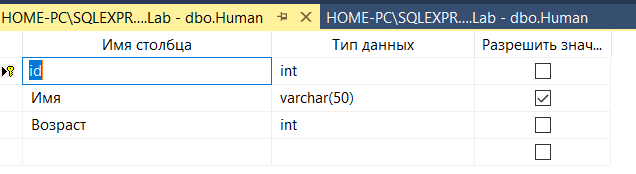
**Лабораторная работа 2. Транзакции и блокировки. Уровни изолированности транзакций. Распределённые транзакции.**

Предметная область БД, на которой необходимо производить упражнения, может быть любой. Рекомендуется использовать БД, которая была разработана в ходе лабораторного практикума по дисциплине «Управление данными».

**Задание:**

1 Наглядно продемонстрировать влияние параметра xact\_abort.

Создадим таблицу Human



set xact\_abort off

begin transaction;

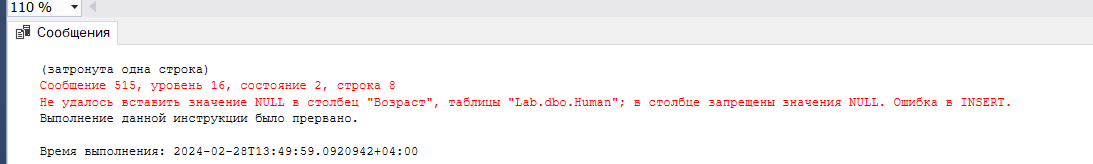
insert into Human (Имя, Возраст)

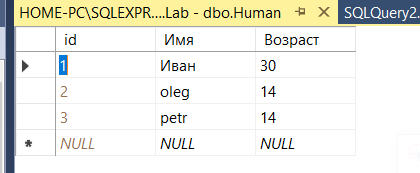
values ('petr',14)

insert into Human (Имя, Возраст)

values ('ann',null)

commit transaction;





В таблицу добавилась информация только про Петра.

При xact\_abort on мы видим следущее:

set xact\_abort on

begin transaction;

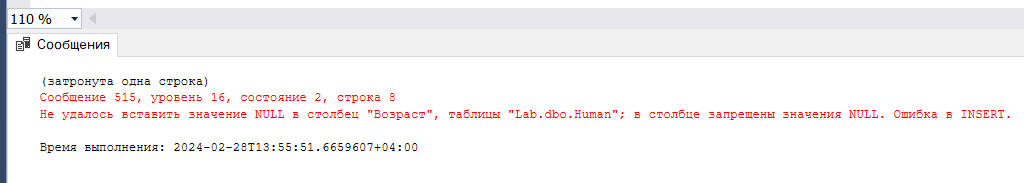
insert into Human (Имя, Возраст)

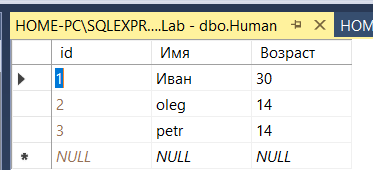
values ('dasha',80)

insert into Human (Имя, Возраст)

values ('masha',null)

commit transaction;





В таблицу ничего не добавилось

Если оба запроса правильные:

set xact\_abort on

begin transaction;

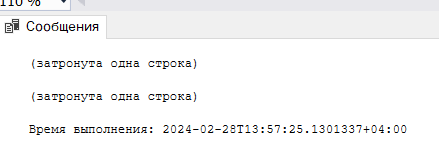
insert into Human (Имя, Возраст)

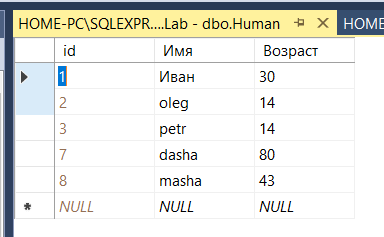
values ('dasha',80)

insert into Human (Имя, Возраст)

values ('masha',43)

commit transaction;





2 Наглядно продемонстрировать явные, неявные транзакции в обоих режимах работы сервера (режим неявного подтверждения и автоматического подтверждения).

1.Явная транзакция в режиме автоматического подтверждения.

SET IMPLICIT\_TRANSACTIONS OFF;

PRINT @@TRANCOUNT

BEGIN TRANSACTION

PRINT @@TRANCOUNT

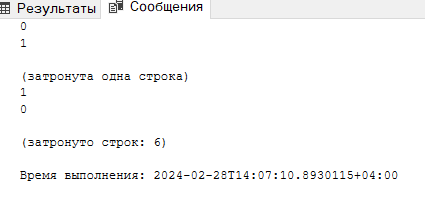
INSERT INTO Human (Имя,Возраст) VALUES ('Ксюша', 10)

PRINT @@TRANCOUNT

COMMIT TRANSACTION

PRINT @@TRANCOUNT

SELECT \* FROM Human;



Мы видим, что значение @@TRANCOUT равно единице внутри блока явной транзакции, а после COMMIT = 0, значит транзакция завершилась.

2. Явная транзакция в режиме неявного подтверждения

SET IMPLICIT\_TRANSACTIONS on;

PRINT @@TRANCOUNT

BEGIN TRANSACTION;

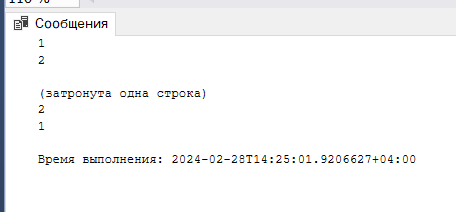
PRINT @@TRANCOUNT

INSERT INTO Human (Имя,Возраст) VALUES ('Денис', 10);

PRINT @@TRANCOUNT

COMMIT TRANSACTION;

PRINT @@TRANCOUNT



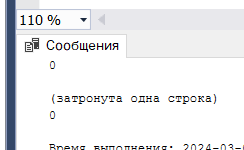
3.Неявная транзакция в режиме автоматического подтверждения

SET IMPLICIT\_TRANSACTIONS off;

PRINT @@TRANCOUNT

INSERT INTO Human (Имя,Возраст) VALUES ('Юля', 89);

PRINT @@TRANCOUNT



Так как выключен режим неявных транзакций и это неявный тип, то значение @TRANCOUNT не изменилось

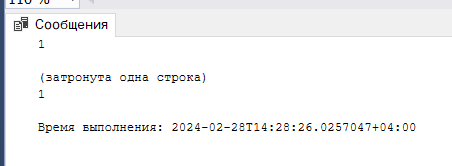
4. Неявная транзакция в режиме неявного подтверждения

SET IMPLICIT\_TRANSACTIONS on;

PRINT @@TRANCOUNT

INSERT INTO Human (Имя,Возраст) VALUES ('Никита', 21);

PRINT @@TRANCOUNT



При неявном режиме работы транзакций каждая операция представляла собой отдельную транзакцию

3 Создать несколько подключений и наглядно продемонстрировать на них разницу между различными уровнями изолированности транзакций (5 уровней).

1. Read Uncommitted

отменит операцию обновления, то черновые данные по-прежнему останутся во второй транзакции;

select \* from Human

begin transaction

set transaction isolation level read uncommitted

Update Human set Возраст=100 where id=1

waitfor delay '00:00:10'

rollback transaction

select \* from Human



Возраст не поменялся, так как мы откатили транзакцию.

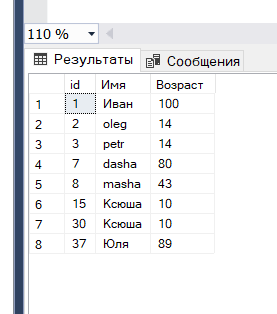
Однако второй запрос показал измененное значение возраста, несмотря на откат первой транзакции.

begin transaction

set transaction isolation level read uncommitted

select \* from Human

commit transaction



2. Read Committed

select \* from Human

begin transaction

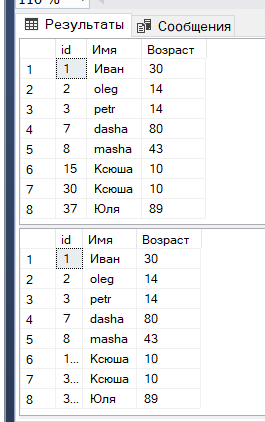
set transaction isolation level read committed

Update Human set Возраст=1 where id=1

waitfor delay '00:00:10'

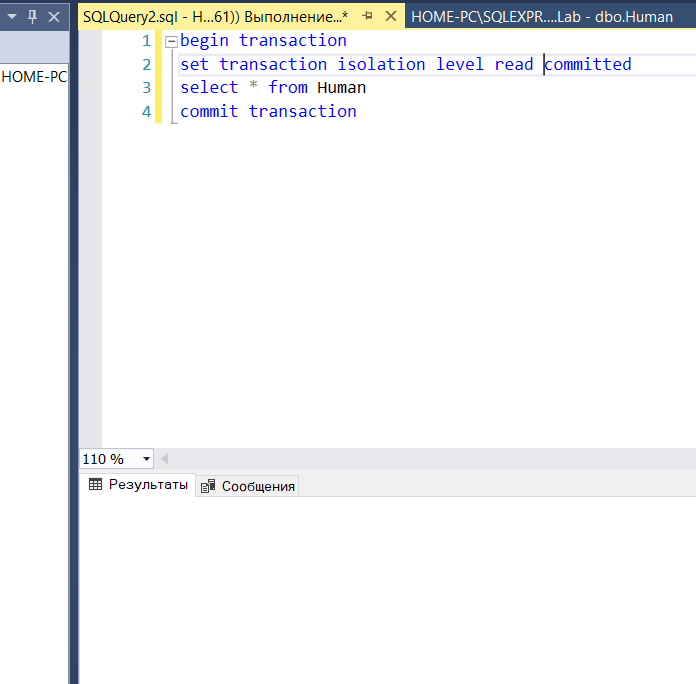
rollback transaction

select \* from Human



Данные в первом запросе не изменились.

А второй запрос не может считать данные до тех пор, пока первая транзакция не заврешится.



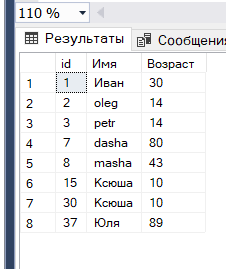
Проблема повторного чтения строк

begin transaction

set transaction isolation level read committed

select \* from Human

commit transaction



Первая транзакция считала данные

begin transaction

set transaction isolation level read committed

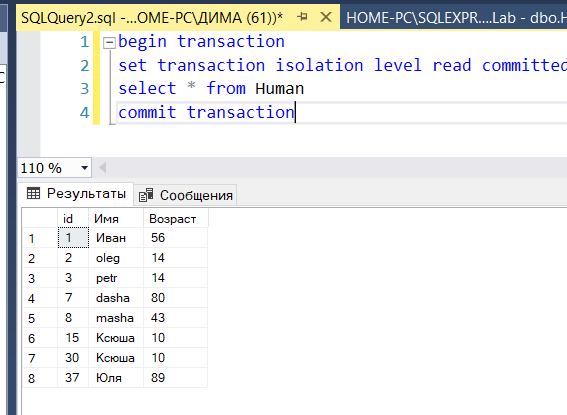
Update Human set Возраст=56 where id=1

commit transaction

select \* from Human

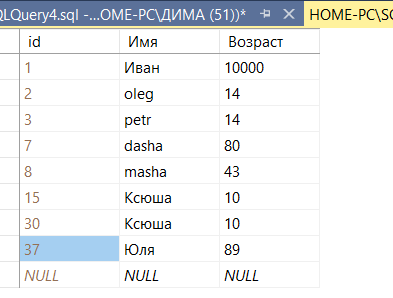
Вторая транзакция их изменила.

Теперь первая транзакция повторно считала данные:



Получается, что одна и та же транзакция показывает разные данные.

3. Repeatable read



Считываем данные таблицы с задержкой.

begin transaction

set transaction isolation level repeatable read

select \* from Human

waitfor delay '00:00:10'

select \* from Human

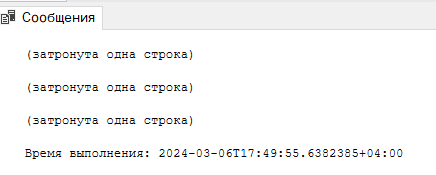
commit transaction

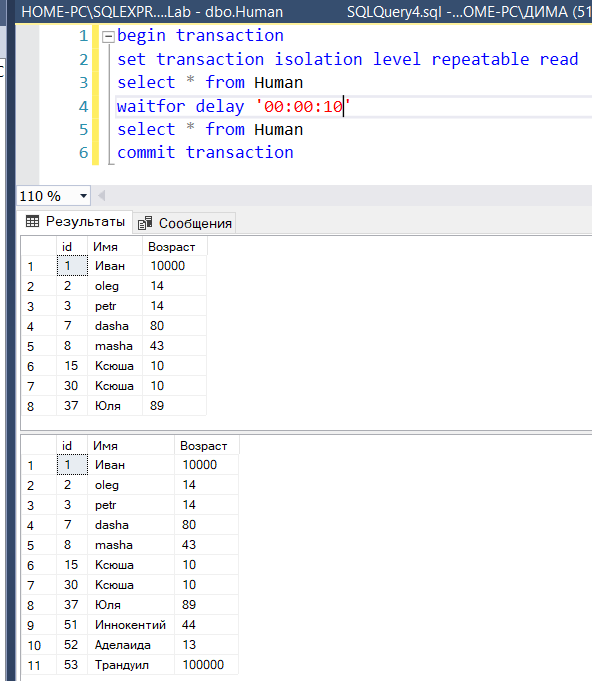
Паралельно выполняем второй запрос вставки новых данных.

insert into Human(Имя,Возраст) values ('Иннокентий',44)

insert into Human(Имя,Возраст) values ('Аделаида',13)

insert into Human(Имя,Возраст) values ('Трандуил',100000000000)



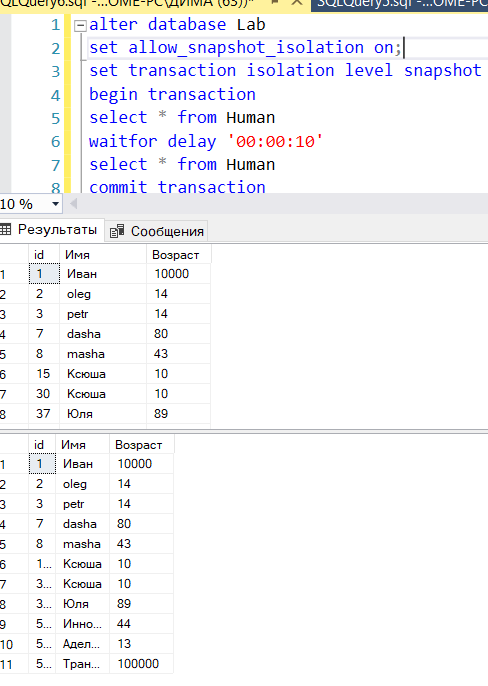


В итоге нам показывают разный набор данных.

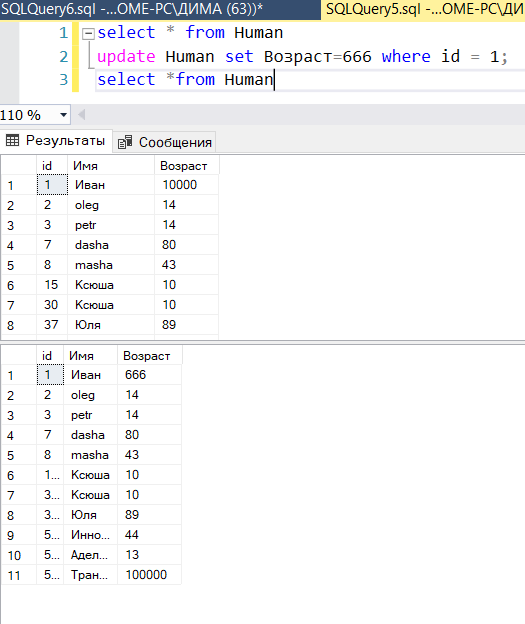
4. Snapshot

Транзакции в режиме изоляции моментального снимка не удалось получить доступ к базе данных "Lab", так как режим изоляции моментального снимка не допускается в этой базе данных. Используйте инструкцию ALTER DATABASE для разрешения использования режима изоляции моментального снимка.

Первый запрос



Второй запрос



Как видно в первом запросе данные не изменились, так как изменения, производимые другими транзакциями, в снимке не отображаются.

5. Serializable

Данный уровень исключает чтение «фантомных» записей.

begin transaction

set transaction isolation level serializable

select \* from Human

waitfor delay '00:00:10'

select \* from Human

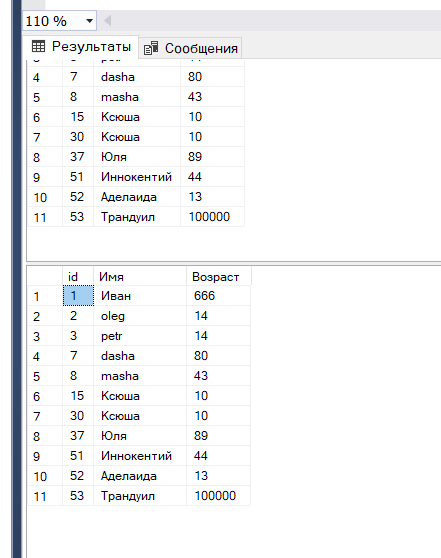
commit transaction

Второй запрос, втсавляющий данные

insert into Human(Имя,Возраст) values ('Антон',12)

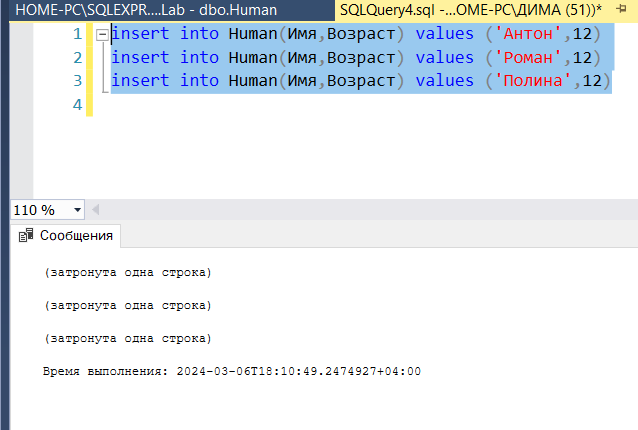
insert into Human(Имя,Возраст) values ('Роман',12)

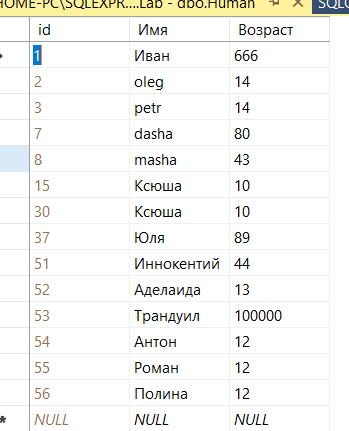
insert into Human(Имя,Возраст) values ('Полина',12)



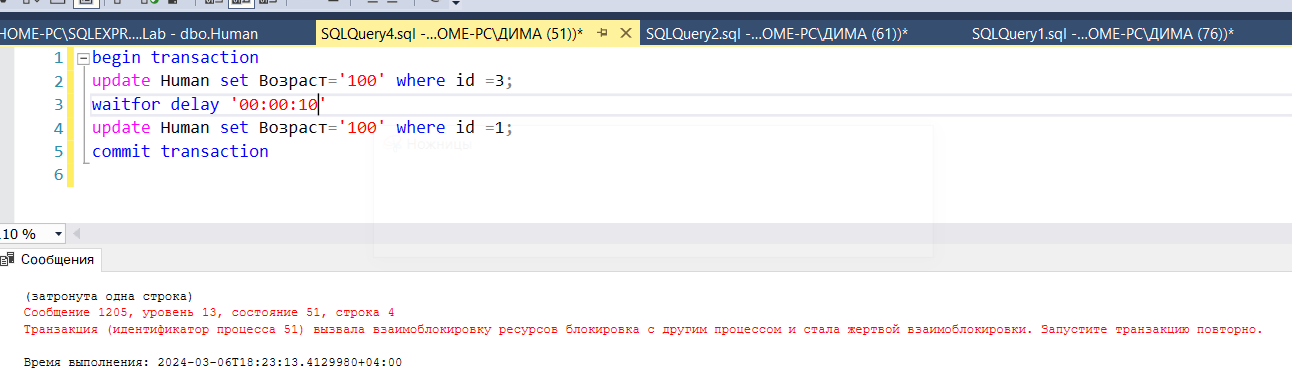
Сначала выполнился первый запрос, считывающий данные.

Только затем вылонился запрос, вставляющий новые данные.





4 Промоделировать ситуацию взаимоблокировки транзакций. Объяснить полученные результаты.



Первая транзакция

begin transaction

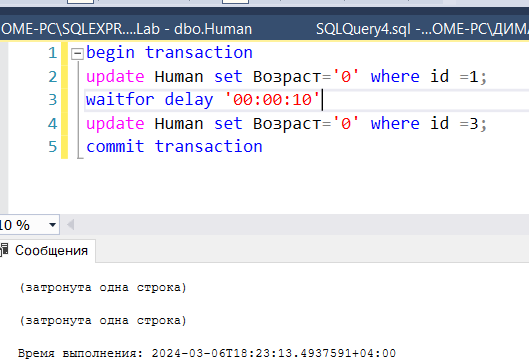
update Human set Возраст='100' where id =3;

waitfor delay '00:00:10'

update Human set Возраст='100' where id =1;

commit transaction

Вторая транзакция

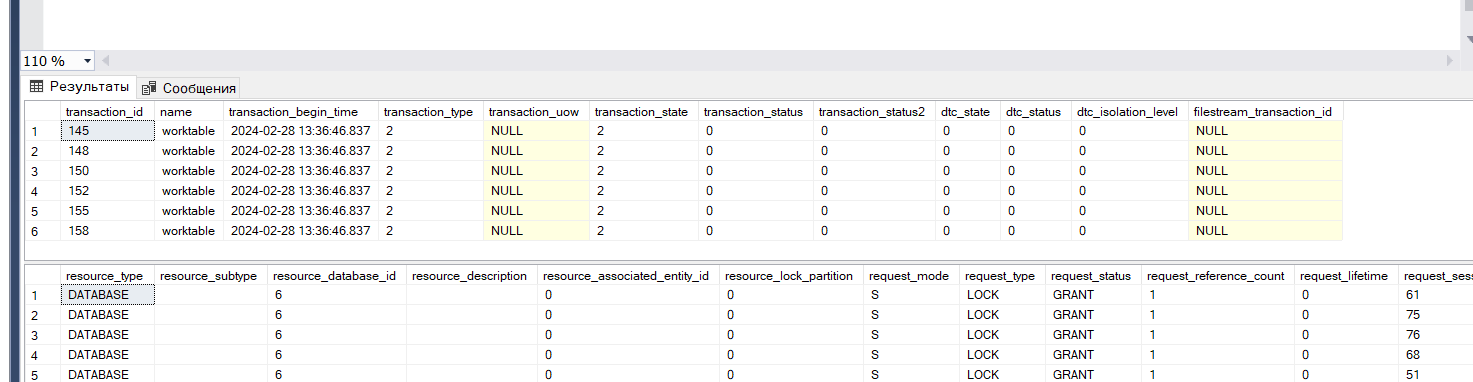


Доступ к таблице был закрыт.

5 Наглядно продемонстрировать получение информации о текущих транзакциях и блокировках, исходя из данных системного словаря.

select \* from sys.dm\_tran\_active\_transactions

select \* from sys.dm\_tran\_locks



\* Добавить связанный удалённый сервер и организовать распределённую транзакцию между БД на локальном сервере и БД на привязанном сервере.

**Вопросы к отчёту:**

* Сущность и предназначение транзакции;

Транзакции позволяют управлять набором команд, которые выполняются в базе данных, как единым целым.

Если какая-либо команда внутри транзакции завершится с ошибкой – все команды транзакции будут отмены и данные вернутся в то состояние, в каком они были до начала транзакции;

* Базовые свойства транзакции;

Свойства транзакций

Атомарность – обеспечивает, чтобы все операции входящие в единицу работы были завершены успешно. В противном случае транзакция прерывается в момент сбоя, и все предыдущие операции возвращаются в прежнее состояние.

Согласованность - Это свойство гарантирует, что данные остаются в согласованном состоянии до и после выполнения транзакции. То есть, если данные соответствовали каким-то правилам до транзакции, то после ее завершения они должны соответствовать этим правилам.

Изолированность — Это свойство определяет уровень изоляции транзакции от других параллельно выполняемых транзакций. Оно гарантирует, что результаты выполнения транзакции не будут видны другим транзакциям до ее завершения.

Устойчивость (Durability): После успешного завершения транзакции изменения, внесенные в базу данных, должны быть сохранены даже в случае сбоя системы или отключения питания.

Типы транзакций. Режимы подтверждения транзакций сервера;

1. Явная транзакция:

- Явно начинается с помощью команды BEGIN TRANSACTION и завершается командами COMMIT или ROLLBACK.

- Разработчик самостоятельно управляет началом, завершением и откатом транзакции.

- Позволяет более гибко управлять транзакциями, включая возможность вложенных транзакций.

- Предпочтительный способ для сложных операций с данными, где требуется точное контролирование транзакций

2. Неявная транзакция:

- Транзакция начинается автоматически при выполнении первой изменяющей операции (INSERT, UPDATE, DELETE) и завершается автоматически при завершении запроса.

- SQL Server автоматически фиксирует (COMMIT) транзакцию, если все операции выполнены успешно, или откатывает (ROLLBACK) ее, если произошла ошибка.

- Менее гибкий способ управления транзакциями, так как разработчик не имеет прямого контроля над началом и завершением транзакции.

- Подходит для простых операций с данными, где не требуется сложного управления транзакциями.

Что касается режимов подтверждения транзакций сервера, то они определяют поведение сервера базы данных в отношении подтверждения изменений в рамках транзакции.

IMPLICIT\_TRANSACTIONS это параметр, который управляет режимом неявных транзакций.

Если установлено значение ON, система находится в неявном режиме транзакции. Это означает, что SQL Server автоматически начинает транзакцию при выполнении первой изменяющей операции и завершает ее при завершении запроса.

Если параметр IMPLICIT\_TRANSACTIONS установлен в значение OFF, то неявные транзакции отключены, и операции изменения данных выполняются в рамках автономных операций. В этом случае, каждая операция изменения данных считается отдельной транзакцией, которая фиксируется (COMMIT) или откатывается (ROLLBACK) независимо от других операций.

@@TRANCOUNT может быть использован для проверки количества активных транзакций в текущем соединении.

Если включена явная транзакция с помощью оператора BEGIN TRANSACTION, то @@TRANCOUNT увеличивается на 1 при начале транзакции и уменьшается на 1 при фиксации (COMMIT TRANSACTION) или откате (ROLLBACK TRANSACTION) транзакции.

В режиме неявных транзакций (параметр IMPLICIT\_TRANSACTIONS равен ON), каждая операция изменения данных (INSERT, UPDATE, DELETE) автоматически начинает неявную транзакцию. @@TRANCOUNT увеличивается на 1 при начале такой неявной транзакции и уменьшается на 1 при фиксации или откате операции.

- При использовании неявных транзакций @@TRANCOUNT может увеличиться несколько раз в рамках одного запроса, если выполняются несколько операций изменения данных.

* Уровни изолированности транзакций. Read uncommitted;

READ UNCOMMITTED – не накладывает никаких ограничений на взаимодействие Обеспечивает лучшую производимость и максимальный параллелизм;

Уровень изолированности транзакций "Read Uncommitted" (Чтение незафиксированных данных) является самым низким уровнем изоляции данных в базе данных. В этом режиме транзакция может видеть изменения, сделанные другими транзакциями, даже если эти изменения еще не зафиксированы (не завершена транзакция, которая их внесла).

Основные характеристики уровня изолированности "Read Uncommitted":

1. Dirty Reads (Грязное чтение):

Чтение «грязных» данных (чтение незафиксированных данных) – такая ситуация происходит, когда одна транзакция обновляет данные в таблице, а другая, в тоже время, пытается эти данные считать. Если при этом первая транзакция отменит операцию обновления, то черновые данные по-прежнему останутся во второй транзакции;

* Уровни изолированности транзакций. Read commited;

READ COMMITTED – режим по умолчанию, разрешает транзакции считывать только зафиксированные данные. Предотвращает ситуацию чтения «грязных» данных;

Этот уровень уже запрещает грязное чтение, в данном случае все процессы, запросившие данные, которые изменяются в тот же момент в другой транзакции, будут ждать завершения этой транзакции и подтверждения фиксации данных. Данный уровень по умолчанию используется SQL сервером.

Повторное чтение строк – допустим, работают параллельно две транзакции. Первая транзакция считывает набор данных. Вторая транзакция обновляет эти данные и фиксирует изменения. Первая транзакция повторно считывает данные, которые теперь не совпадают с результатами первого запроса;

* Уровни изолированности транзакций. Repeatable read;

REPEATABLE READ – не позволяет считывание незафиксированных данных. Защищает от ситуаций чтения «грязных» данных и повторного чтения строк;

Фантомные строки – данная ситуация происходит когда одна транзакция считывает набор данных, а вторая одновременно вносит изменения в тот же набор данных. В результате первая транзакция может получить часть новых строк (обновленных) и часть старых строк (до которых вторая транзакция еще не успела дойти).

* Уровни изолированности транзакций. Snapshot;

SNAPSHOT – режим формирует моментальный снимок считываемых данных и передает его в транзакцию. Изменения, производимые другими транзакциями, в снимке не отображаются. Транзакция может изменять только снимок, а не реальные данные. Производительность режима оставляет желать лучшего. Специфический режим, используется крайне редко;

* Уровни изолированности транзакций. Serializable;

SERIALIZABLE – полностью блокирует используемые данные, пока транзакция не закончится. Обеспечивает защиту от всех негативных ситуаций, однако обладает наихудшей производительностью, в связи с чем, не может использоваться в приложениях, активно использующих базу

Данный уровень исключает чтение «фантомных» записей.

* Блокировки. Вид блокировок. Концепция менеджера блокировок;

Для обеспечения согласованности данных в случае одновременного обращения к данным несколькими пользователями применяются блокировки. Каждая прикладная программа блокирует требуемые ей данные, что гарантирует, что никакая другая программа не сможет модифицировать эти данные. Когда другая прикладная программа пытается получить доступ к заблокированным данным для их модификации, то система или завершает эту попытку ошибкой, или заставляет программу ожидать снятия блокировки.

Database (DB): Это блокировка сессии. Это нужно, чтобы предотвратить удаление базы данных, когда к ней подключены один или более пользователей.

Page (PAG): Когда SQL Server требуется заблокировать одновременно множество строк, а свободные слоты блокировок заканчиваются, то он может использовать страничные блокировки.

Эксклюзив (X); Для монопольной блокировки требуется, чтобы никакая другая активная транзакция не могла касаться заблокированного объекта, пока удерживается монопольная блокировка.

Общий (S); Общие блокировки могут быть разделены между транзакциями, которые желают только доступа для чтения к рассматриваемым данным.

Обновление (U); Блокировки обновления представляют собой комбинацию эксклюзивной и общей блокировки. Они обычно используются, когда запись или набор записей должны быть найдены и обновлены.

Менеджер блокировок в SQL отвечает за управление блокировками в рамках базы данных. Он контролирует, какие блокировки назначаются транзакциям, как долго они удерживаются и какие конфликты могут возникать между транзакциями из-за блокировок

* Взаимоблокировка. Примеры;

Взаимоблокировка (Deadlock) - это ситуация, когда две или более транзакции блокируют друг друга, ожидая освобождения ресурсов, которые удерживает другая транзакция. Это приводит к тому, что ни одна из транзакций не может завершиться, и процесс базы данных оказывается заблокированным.

* Понятие связанного сервера и распределённого запроса;

Связанный сервер (Linked Server) - это объект в SQL Server, который позволяет устанавливать соединение с другим источником данных. После настройки связанного сервера можно выполнять запросы к удаленным источникам данных, как если бы они были локальными.

Распределенный запрос (Distributed Query) - это запрос, который обращается к данным, находящимся на различных серверах или источниках данных. Распределенные запросы позволяют объединять данные из различных источников в одном запросе и выполнять операции над этими данными.

* Структура файлов журналов. Понятие логического и физического журналов, LSN, minLSN, контрольной точки;

Файлы журналов (Transaction Log Files) - это часть базы данных SQL Server, которая записывает все транзакции, происходящие в базе данных. Файлы журналов имеют важное значение для обеспечения целостности данных и восстановления базы данных в случае сбоев.

Журнал транзакций может располагаться в нескольких файлах.

1. Логический и физический журналы:

- Логический журнал (Logical Log) - это логическая структура, которая содержит информацию о транзакциях, изменениях данных и других операциях в базе данных.

- Физический журнал (Physical Log) - это файл на диске, который хранит фактические данные логического журнала.

2. LSN (Log Sequence Number):

- LSN (Log Sequence Number) - это уникальный идентификатор, который присваивается каждой записи в логическом журнале.

3. minLSN (Minimum Log Sequence Number):

- minLSN (Minimum Log Sequence Number) - это минимальный LSN, который указывает на начало транзакции или операции в логическом журнале. minLSN используется при выполнении восстановления базы данных для определения точки начала восстановления.4

4. Контрольная точка (Checkpoint):

- Контрольная точка (Checkpoint) - это момент времени, когда SQL Server записывает все изменения из буфера данных на диск и обновляет информацию о базе данных в файлах данных. Контрольные точки позволяют уменьшить время восстановления базы данных и уменьшить возможные потери данных при сбоях.

* Распределённые транзакции. Двухфазная фиксация. Компонент DTC;

Распределённые транзакции - это транзакции, которые включают в себя операции, выполняемые на нескольких различных серверах или ресурсах данных. Например, при выполнении транзакции, которая включает операции записи данных в базу данных на одном сервере и операции чтения данных из другой базы данных на другом сервере, мы имеем дело с распределённой транзакцией.

Двухфазная фиксация (Two-Phase Commit) - это механизм обеспечения атомарности распределённых транзакций. Он состоит из двух фаз:

1. Первая фаза (Фаза голосования): Координатор (Coordinator) отправляет запрос на подтверждение завершения транзакции всем участникам (Participants). Участники голосуют за фиксацию (Commit) или откат (Rollback) транзакции.

2. Вторая фаза (Фаза завершения): Если все участники проголосовали за фиксацию, то координатор отправляет команду фиксации всем участникам. В противном случае, если хотя бы один участник проголосовал за откат, то координатор отправляет команду отката всем участникам.

Компонент DTC (Distributed Transaction Coordinator) - это компонент в операционной системе Windows, который отвечает за управление распределёнными транзакциями. DTC обеспечивает координацию между различными серверами и ресурсами данных, управляет двухфазной фиксацией и обеспечивает целостность и надежность распределённых транзакций.