**24. Объекты и концепции базы данных. Транзакции. Управление транзакциями**

**Транзакция** - это группа последовательно выполняемых операторов SQL, которые либо должны быть выполнены все, либо не должен быть выполнен ни один из них. **Главная задача транзакций** - обеспечить целостность данных в случаях, когда несколько SQL-операторов выполняют зависящие друг от друга изменения данных. Классический пример, приводимый, наверно, во всех учебниках по базам данных - перевод денег с одного счета на другой:

UPDATE accounts SET AccSum = AccSum - 1000 WHERE AccNumber = 12345;

UPDATE accounts SET AccSum = AccSum + 1000 WHERE AccNumber = 67890;

Что будет, если первый оператор выполнится, а второй по какой-то причине - нет (сбой сервера, неправильный номер счета, переполнение... - мало ли какая может быть ошибка)? Деньги с одного счета списаны, а на другой не поступили...

Механизм **транзакций** как раз и позволяет корректно выходить из подобных ситуаций. Объединив эти два оператора UPDATE в одну транзакцию, мы обеспечим выполнение (или невыполнение) их обоих как одного целого.

Стандарт SQL-92 предусматривает, что транзакция в текущей сессии начинается неявно при выполнении первого из SQL-операторов, изменяющих данные, а завершена может быть либо явно - операторами COMMIT [WORK] или ROLLBACK (COMMIT [WORK] подтверждает транзакцию и фиксирует внесенные изменения, ROLLBACK транзакцию откатывает и возвращает данные в базе к тому состоянию, в каком они были до начала транзакции), либо неявно - при завершении текущей сессии или при системном сбое. К сожалению, стандарт SQL-92 не определяет, что должно делаться по умолчанию при завершении текущей сессии - подтверждение транзакции или откат, и в разных СУБД (и даже, бывает, в разных клиентах для одной СУБД) это реализуется по-разному. Поэтому на действия по умолчанию лучше не полагаться и всегда делать COMMIT / ROLLBACK явно.

**А вот в MySQL с транзакциями все гораздо хитрее...**

Начнем с того, что в MySQL существуют таблицы нескольких типов: ISAM, HEAP, MyISAM, InnoDB, BDB... Из них транзакции могут поддерживать только два последних: InnoDB и BDB. При этом "родными" для MySQL являются таблицы типа MyISAM, которые транзакции не поддерживают. А для того, чтобы можно было создавать транзакционные таблицы типов InnoDB и BDB и работать с ними, сервер MySQL должен быть соответствующим образом сконфигурирован (что далеко не всегда во власти разработчиков приложений, особенно веб-приложений).

По умолчанию MySQL работает в режиме autocommit. Это означает, что результаты выполнения любого SQL-оператора, изменяющего данные, будут сразу фиксироваться (записываться на диск).

Режим autocommit можно отключить командой SET AUTOCOMMIT=0. При отключенном режиме autocommit каждую транзакцию надо явно завершать операторами COMMIT / ROLLBACK.

Для однократного перехода в транзакционный режим можно использовать команду START TRANSACTION (в MySQL начиная с версии 4.0.11) или BEGIN [WORK]:

START TRANSACTION;

UPDATE accounts SET AccSum = AccSum - 1000 WHERE AccNumber = 12345;

UPDATE accounts SET AccSum = AccSum + 1000 WHERE AccNumber = 67890;

COMMIT;

Обратите внимание на серьезный подводный камень. Если в транзакции участвуют таблицы разных типов (транзакционные и нетранзакционные), то вести они себя будут совсем по-разному. Изменения в нетранзакционные таблицы будут вноситься немедленно, несмотря на отключенный режим autocommit, и их невозможно будет откатить с помощью оператора ROLLBACK! А если ROLLBACK все же будет выполнен - изменения, внесенные в транзакционные таблицы, откатятся, а в нетранзакционные - сохранятся. К чему это может привести - думаю, разжевывать не надо...

Все операции, выполняемые с данными на SQL сервере, происходят в контексте транзакций. **Транзакция** - это групповая операция, т.е. набор действий с базой данных; самым существенным для этих действий является правило либо все, либо ни чего. Если во время выполнения данного набора действий, на каком-то этапе невозможно произвести очередное действие, то нужно выполнить возврат базы данных к начальному состоянию (произвести откат транзакции). Таким образом (при правильном планировании транзакций), обеспечивается целостность базы данных. В данном уроке объясняется, как начинать, управлять и завершать транзакции с помощью SQL выражений. А так же рассматривается вопрос об использовании транзакций в приложениях, созданных в Delphi. Вся приведенная информация касается InterBase.

SQL-выражения для управления транзакциями

**Для управления транзакциями имеется три выражения:**

SET TRANSACTION - Начинает транзакцию и определяет ее поведение.

COMMIT - Сохраняет изменения, внесенные транзакцией, в базе данных и завершает транзакцию.

ROLLBACK - Отменяет изменения, внесенные транзакцией, и завершает транзакцию.

**1. Запуск транзакции**

Выполнять транзакции можно, например, из Windows Interactive SQL, из программы, из сохраненной процедуры или триггера. В общем виде, синтаксис команды SQL для запуска транзакции:

SET TRANSACTION [Access mode] [Lock Resolution]

[Isolation Level] [Table Reservation]

Значения, принимаемые по-умолчанию:

выражение

SET TRANSACTION

равносильно выражению

SET TRANSACTION READ WRITE WAIT ISOLATION LEVEL SNAPSHOT

Access Mode - определяет тип доступа к данным. Может принимать два значения:

READ ONLY - указывает, что транзакция может только читать данные и не может модифицировать их.

READ WRITE - указывает, что транзакция может читать и модифицировать данные. Это значение принимается по умолчанию.

Пример:

SET TRANSACTION READ WRITE

Isolation Level - определяет порядок взаимодействия данной транзакции с другими в данной базе. Может принимать значения:

SNAPSHOT - значение по умолчанию. Внутри транзакции будут доступны данные в том состоянии, в котором они находились на момент начала транзакции. Если по ходу дела в базе данных появились изменения, внесенные другими завершенными транзакциями, то данная транзакция их не увидит. При попытке модифицировать такие записи возникнет сообщение о конфликте.

SNAPSHOT TABLE STABILITY - предоставляет транзакции исключительный доступ к таблицам, которые она использует. Другие транзакции смогут только читать данные из них.

READ COMMITTED - позволяет транзакции видеть текущее состояние базы.

Конфликты, связанные с блокировкой записей происходят в двух случаях:

Транзакция пытается модифицировать запись, которая была изменена или удалена уже после ее старта. Транзакция типа READ COMMITTED может вносить изменения в записи, модифицированные другими транзакциями после их завершения.

Транзакция пытается модифицировать таблицу, которая заблокирована другой транзакцией типа SNAPSHOT TABLE STABILITY.

Lock Resolution - определяет ход событий при обнаружении конфликта блокировки. Может принимать два значения:

WAIT - значение по умолчанию. Ожидает разблокировки требуемой записи. После этого пытается продолжить работу.

NO WAIT - немедленно возвращает ошибку блокировки записи.

Table Reservation - позволяет транзакции получить гарантированный доступ необходимого уровня к указанным таблицам. Существует четыре уровня доступа:

PROTECTED READ - запрещает обновление таблицы другими транзакциями, но позволяет им выбирать данные из таблицы.

PROTECTED WRITE - запрещает обновление таблицы другими транзакциями, читать данные из таблицы могут только транзакции типа SNAPSHOT или READ COMMITTED.

SHARED READ - самый либеральный уровень. Читать могут все, модифицировать - транзакции READ WRITE.

SHARED WRITE - транзакции SNAPSHOT или READ COMMITTED READ WRITE могут модифицировать таблицу, остальные - только выбирать данные.

**2. Завершение транзакции**

Когда все действия, составляющие транзакцию успешно выполнены или возникла ошибка, транзакция должна быть завершена, для того, чтобы база данных находилась в непротиворечивом состоянии. Для этого есть два SQL-выражения:

COMMIT - сохраняет внесенные транзакцией изменения в базу данных. Это означает, что транзакция завершена успешно.

ROLLBACK - откат транзакции. Транзакция завершается и никаких изменений в базу данных не вносится. Данная операция выполняется при возникновении ошибки при выполнении операции (например, при невозможности обновить запись).