Администрирование многопользовательских систем управления баз данных

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnovmgupi@gmail.com

Лекция 2

Управление многопользовательскими БД.

Понятие многопользовательской БД

Базы данных, одним из ключевых требований реализации которых является возможность совместной работы нескольких пользователей с одной и той же совокупностью данных называются много пользовательскими.

Ключевые функции администратора БД

1. Управление структурой БД.
2. Управление параллельной обработкой данных (блокировки).
3. Распределение прав и обязанностей по обработке данных (один из факторов обеспечения безопасности информации).
4. Обеспечение безопасности БД.
5. Осуществление процедур сохранения, восстановления и переноса БД.
6. Управление СУБД.
7. Поддержание репозитория данных в рабочем состоянии.

Функции администратора по управлению структурой БД

1. Участие в разработке БД и приложений:
- оказание поддержки на стадии определения требований к БД;
- Частичное участи в проектировании и реализации БД.
2. Оказание помощи в изменении структуры БД:
- поиск решений во взаимодействии с пользователями;
- оценка воздействия изменений в БД на изменение в бизнес-процессах организации и опыт взаимодействия пользователя с приложениями баз данных;
- обсуждение вопросов управления конфигурациями БД;
- проработка вариантов решения проблем, возникающих после внесения изменений в конфигурации БД;
- ведение сопутствующей документации.

Определение параллельной обработки данных

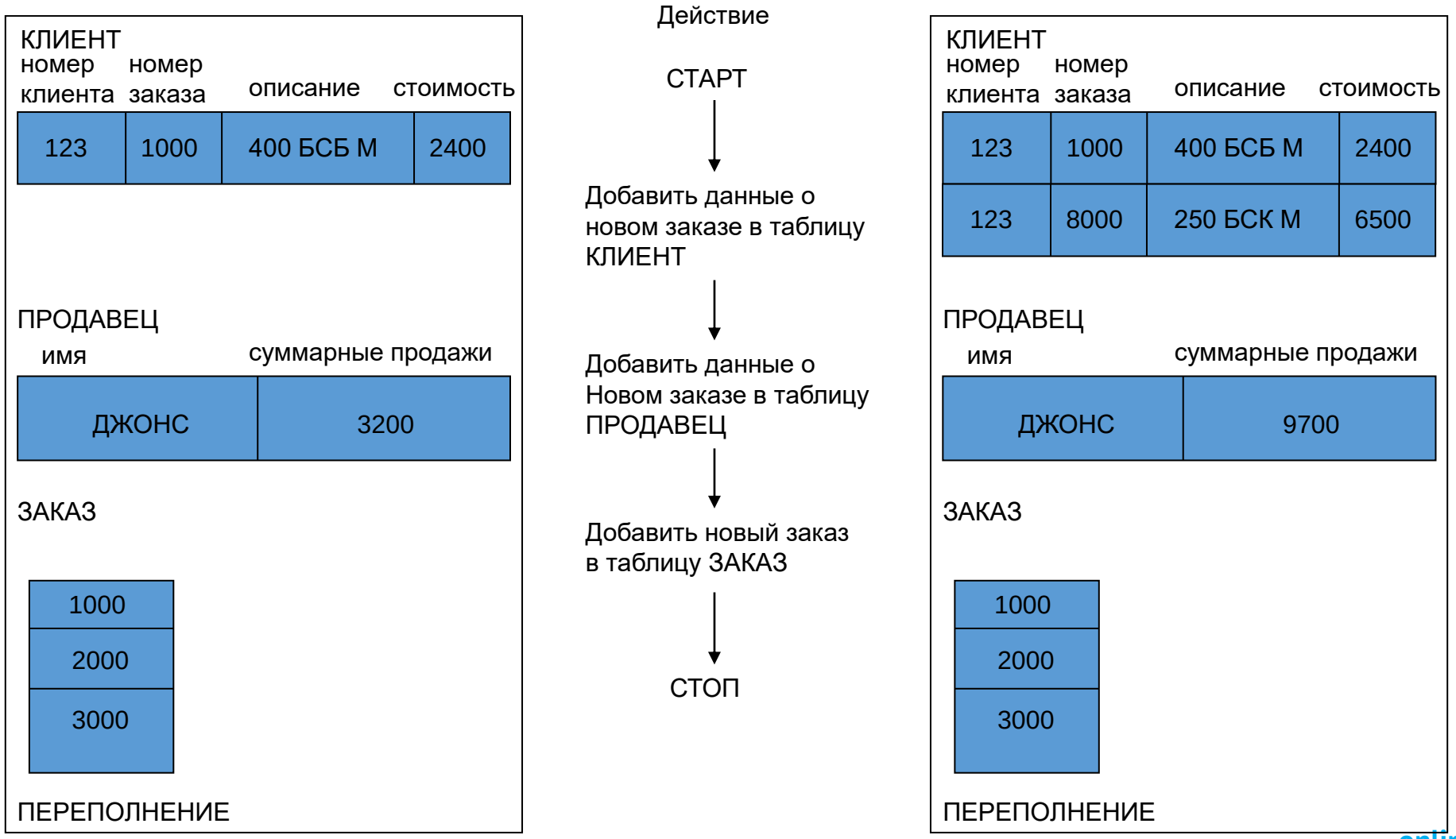
Управление параллельной обработкой данных – меры и мероприятия, предпринимаемые с целью ограничить влияние действий, осуществляемых пользователями во время сеанса работы с БД друг на друга.

Атомарные транзакции (LUW)

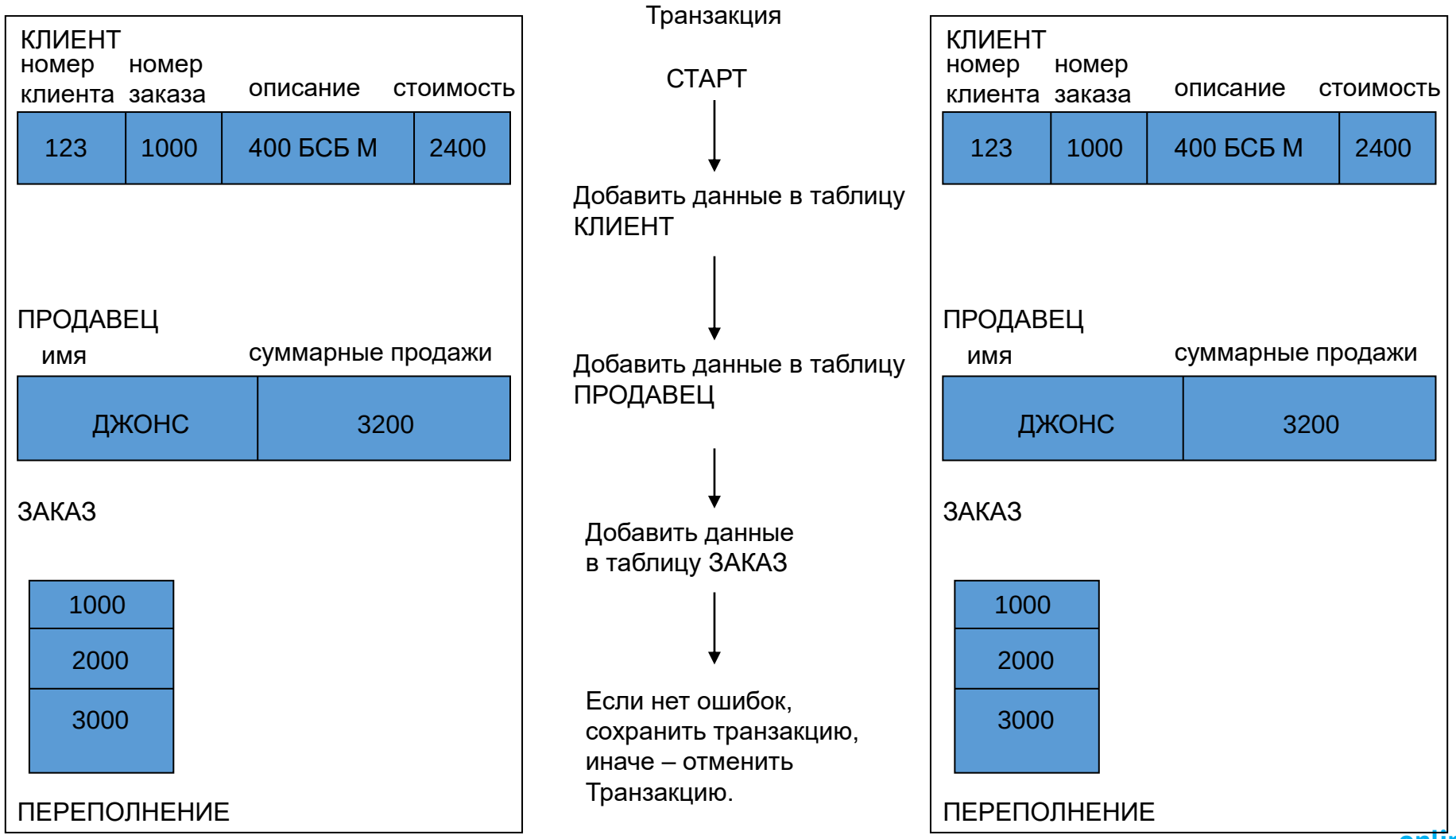
Ключевая функция многопользовательских СУБД. Применяется в большинстве ПО по управлению базами данных.

АТ (атомарной транзакцией) или LUW (logical units of work) считаются серии действий, предпринимаемых с базой данных, которые выполняются успешно все, или не выполняются совсем (в обратном случае).

Визуализация проблемы потерянного обновления



Визуализация решения проблемы LUW через транзакцию



Логическое описание транзакции к приведенному примеру

BEGIN Транзакция
  Изменить данные Клиент
  Изменить данные Продавец
  Вставить данные в Заказ
IF Все успешно THEN
  COMMIT Транзакция
ELSE
  ROLLBACK Транзакция
END IF

```
USE test;

-- Начало транзакции
BEGIN TRANSACTION
    UPDATE Employee
    SET Id = 14568
    WHERE Id = 10102

    IF (@@error <> 0)
-- Отменить транзакцию, если есть ошибки
ROLLBACK
COMMIT
```

*хорошая статья по синтаксису транзакций в T-SQL с примерами:
https://professorweb.ru/my/sql-server/2012/level3/3_14.php

*описание функции @@error в T-SQL с примерами:
https://msdn.microsoft.com/ru-ru/library/ms188790(v=sql.120)

Проблема параллельной обработки запросов

Два источника проблемы параллельной обработки данных или запросов:

Проблема потерянного обновления (lost update problem, LUP) – пользователи получают при параллельной обработке запроса верные данные на тот момент, когда они составили запрос.

Проблема непоследовательного чтения (inconsistent read problem, IRP) – некоторые пользователи получают изначально неверные данные (ситуация, обратная LUP).

Нормальный режим параллельной обработки

Пользователь А

1. Считать элемент 100
2. Изменить элемент 100
3. Записать элемент 100

Пользователь Б

1. Считать элемент 200
2. Изменить элемент 200
3. Записать элемент 200

Порядок обработки на сервере БД

1. Считать элемент 100 для А
2. Считать элемент 200 для Б
3. Изменить элемент 100 для А
4. Записать элемент 100 для А
5. Изменить элемент 200 для Б
6. Записать элемент 200 для Б

Демонстрация проблемы “потерянного обновления”

Пользователь А

1. Считать элемент 100 (пусть количество элементов равно 10)
2. Уменьшить количество элементов на 5
3. Записать элемент 100

Пользователь Б

1. Считать элемент 100 (пусть количество элементов равно 10)
2. Уменьшить количество элементов на 3
3. Записать элемент 100

Порядок обработки на сервере БД

1. Считать элемент 100 (для А)
2. Считать элемент 100 (для Б)
3. Установить количество элементов равным 5 (для А)
4. Записать элемент 100 для А
5. Установить количество элементов равным 7 (для Б)
6. Записать элемент 100 для Б

Введение блокировок, как решения проблемы параллельной обработки

Решение проблем – использование оператора SQL lock.

Блокировка (lock) – принудительная блокировка элемента, подлежащего изменению, на время проведения транзакции. Элемент на это время полностью или частично блокирует доступ к себе иным пользователям кроме транзакции.

Сопутствующая терминология:
Неявная блокировка (implicit lock) – устанавливается СУБД
Гранулярность блокировки (granularity) – определяет масштаб блокируемых элементов
Исключительная блокировка (exclusive lock) – полностью блокирует выбранный элемент.
Мягкая блокировка (shared lock) – блокирует элемент от изменения, но позволяет с него считывать данные.

Параллельная обработка с явными блокировками

Пользователь А

1. Заблокировать элемент 100
2. Считать элемент 100
3. Уменьшить количество элементов на 5
4. Записать элемент 100

Пользователь Б

1. Заблокировать элемент 100
2. Считать элемент 100
3. Уменьшить количество элементов на 3
4. Записать элемент 100

Порядок обработки на сервере БД

Заблокировать элемент 100 для А
Считать элемент 100 для А
Заблокировать элемент 100 для Б -> неудача -> Б переводится в состояние ожидания
Установить количество элементов равным 5 для А
Записать элемент 100 для А
Снять блокировку А с элемента 100
Заблокировать элемент 100 для Б
Считать элемент 100 для Б
Установить количество элементов равным 2 для Б
Записать элемент 100 для Б
Снять блокировку Б с элемента 100

Взаимная блокировка, “deadlock”

Пользователь А

Заблокировать бумагу
Взять бумагу
Заблокировать карандаши

Пользователь Б

Заблокировать карандаши
Взять карандаши
Заблокировать бумагу

Порядок обработки на сервере БД

Заблокировать бумагу для А
Заблокировать карандаши для Б
Обработать запрос А, обновить данные о бумаге
Обработать запрос Б, обновить данные о карандашах
Перевести А в состояние ожидания (карандашей)
Перевести Б в состояние ожидание (бумаги)

БЛОКИРОВКА

Принцип оптимистической блокировки

Все пройдет хорошо, процесс будет успешный, конфликта транзакции не случится.

Если случится конфликт – переделаем транзакцию столько раз, сколько нужно для отсутствия конфликта.

Блокировка ставится только на транзакцию.

Пример оптимистической блокировки

```
SELECT    PRODUCT.Name, PRODUCT.Quantity
FROM      PRODUCT
WHERE     PRODUCT.Name = 'Pencil';

Set NewQuantity = PRODUCT.Quantity - 5;

{process transaction - take exception action if NewQuantity < 0, etc.

Assuming all is OK: }

LOCK      PRODUCT;

UPDATE    PRODUCT
SET       PRODUCT.Quantity = NewQuantity
WHERE     PRODUCT.Name = 'Pencil'
      AND PRODUCT.Quantity = OldQuantity;

UNLOCK    PRODUCT;

{check to see if update was successful;
if not, repeat transaction}
```

Принцип пессимистической блокировки

Все пройдет плохо, обязательно может случиться конфликт.

Блокируются все запросы к элементу БД, включая транзакцию.

Проверки на успешность транзакции не требуется.

Пример пессимистической блокировки

```
LOCK      PRODUCT;

SELECT    PRODUCT.Name, PRODUCT.Quantity
FROM      PRODUCT
WHERE     PRODUCT.Name = 'Pencil';

Set NewQuantity = PRODUCT.Quantity - 5;

{process transaction - take exception action if NewQuantity < 0, etc.

Assuming all is OK: }

UPDATE    PRODUCT
SET       PRODUCT.Quantity = NewQuantity
WHERE     PRODUCT.Name = 'Pencil';

UNLOCK    PRODUCT;

{no need to check if update was successful}
```

Настройка уровней изоляции транзакции

Тип проблемы	Незавершенное чтение	Завершенное чтение	Воспроизв. чтение	Сериализуемость
Грязное чтение	Возможно	Невозможно	Невозможно	Невозможно
Невоспроизвод. чтение	Возможно	Возможно	Невозможно	Невозможно
Фантомное чтение	Возможно	Возможно	Возможно	Невозможно

Пример настройки транзакции

```
SET NOCOUNT ON
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ DECLARE
@Id int
WHILE 1 = 1 BEGIN
SET @Id = 500000 * rand()
        BEGIN TRANSACTION
        UPDATE DevicesData SET Value = Value + 1 WHERE
DeviceId = @Id
        UPDATE DevicesData SET Value = Value - 1 WHERE
DeviceId = 500000 + @Id
COMMIT
WAITFOR DELAY '00:00:00.100’
END
```

*страница с примерами настроек Т https://arbinada.com/en/node/619
*чуть больше теории по настройкам Т https://habrahabr.ru/post/317884/

Спасибо за внимание!