**Задания 1-6 выполняем на MS SQL. Задание 7 на postgresql**

Использовал SQL Server 2019 Express, PostgreSQL 10

Collation = Cyrillic\_General\_CI\_AS

1. **Создайте в БД объекты, пропишите скрипты на создание и ключи, а так же выполните запросы:**

1.1) Посчитать общую сумму всех долгов по договорам в те месяцы, где между последним и 15 днем 16 дней.

Если читать буквально, то таких месяцев нет, т.к. между 15 и 31 (не включая эти даты) будет только 15

Предположу, что все таки речь идет о месяцах, в которых 31 день (31-15=16)

select DateReport,SUM(Dolg) as Dolg from dbo.Balance where DAY(DateReport)-15=16 GROUP BY DateReport

1.2) Посчитать общую сумму по всем договорам, сформированную на конец года.

Вариант 1 – из расчета, что по всем договорам суммы долга на последнюю дату года уже посчитаны:

select SUM(Dolg) as Dolg from dbo.Balance where DateReport='2014-12-31'

Вариант 2 – на случай, если НЕ по всем договорам суммы долга на последнюю дату года посчитаны:

select SUM(a.Dolg) as Dolg from dbo.Balance a where a.DateReport = (select max(DateReport) from Balance b where b.Dogovor=a.Dogovor AND b.DateReport<='2014-12-31')

1.3) Показать минимальную ставку резерва в году; месяц, в котором была эта ставка; сумму сформированного резерва в этот месяц в одном окне результатов.

Не очень понимаю, что означает в таблице Statistica поле Days, а также единица измерения значений в поле Stavka.

Предположу, что Days – это максимальное кол-во дней в месяце, на которое нужно посчитать резерв, причем за каждый день берется процент в размере Stavka от суммы Dolg, т.е. резерв на каждый месяц для каждого договора считается по формуле:

Balance.Dolg \* Statistica.Stavka /100 \* Statistica.Days

Также предположу, что в таблице Statistica есть ставки для всех месяцев.

select s.Stavka as MinStavkaInYear, b.DateReport, SUM(b.Dolg \* s.Days \* s.Stavka / 100) as Reserv

from dbo.Balance b, dbo.Statistica s

where b.DateReport=s.DateReport

and b.DateReport between '2014-01-01' and '2014-12-31'

and s.Stavka=(select MIN(ss.Stavka) from dbo.Statistica ss where ss.DateReport between '2014-01-01' and '2014-12-31')

GROUP BY b.DateReport, s.Stavka

1.4) Показать, сколько было сформировано резервов по договору с на конец октября 2014 года, если доля резервов по данному продукту относительно общей суммы резервов составляет 14%; ставку резервирования по данному договору на конец октября 2014. Показать в одном Окне результатов.

**Balance (ID , DateReport, Dogovor, Dolg)**

Скрипт на создание таблицы -  [Balance.sql](https://github.com/sprokushev/RTMIS/blob/master/Balance.sql)

Поскольку ID не уникально, предположил, что уникальные ключи:

* ID, Dogovor
* Dogovor, DateReport

Скрипт для загрузки тестовых данных - loadBalance.sql

1, '20140131', 'a', 100000

1, '20140131', 'b', 100000

2, '20140228', 'a', 100000-3000

2, '20140228', 'b', 100000-1000

3, '20140331', 'a', 100000-3000-3000

3, '20140331', 'b', 100000-1000-1000

4, '20140430', 'a', 100000-3000-3000-3000

4, '20140430', 'b', 100000-1000-1000-1000

5, '20140531', 'a', 100000-3000-3000-3000-3000

5, '20140531', 'b', 100000-1000-1000-1000-1000

6, '20140630', 'a', 100000-3000-3000-3000-3000-3000

6, '20140630', 'b', 100000-1000-1000-1000-1000-1000

6, '20140630', 'c', 200000

7, '20140731', 'a', 100000-3000-3000-3000-3000-3000-3000

7, '20140731', 'b', 100000-1000-1000-1000-1000-1000-1000

7, '20140731', 'c', 200000-10000

8, '20140831', 'a', 100000-3000-3000-3000-3000-3000-3000-3000

8, '20140831', 'b', 100000-1000-1000-1000-1000-1000-1000-1000

8, '20140831', 'c', 200000-10000-10000

9, '20140930', 'a', 100000-3000-3000-3000-3000-3000-3000-3000-3000

9, '20140930', 'b', 100000-1000-1000-1000-1000-1000-1000-1000-1000

9, '20140930', 'c', 200000-10000-10000-10000

10, '20141031', 'a', 100000-3000-3000-3000-3000-3000-3000-3000-3000-3000

10, '20141031', 'b', 100000-1000-1000-1000-1000-1000-1000-1000-1000-1000

10, '20141031', 'c', 200000-10000-10000-10000-10000

11, '20141130', 'a', 100000-3000-3000-3000-3000-3000-3000-3000-3000-3000-3000

11, '20141130', 'b', 100000-1000-1000-1000-1000-1000-1000-1000-1000-1000-1000

11, '20141130', 'c', 200000-10000-10000-10000-10000-10000

12, '20141231', 'a', 100000-3000-3000-3000-3000-3000-3000-3000-3000-3000-3000-3000

12, '20141231', 'b', 100000-1000-1000-1000-1000-1000-1000-1000-1000-1000-1000-1000

12, '20141231', 'c', 200000-10000-10000-10000-10000-10000-10000

**Statistica (ID,DateReport,Days, Stavka)**

Скрипт на создание таблицы -  [Statistica.sql](https://github.com/sprokushev/RTMIS/blob/master/Balance.sql)

Скрипт для загрузки тестовых данных - loadStatistica.sql

Есть потеря точности начиная с 16-го знака после запятой. Влияния на итоговые результаты я не увидел.

1, '20140131', 16, 0.035

2, '20140228', 13, 0.035\*1.15

3, '20140331', 16, 0.035\*1.15\*1.15

4, '20140430', 15, 0.035\*1.15\*1.15\*1.15

5, '20140531', 16, 0.035\*1.15\*1.15\*1.15\*1.15

6, '20140630', 15, 0.035\*1.15\*1.15\*1.15\*1.15/1.05

7, '20140731', 16, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05

8, '20140831', 16, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05/1.05

9, '20140930', 15, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05/1.05/1.05

10, '20141031', 16, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05/1.05/1.05/1.05

11, '20141130', 15, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05/1.05/1.05/1.05/1.05

12, '20141231', 16, 0.035\*1.15\*1.15\*1.15\*1.15/1.05/1.05/1.05/1.05/1.05/1.05/1.05

2. Каким образом вы отслеживаете изменения, производимые на БД разработки для последующего переноса их в рабочее окружение, какие инструменты используете?

Visual Studio 2019 +  Data Storage and processing Workload + Git

Данное задание выполнил с использованием данных инструментов, результат:

<https://github.com/sprokushev/RTMIS.git>

Создал 2 базы: TEST, PROD

Разработку делал в TEST, затем модификации БД после каждого задания переносил в PROD (за исключением загрузки тестовых данных), все изменения отражал в репозитории Git

1. Опишите алгоритм для произвольной СУБД по «дефрагментации» первичных ключей. Т.е. имеется последовательность первичных ключей вида 1, 2, 7, 8, 10. Необходимо, чтобы была последовательность вида 1, 2, 3, 4, 5.
2. Ограничить доступ к БД со стороны пользователей и приложений, с целью предотвращения модификации данных на период»дефрагметнации»
3. Выполнить бакап БД
4. Для каждой «дефрагментируемой» таблицы:

3.1 Определяем перечень таблиц в БД, у которых есть Foreign Keys на «дефрагментируемую» таблицу. Временно отключаем (disable) или удаляем указанные внешние ключи.

3.2 Осуществляем выборку текущих значений первичного ключа, упорядочиваем по возрастанию

select id from <table> order by id asc

3.3 В отдельной переменной COUNTER ведем счетчик с новыми значениями ключа (начиная с 1)

3.4 Перебираем «старые» значения первичного ключа, если обнаруживаем расхождение с COUNTER - осуществляем update в самой таблице и во всех таблицах имеющих foreign key на нее:

BEGIN TRANSACTION

UPDATE <table> SET ID=COUNTER WHERE ID=<old ID>

UPDATE <table\_has\_fk> SET key\_id= COUNTER where key\_id=<old ID>

COUNTER= COUNTER+1

COMMIT TRANSACTION

3.5. Включаем (enable) или создаем заново отключенные/удаленные ранее constraints

4 Делаем rebuild индексов и сбор статистики для всех таблиц, которые были затронуты «дефрагментацией»

**4.** Предложите варианты, с помощью которых можно распараллелить нагрузку (чтение и запись) на высоконагруженную БД (MSSQL) без кардинальной переделки приложения, – т.е. шардинг, например, не подойдет.

На уровне дисковой подсистемы сервера – добавить database files в файловую группу БД и разнести файлы по разным дискам

Других вариантов для MS SQL по балансировке нагрузки я не предложу.

Для Oracle есть возможность развертывания в режиме Real Application Cluster с балансировкой нагрузки, но на сколько я знаю, у MS SQL есть только возможность распараллелить чтение (read-only routing for an Always On availability group).

**5.** Реализуйте структуру (таблицы, хранимые процедуры и т.д.), для произвольной СУБД (MS SQL), с помощью которой в БД можно хранить древовидные данные, не используя специфические типы (например, Hierarchyid из MSSQL). Основное требование – структура должна поддерживать возможность выборки: всех дочерних элементов определенного узла, быструю выборку всех подчиненных элементов для произвольного узла самого верхнего уровня, быструю выборку всех подчиненных элементов всех уровней для произвольного узла. Все должно работать без рекурсии.

Взял таблицу dbo.person из следующей задачи, добавил father\_id

Создал таблицу closure\_person – в ней хранится информация обо всех предках/потомках и глубина относительно родителя

father\_id=0 – корневой элемент иерархии

Процедура AddPerson – добавить строку в person

Процедура DelPerson – удалить строку в person

Скрипт – loadPerson.sql – загрузка тестовых данных

выбрать всех детей для id=3:

select \* from dbo.GetChilds(3) order by birthday;

выбрать всех потомков (детей и их детей) для id=3

select \* from dbo.GetAllChilds(1) order by depth,birthday;

выбрать цепочку предков (родитель, его родитель, и так далее)

Выборка всех дочерних элементов определенного узла:

Выборка всех подчиненных элементов всех уровней для произвольного узла:

Выборка всех родительских узлов для произвольного узла:

выбрать цепочку предков элемента

6. Создайте таблицу **dbo.person,** в которой будет храниться список людей с полями:

-Ид человека

-Фамилия

-Имя

-Отчество

-Дата рождения

-Пол

Скрипт person.sql

И таблицу в которой будут храниться адреса человека **dbo.Address** с полями:

-Ид адреса

-Ид человека

-Страна

-Город

-Улица

-Дом

-Квартира

**Пропишите ключи.**

**7.**

**7.**1)Создать таблицу с наименованиями стран и их столиц в схеме dbo.

7.1.1) В таблице должны присутствовать поля с уникальным идентификатором записи, значениями по умолчанию и без.

7.2) Создать функции по добавлению и удалению данных.

7.3) Написать скрипт по добавлению/удалению данных в созданную таблицу.

7.4) Создать функцию для приведения текстовых данных в любой таблице к верхнему или нижнему регистру (опционально).

7.5) Создать обычное и материализованное представление для созданной таблицы.

7.6) Провести выборку oid, наименования схем и наименование созданных объектов.