннее соединение по равенству (Equijoin) 18
Задание 19
Полезные ссылки 20

Основные понятия, классификация

Данные – зарегистрированная информация: представление фактов, понятий или инструкций в форме, приемлемой для общения, интерпретации, или обработки человеком или с помощью автоматических средств[1].

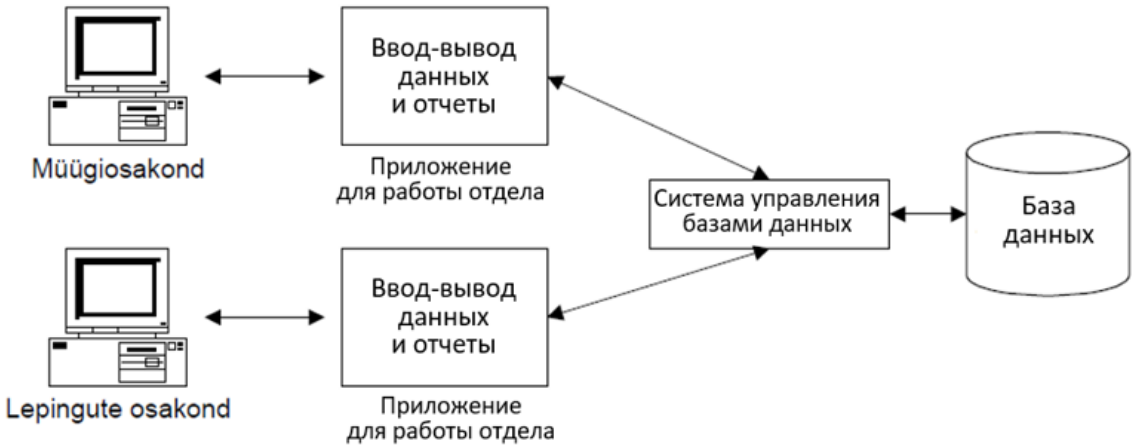
Данные – формы представления информации, с которыми имеют дело информационные системы и их пользователи[2].

Неупорядоченные данные – сырьё, необработанные сведения – основной источник информации.

Информацией можно назвать результат анализа и преобразования данных. В базе хранятся различные данные, а система управления базой может выдавать по определенному запросу требуемую информацию.

База данных (DataBase, Andmebaas) – это организованная структура, предназначенная для хранения информации. Обычно БД представляются в виде совокупности взаимосвязанных файлов или таблиц, предназначенных для решения конкретной задачи.

Система управления базами данных, СУБД (Database Management System) – комплекс программных средств, предназначенных для управления структурой базы данных, контроля доступом к данным, хранящимся в ней, и отвечающий на все корректно сформулированные запросы.



[1] ISO/IEC/IEEE 24765-2010 Systems and software engineering — Vocabulary:
a representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic

[2] ISO/IEC 10746-2:1996, Information technology — Open Distributed Processing — Reference Model: Foundations: *the representation forms of information dealt with by information systems and users*

Основные функции СУБД

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными).

Классификация СУБД

По модели данных

По **типу управляемой базы данных** СУБД:

- Иерархические
- Сетевые
- Реляционные (SQL)
- NoSQL
- Объектно-реляционные
- Объектно-ориентированные
- Документо- ориентированные
- ...

По архитектуре организации хранения данных

- локальные СУБД (все части локальной СУБД размещаются на одном компьютере)
- распределенные СУБД (части СУБД могут размещаться на двух и более компьютерах)

По способу доступа к БД

- Файл-серверные
- Клиент-серверные
- Встраиваемые

Рейтинг популярности согласно ресурсу DB-Engines

Оценки в рейтинге выставляются согласно 6 параметрам, среди которых популярность в поисковых системах, социальных сетях и на форумах, частота упоминание в резюме, количество вакансий.

DB-Engines Ranking

381 systems in ranking, December 2021

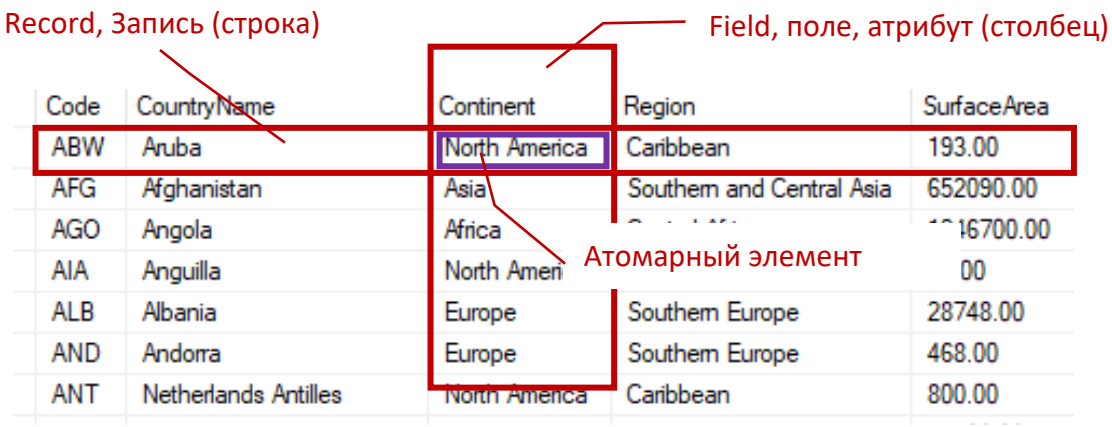
Rank Dec 2021	Rank Nov 2021	Rank Dec 2020	DBMS	Database Model	Score Dec 2021	Score Nov 2021	Score Dec 2020
1.	1.	1.	Oracle	Relational, Multi-model	1281.74	+9.01	-43.86
2.	2.	2.	MySQL	Relational, Multi-model	1206.04	-5.48	-49.41
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	954.02	-0.27	-84.07
4.	4.	4.	PostgreSQL	Relational, Multi-model	608.21	+10.94	+60.64
5.	5.	5.	MongoDB	Document, Multi-model	484.67	-2.67	+26.95
6.	6.	↑ 7.	Redis	Key-value, Multi-model	173.54	+2.04	+19.91
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	167.18	-0.34	+6.74
8.	8.	8.	Elasticsearch	Search engine, Multi-model	157.72	-1.36	+5.23
9.	9.	9.	SQLite	Relational	128.68	-1.12	+7.00
10.	↑ 11.	↑ 11.	Microsoft Access	Relational	125.99	+6.75	+9.25
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.20	-1.68	+0.36
12.	12.	12.	MariaDB	Relational, Multi-model	104.36	+2.17	+10.75

Реляционная база данных

Основоположником реляционных баз данных считается английский математик **Эдгар Кодд (Edgar Codd)**, который в 1970 году опубликовал в одном журнале статью «Реляционная модель данных для больших банков данных совместного использования». В статье Эдгар рассказывал про новую систему управления данными, основанную на математических принципах теории множеств и теории предикатов. Такая модель поддерживала точность и непротиворечивость данных, а также удобные извлечение и модификацию данных, со структурой, не зависящей от приложений и платформ. Предложеные в 1985 году **12 правил Кодда** (*Codd's 12 rules*) - 13 правил (в данном случае исчисление начинается с 0) заложили основы теории реляционных баз данных.

Реляционная база данных:

- Опирается на простой и в то же время мощный математический аппарат – реляционную алгебру (теорию множеств и математическую логику)
- Объекты-сущности информационной системы представляют плоскими **таблицами** данных.
- Таблица – двумерная структура, состоящая из **строк** и **столбцов**.



Code	CountryName	Continent	Region	SurfaceArea
ABW	Aruba	North America	Caribbean	193.00
AFG	Afghanistan	Asia	Southern and Central Asia	652090.00
AGO	Angola	Africa		16700.00
AIA	Anguilla	North Ameri		00
ALB	Albania	Europe	Southern Europe	28748.00
AND	Andorra	Europe	Southern Europe	468.00
ANT	Netherlands Antilles	North America	Caribbean	800.00

Свойства реляционной таблицы

- Каждая срока таблицы (кортеж) представляет собой **отдельную** сущность внутри набора сущностей.
- Каждый столбец представляет собой **атрибут**, у каждого столбца есть свое имя.
- На каждом пересечении строки и столбца имеется **единственное** значение.
- Каждая таблица должна иметь атрибут или несколько атрибутов, **уникально** идентифицирующих каждую строку.
- Все значения в таблице должны отображаться в одинаковом формате. Например, если атрибуту присваивается формат целого, то все значения в столбце, представляющем данный атрибут, должны быть целыми.
- Каждый столбец имеет определенный диапазон значений, называемый **доменом** атрибута (attribute domain).
- Порядок строк и столбцов для СУБД **не существенен**.

Пример. Таблица tCity

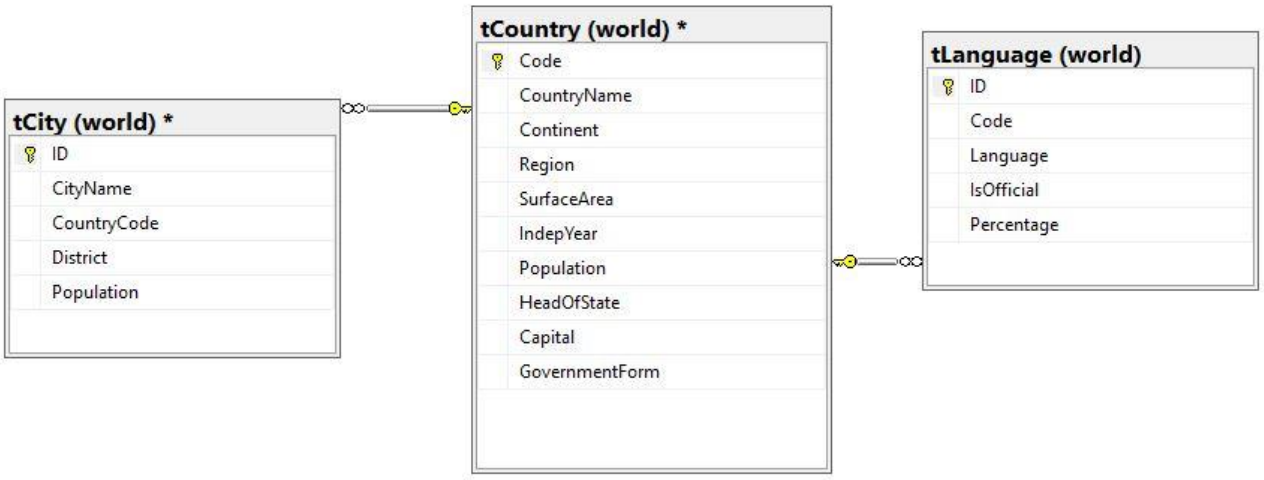
ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1760000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland ...	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323

Таблица tCountry

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeEx...	GNP	GNPOld	LocalName	GovernmentF...	HeadOfState	Capital
ABW	Aruba	North America	Caribbean	193,00	*NULL*	103000	78,4	828,00	793,00	Aruba	Nonmetropolit...	Beatrix	129
AFG	Afghanistan	Asia	Southern and C...	652090,00	1919	22720000	45,9	5976,00	*NULL*	Afganistan/...	Islamic Emirate	Mohammad O...	1
AGO	Angola	Africa	Central Africa	1246700,00	1975	12878000	38,3	6648,00	7984,00	Angola	Republic	José Eduardo d...	56
AIA	Anguilla	North America	Caribbean	96,00	*NULL*	8000	76,1	63,20	*NULL*	Anguilla	Dependent Terri...	Elisabeth II	62
ALB	Albania	Europe	Southern Europe	28748,00	1912	3401200	71,6	3205,00	2500,00	Shqip?ria	Republic	Rexhep Mejdani	34
AND	Andorra	Europe	Southern Europe	468,00	1278	78000	83,5	1630,00	*NULL*	Andorra	Parliamentary ...		55
ANT	Netherlands Ant...	North America	Caribbean	800,00	*NULL*	217000	74,7	1941,00	*NULL*	Nederlands...	Nonmetropolit...	Beatrix	33
ARE	United Arab Emi...	Asia	Middle East	83600,00	1971	2441000	74,1	37966,00	36846,00	Al-Imarat al...	Emirate Federat...	Zayid bin Sulta...	65
ARG	Argentina	South America	South America	2780400,00	1816	37032000	75,1	340238,00	323310,00	Argentina	Federal Republic	Fernando de la ...	69
ARM	Armenia	Asia	Middle East	29800,00	1991	3520000	66,4	1813,00	1627,00	Hajastan	Republic	Robert Kot?arjan	126

Связи в реляционной базе данных

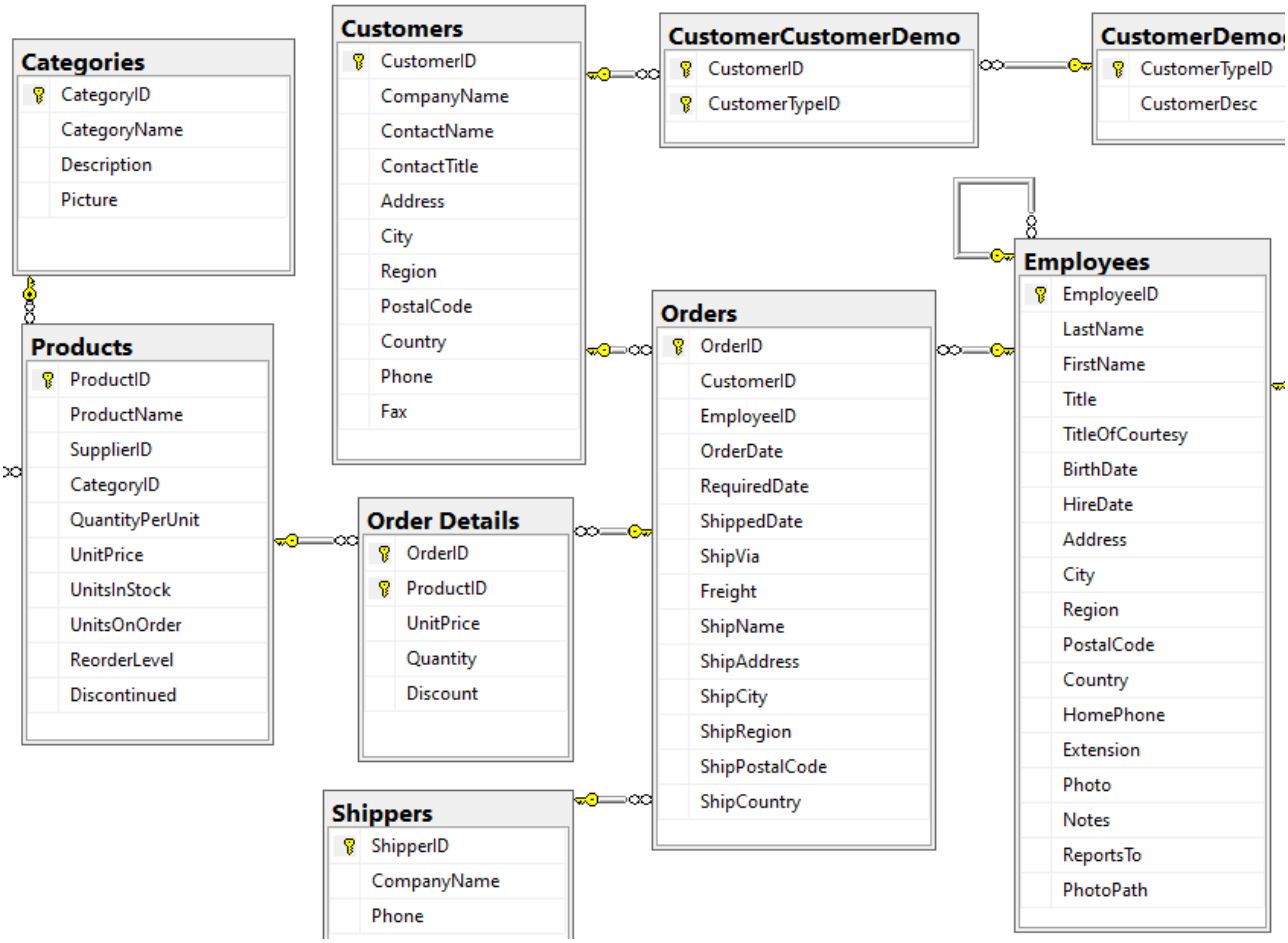
Отношение, связь (relationship) – ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.



Ключи

- Ключ состоит из одного или более атрибутов, определяющих другие атрибуты.
- Каждая таблица должна иметь **первичный ключ** (Primary Key) – это атрибут (или несколько атрибутов), **уникально** идентифицирующий данную сущность (строку).
- Роль ключа основана на концепции определяемости (Determination). В контексте таблицы базы данных выражение «A определяет B» означает, что зная величину A можно определить значение атрибута B.
- При определении первичного ключа в таблице говорят, что таблица проявляет **целостность на уровне сущности** (Entity Integrity). Для обеспечения целостности на уровне сущности в первичном ключе **недопустимы пустые значения** (Null).
- Ключ может состоять из более чем одного атрибута для определения функциональной зависимости – составной ключ (Composite Key). Любой атрибут, который является частью составного ключа, называется ключевым атрибутом (Key Attribute).
- **Внешний** ключ (Foreign Key) – атрибут, значение которого совпадает со значением первичного ключа в связанной таблице.

Схема базы данных Northwind



Задание 1

Изучить базу данных Northwind:

- Таблицы, поля, типы данных
- Связи между таблицами, типы связей
- Ключи (первичные и внешние ключи)
- Ограничения таблиц и атрибутов (NOT NULL, CHECK, UNIQUE), заничения по умолчанию DEFAULT

Язык SQL

На сегодняшний день язык SQL (Structured Query Language) является признанным стандартом языка баз данных, поддерживаемым всеми основными поставщиками реляционных СУБД.

Стандарт на язык SQL был выпущен Американским национальным институтом стандартов (ANSI) в 1986 г., а в 1987 г. Международная организация стандартов (ISO) приняла его в качестве международного. В настоящее время действует стандарт, принятый в 2003 году с небольшими модификациями, внесёнными позже (SQL:2016).

Под **реализацией** языка SQL понимается программный продукт SQL соответствующего производителя. Для расширения функциональных возможностей многие разработчики, придерживающиеся принятых стандартов, добавляют к стандартному языку SQL различные расширения. Все конкретные реализации языка несколько отличаются друг от друга. В интересах самих же производителей гарантировать, чтобы их реализация соответствовала современным стандартам ANSI в части переносимости и удобства работы пользователей. Тем не менее каждая реализация SQL содержит усовершенствования, отвечающие требованиям того или иного сервера баз данных. Эти усовершенствования или расширения языка SQL представляют собой дополнительные команды и опции, являющиеся добавлениями к стандартному пакету и доступные в данной конкретной реализации.

Основные категории команд языка SQL предназначены для выполнения различных функций, включая построение объектов базы данных и манипулирование ими, начальную загрузку данных в таблицы, обновление и удаление существующей информации, выполнение запросов к базе данных, управление доступом к ней и ее общее администрирование.

Основные категории команд языка SQL:

- DDL (Data Definition Language) – язык определения данных;
- DML (Data Manipulation Language) – язык манипулирования данными;
- DQL (Data Query Language) – язык запросов;
- DCL (Data Control Language) – язык управления данными;
- команды администрирования данных;
- команды управления транзакциями

Язык **DDL** позволяет создавать и изменять **структуру** объектов базы данных, например, создавать и удалять таблицы.

Основными командами языка DDL являются:

- CREATE TABLE, ALTER TABLE, DROP TABLE
- CREATE INDEX, ALTER INDEX, DROP INDEX

Язык **DML** используется для манипулирования информацией внутри объектов реляционной базы данных посредством трех основных команд:

- INSERT
- UPDATE
- DELETE

Язык запросов **DQL** наиболее известен пользователям реляционной базы данных, несмотря на то, что он включает одну команду **SELECT**. Эта команда вместе со своими многочисленными опциями и предложениями используется для формирования запросов к реляционной базе данных.

Команды **управления данными** – GRANT, REVOKE

- позволяют управлять доступом к информации, находящейся внутри базы данных;
- как правило, используются для создания объектов, связанных с доступом к данным;
- служат для контроля над распределением привилегий между пользователями.

Команды **администрирования данных**

- осуществляют контроль за выполняемыми действиями;
- анализируют операции базы данных;
- также могут оказаться полезными при анализе производительности системы.

Не следует путать администрирование данных с администрированием базы данных, которое представляет собой общее управление базой данных и подразумевает использование команд всех уровней.

Составление SQL команд

Оператор SQL состоит из зарезервированных слов, а также из слов, определяемых пользователем. Зарезервированные слова являются постоянной частью языка SQL и имеют фиксированное значение. Их следует записывать в точности так, как это установлено, нельзя разбивать на части для переноса с одной строки на другую. Слова, определяемые пользователем, задаются им самим (в соответствии с синтаксическими правилами) и представляют собой **идентификаторы** или имена различных объектов базы данных.

Идентификаторы языка SQL предназначены для обозначения объектов в базе данных и являются именами таблиц, представлений, столбцов и других объектов базы данных. Стандарт SQL задает набор символов, который используется по умолчанию:

- строчные и прописные буквы латинского алфавита (A-Z, a-z), цифры (0-9) и символ подчеркивания (_).

На формат идентификатора накладываются следующие ограничения:

- идентификатор может иметь длину до 128 символов;
- идентификатор должен начинаться с буквы;
- идентификатор не может содержать пробелы.

Большинство компонентов языка **не чувствительны** к регистру.

Учебная база данных

Все примеры и задания выполняются в базе данных World.

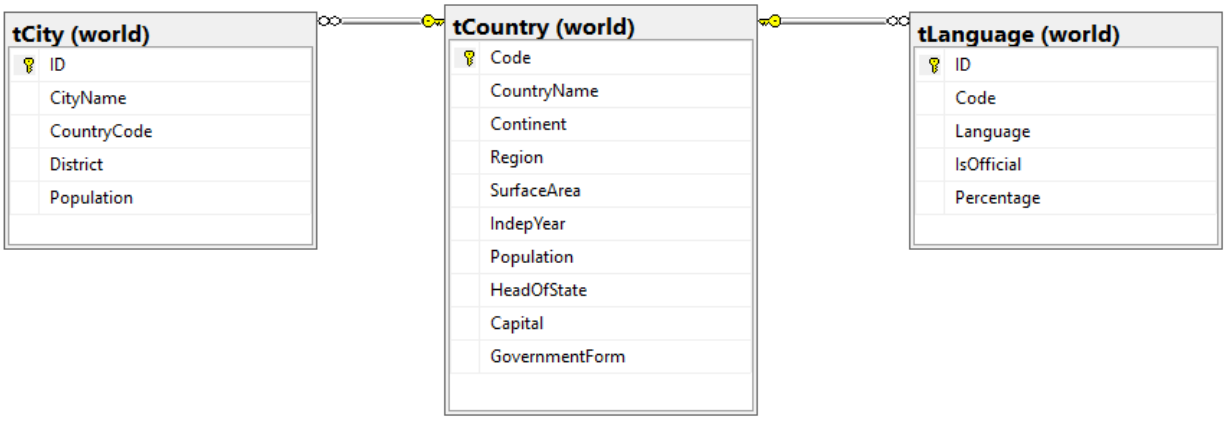


Таблица **tCountry** этой базы данных содержит информацию о странах мира: код страны, её название, континент и регион, в котором страна находится, занимаемая площадь, год обретения страной независимости, население страны, форма правления, глава страны и код столицы.

Таблица **tCity** этой базы данных содержит информацию о городах мира: код города (ID), его название (cityName), страна (countryName), континент (continent), регион (region) и область страны (district), в которой находится город, число его жителей (population).

В таблице **tLanguage** хранится информация о том, какой процент населения стран на каких языках разговаривает: код страны (Code), язык (Language), является ли язык официальным (IsOfficial), процент населения страны, разговоривающего на этом языке (Percentage).

Оператор SELECT

Оператор **SELECT** – один из наиболее важных и самых распространенных операторов SQL. Он позволяет производить выборки данных из таблиц и преобразовывать к нужному виду полученные результаты.

Используя SELECT, вы можете указывать:

- Какую таблицу/таблицы следует использовать
- Какие столбцы (поля) таблицы отображать
- Как назвать столбцы
- Какие строки извлекать из таблицы
- Как упорядочить строки

Обработка элементов оператора SELECT выполняется в следующей последовательности:

SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
FROM – определяются имена используемых таблиц;
WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;
ORDER BY – определяется упорядоченность результатов выполнения операторов.

Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения **SELECT** и **FROM** являются обязательными, все остальные могут быть не использоваться.

Фраза **SELECT** определяет поля (столбцы), которые будут входить в результат выполнения запроса. В списке они разделяются запятыми и приводятся в такой очередности, в какой должны быть представлены в результате запроса.

Необязательный параметр псевдонима (в нашем примере - Riik) – это сокращение, устанавливаемое для имени поля таблицы.

```
SELECT CountryName AS Riik
       FROM tCountry;
```

Символом * можно выбрать все поля.

```
SELECT * FROM tCountry;
```

В запросах могут создаваться вычисляемые поля с использованием операторов и функций:

```
SELECT CountryName AS Riik, Round(Population/1000,1) AS [Tuh elanikud]
FROM tCountry;
```

Псевдоним, так же как и любое имя поля, содержащее пробелы, следует поместить в квадратные скобки.

Задание

Создать SQL запросы для вывода следующей информации:

1. Названия страны, её площадь и количество жителей. Полям результирующего запроса дать соответствующие псевдонимы.
2. Все записи таблиц tCity и tCountry.
3. Названия всех континентов, представленных в таблице tCountry.

Удаление избыточных данных

При получении списка континентов, представленных в таблице tCountry с помощью следующего запроса получаем столько строк, сколько записей в таблице.

```
SELECT Continent
FROM tCountry;
```

Для устранения повторяющихся значений используют аргумент DISTINCT (ОТЛИЧИЕ):

```
SELECT DISTINCT Continent
FROM tCountry;
```

Задание

Создать SQL запросы для вывода следующей информации:

1. Названия всех континентов, представленных в таблице tCountry.
2. Список стран, города которых представлены в таблице tCity.
3. Список форм правления странами, представленными в таблице tCountry.

Оператор ORDER BY

Фраза **ORDER BY** позволяет упорядочить вывод запроса согласно значениям в того или иного поля (столбца). Для каждого столбца можно определить возрастание (ASC) или убывание (DESC). По умолчанию установлено возрастание.

Данный оператор используется в конце запроса.

Пример 1 – сортировка данных в запросе по полю Population – населению страны в порядке убывания

```
SELECT CountryName, Population
FROM tCountry
ORDER BY Population DESC;
```

Пример 2 – сортировка данных сначала в алфавитном порядке по континенту, а потом в порядке убывания населения страны

```
SELECT CountryName, Population
FROM tCountry
       ORDER BY Continent, Population DESC;
```

Задание

Создать SQL запросы для вывода следующей информации:

1. Список стран, отсортированный в алфавитном порядке
2. Список стран, упорядоченный по занимаемой ими площади
3. Список стран, расположенных в алфавитном порядке регионов, в которых они находятся и для каждого региона - по убыванию населения страны.

Выборка с использованием фразы WHERE

Раздел **WHERE** - не обязательный раздел в SQL предложениях. WHERE - фраза команды SELECT, которая определяет условие, которому должны удовлетворять все строки, используемые для формирования результирующего набора. Предикат содержит одно или несколько выражений, выполняющих сравнения. Когда предложение WHERE представлено, программа базы данных просматривает всю таблицу по одной строке и исследует каждую строку чтобы определить верно ли утверждение, записанное в WHERE. Команда извлекает только те строки из таблицы для которой такое утверждение верно (TRUE).

Существует пять основных типов условий поиска (или предикатов):

- **Сравнение**: сравниваются результаты вычисления одного выражения с результатами вычисления другого.
- **Диапазон**: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений.
- **Принадлежность множеству**: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.
- **Соответствие шаблону**: проверяется, отвечает ли некоторое строковое значение заданному шаблону.
- **Значение NULL**: проверяется, содержит ли данный столбец NULL (пустое значение).

Сравнение

В синтаксисе фразы WHERE для отбора нужных строк таблицы можно использовать операторы сравнения = (равно), <> или != (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), которые могут предваряться оператором NOT, создавая, например, отношения "не меньше" и "не больше". Возможность использования нескольких условий обеспечивает использование соединенных логических операторов AND и OR.

Пример 3. Показать страны, получившие независимость до нашей эры.

```
SELECT CountryName, Continent, IndepYear
FROM tCountry
	WHERE IndepYear < 0;
```

Пример 4. Показать страны, получившие независимость в 20-м веке.

```
SELECT CountryName, Continent, IndepYear
FROM tCountry
	WHERE IndepYear >= 1900 AND IndepYear < 2000;
```

Вычисление выражения в условиях выполняется по следующим правилам:

- Выражение вычисляется слева направо
- Первыми вычисляются подвыражения в скобках
- Операторы NOT выполняются до выполнения операторов AND и OR
- Операторы AND выполняются до выполнения операторов OR

Для устранения любой возможной неоднозначности рекомендуется использовать скобки.

Диапазон

Оператор **BETWEEN** используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и максимальным значениями. При этом указанные значения включаются в условие поиска.

Пример 5. Показать страны, получившие независимость в 20-м веке.

```
SELECT CountryName, Continent, IndepYear
FROM tCountry
	WHERE IndepYear BETWEEN 1900 AND 1999;
```

Принадлежность множеству

Оператор **IN** используется для сравнения некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. При помощи оператора IN может быть достигнут тот же результат, что и в случае применения оператора OR, однако оператор IN выполняется быстрее.

Пример 6. Показать страны Европы и Азии.

```
SELECT CountryName, Continent
FROM tCountry
	WHERE Continent IN ('Europe','Asia');
```

NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке.

Соответствие шаблону

С помощью оператора **LIKE** можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей:

% - любое количество произвольных символов;
_ - заменяет один символ;
[ ] - вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

Пример 7. Показать страны, название которых начинается с буквы A

```
SELECT CountryName
FROM tCountry
	WHERE CountryName LIKE 'A%';
```

Пример 8. Показать страны, в названиях которых есть слог "new"

```
SELECT CountryName
FROM tCountry
	WHERE CountryName LIKE '%new%';
```

Пример 9. Показать страны, название которых начинается с букв X, Y или Z (от X до Z)

```
SELECT CountryName
FROM tCountry
	WHERE CountryName LIKE '[X-Z]%';
```

Значение NULL

NULL – специальное значение, указывающее на отсутствие любого значения. NULL – это не то же самое, что знак пробела или ноль. NULL отличается и от строки нулевой длины (пустой строки). Для сравнения текущего значения со значением NULL используется оператор **IS NULL**.

Пример 10. Показать страны, у которых нет столицы (столица которых не определена)

```
SELECT CountryName, Continent
FROM tCountry
	WHERE Capital IS NULL;
```

Ключевое слово TOP

Ключевое слово **TOP** может быть использовано для возврата первых ("верхних") *n* строк или первых *n* процентов результирующего запроса. Обычно TOP используется совместно с оператором сортировки ORDER BY, позволяя получить первых *n* записей с наибольшими или наименьшими значениями того или иного поля.

Пример 11. Показать наибольшую по площади страну

```
SELECT TOP 1 CountryName, SurfaceArea
FROM tCountry
	ORDER BY SurfaceArea DESC;
```

Пример 12. Показать первую пятерку стран наименьшей площади

```
SELECT TOP 5 CountryName, Continent, SurfaceArea
FROM tCountry
	ORDER BY SurfaceArea;
```

Пример 13. Показать 1% от всего списка наименьших по площади стран

```
SELECT TOP 1 PERCENT CountryName, SurfaceArea
FROM tCountry
	ORDER BY SurfaceArea DESC;
```

Параметр в MS SQL Server **WITH TIES** позволяет получить большее число записей при одинаковых значениях фильтра.

Группировка данных

Фраза GROUP BY

Фраза GROUP BY позволяет определять подмножество значений и применять функцию агрегата к подмножеству, то есть инициирует перекомпоновку указанной во FROM таблицы по группам, каждая из которых имеет одинаковые значения в столбце, указанном в GROUP BY.

К функциям агрегирования относятся такие функции как:

```
AVG(column) - среднее значение группы
SUM(column) - сумма значений группы
MAX(column) - наибольшее значение из группы
MIN(column) - наименьшее значение из группы
COUNT(*) -    количество всех значений в группе
COUNT(column) - количество значений в группе за исключением значения NULL
```

Например, для определения числа городов каждой страны, произведем группировку по значению поля (название страны) и для каждой полученной группы найдем функцию COUNT – количество.

```
SELECT CountryName, COUNT(*) AS [Linnade arv]
FROM tCity
	GROUP BY CountryName;
```

Можно добавить в данный запрос информацию о континенте. В этом случае необходимо континент добавить как в предложение SELECT так и в группировку (фразу GROUP BY):

```
SELECT CountryName, Continent, COUNT(*) AS [Linnade arv]
FROM tCity
	GROUP BY CountryName, Continent;
```

Выборка данных

Фраза HAVING

Фраза HAVING играет такую же роль для групп, что и фраза WHERE для строк: она используется для исключения групп, точно так же, как WHERE используется для исключения строк. Эта фраза включается в предложение лишь при наличии фразы GROUP BY, а выражение в HAVING должно принимать единственное значение для группы.

Когда GROUP BY не используется, предложение HAVING работает так же, как и предложение WHERE.

Так, если нужно получить информацию только о странах с большим числом городов, например, больше 100, то запрос будет выглядеть следующим образом:

Пример 14

```
SELECT countryName, COUNT(*) AS [Linnade arv]
FROM tCity
GROUP BY countryName, continent
       HAVING COUNT(*) > 100;
```

Задание

Создать SQL запросы для вывода следующей информации:

1. Количество стран по континентам, у которых неизвестен год обретения назависимости
2. Количество стран возникших на карте мира в каждом веке.

Выборка данных из нескольких таблиц

Имена и псевдонимы таблиц

Полное имя столбца таблицы состоит из имени таблицы и имени столбца, разделенных точкой. Так, столбцы **Code, Continent, CountryName** таблицы **tCountry** имеют следующие полные имена: **tCountry.Code, tCountry.Continent, tCountry.CountryName**.

При составлении запроса на выборку данных из одной таблицы мы могли опускать имена таблиц. Если же в запросе мы должны обращаться к полям **нескольких** различных таблиц, то следует использовать полные имена полей, особенно в тех случаях, когда названия полей совпадают. Кроме того, можно использовать не полные имена таблиц, а их алиасы или псевдонимы.

Для задания псевдонимов используется служебное слово, которое в некоторых реализациях SQL можно опускать

tCountry AS C или **tCountry C**

Теперь к столбцам Code, CountryName, Continent таблицы **tCountry** с псевдонимом C можно обращаться **C.Code, C. CountryName, C.Continent**

```
SELECT C.Code, C.CountryName, C.Continent
FROM world.tCountry C;
```

NB! Данные псевдонимы действуют только в пределах данного SQL запроса.

Внутреннее соединение (INNER JOIN)

Внутренние соединения – наиболее часто встречающийся тип соединений. Они служат для получения только тех строк, для которых существует соответствие записей главной таблицы и присоединяемой по значениям в связанных полях, то есть условие соединения (join_condition) должно выполняться всегда.

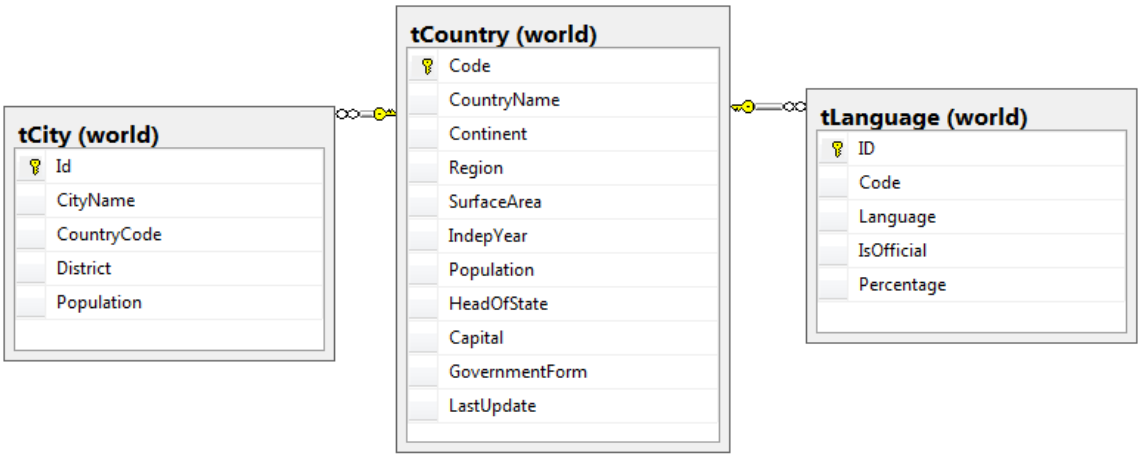
Внутреннее соединение по равенству (Equijoin)

При выполнении запроса с внутренним соединением в операцию включаются только строки, имеющие одинаковые значения в обеих связанных таблицах.

```
SELECT Таблица1.Поле1, Таблица2.Поле2, ...
FROM Таблица1 INNER JOIN Таблица2
ON Таблица1.Поле1 = Таблица2.Поле2;
```

Также можно использовать соединение с синтаксисом, когда условие соединения (join_condition) прописывается во фразе WHERE:

```
SELECT Таблица1.Поле1, Таблица2.Поле2, ...
FROM Таблица1, Таблица2
WHERE Таблица1.Поле1 = Таблица2.Поле2;
```



Так, если в таблице **tCountry** есть поле **Capital**, хранящее код города, являющегося столицей страны, то по этому коду из таблицы **tCity** (поле ID) можем получить название города и всю информацию о нем. Для этого в запросе должно быть выполнено условие соединения **tCountry.Capital = tCity.ID**:

```
SELECT world.tCountry.CountryName AS Riik,
       world.tCity.CityName AS Pealinn
FROM world.tCountry, world.tCity
WHERE world.tCountry.Capital = world.tCity.ID;
```

или

```
SELECT world.tCountry.CountryName AS Riik,
       world.tCity.CityName AS Pealinn
FROM world.tCountry INNER JOIN world.tCity
       ON world.tCountry.Capital = world.tCity.ID;
```

Тот же запрос с использованием псевдонимов даст тот же результат:

```
SELECT C.CountryName AS Riik, Ci.CityName AS Pealinn
FROM world.tCountry AS C INNER JOIN world.tCity AS Ci
       ON C.Capital = Ci.ID;
```

Задание

Создать SQL запросы для вывода следующей информации:

1. Названия стран континента Европа, их столицы, количество населения проживающего в столице, отсортированные в порядке убывания населения;
2. Самый маленький по площади континент;
3. Страны, в которых название страны и название столицы совпадают;
4. Число городов в базе данных по каждой стране (название страны, количество городов (отсортированы по убыванию).

Соединение более двух таблиц

При соединении в запросах трех и более таблиц действуют точно те же принципы, что и при соединении двух таблиц. Когда в одном операторе совмещаются несколько соединений, join условия обычно связываются логической операцией AND.

Например, если мы хотим получить информацию о стране, столице и официальном языке этой страны, то придется получать данные из таблиц:

tCountry (CountryName – название страны),
tCity (CityName – название города, являющегося столицей)
и **tLanguage** (Language – язык, который является официальным – поле IsOfficial=True)

```
SELECT C.CountryName AS Riik, Ci.CityName AS Pealinn, L.Language AS Keel
FROM (world.tCountry AS C INNER JOIN world.tCity AS Ci ON C.Capital = Ci.ID)
        INNER JOIN world.tLanguage AS L ON C.Code = L.Code
WHERE L.IsOfficial = 'True'
```

или

```
SELECT C.CountryName AS Riik, Ci.CityName AS Pealinn, L.Language AS Keel
FROM world.tCountry AS C, world.tCity AS Ci, world.tLanguage AS L
WHERE  C.Capital = Ci.ID AND C.Code = L.Code AND L.IsOfficial = 'True'
```

Riik	Pealinn	Keel
Afghanistan	Kabul	Dari
Afghanistan	Kabul	Pashto
Netherlands	Amsterdam	Dutch
Netherlands Antilles	Willemstad	Dutch
Netherlands Antilles	Willemstad	Papiamento
Albania	Tirana	Albaniana
Algeria	Alger	Arabic
American Samoa	Fagatogo	English
American Samoa	Fagatogo	Samoan
Andorra	Andorra la Vella	Catalan

Задание

Вывести информацию о странах – код, название, местоположение, форма правления, столица, количество официальных языков.

Полезные ссылки

- SQL Server Tutorial https://www.sqlservertutorial.net/
- Use scalar functions - Learn | Microsoft Docs
- Введение в MS SQL Server и T-SQL https://metanit.com/sql/sqlserver/1.1.php
- Встроенные функции MS SQL Server https://metanit.com/sql/sqlserver/8.1.php