# Нестандартные модели транзакций*

Г.Р. Домбровская
Санкт-Петербургский университет
E-mail: henr@orl2.usr.pu.ru

## Аннотация

Данная статья представляет собой обзор разработанных в течение последнего десятилетия различных нетрадиционных моделей транзакций, общим признаком которых является частичный отказ от базовых свойств транзакций: атомарности, целостности, изолированности и устойчивости Обсуждаются причины ограниченной применимости традиционных транзакций в современных приложениях, приводится классификация существующих нетрадиционных моделей транзакций, подробно характеризуются отдельные модели. В заключении кратко обозначены направления развития, не отраженные в основной части обзора.

## 1 Введение

Понятие транзакции как логически неделимого действия является одним из важнейших как в теории баз данных, так и, в особенности, в практической реализации. Даже в однопрограммных системах, при отсутствии конфликтов одновременного доступа, для обеспечения надежности работы требуется определить атомарные действия над базой данных для того, чтобы исключить возможность приведения ее в противоречивое состояние. Еще большее значение поддержка транзакций приобретает в многопользовательских системах, в особенности в распределенных, в которых, кроме того, что к одним данным могут обращаться одновременно несколько пользователей, сами данные могут быть расположены в разных локальных узлах.

Исследования в этой области в 70-х и начале 80-х годов были сосредоточены на установлении базовых свойств транзакций, получивших впоследствии наименование ACID свойств Г(atomicity, consistency, isolation, durability) [11, 13, 26], определяемых следующим образом:

**Атомарность** – принцип "все или ничего". Означает, что все операции, производимые транзакцией, рассматриваются как единое целое, и следовательно, должны быть выполнены либо все, либо ни одной.

**Целостность** – устанавливает корректность транзакции, т.е. в случае, если транзакция выполняется независимо от других, она переводит базу данных из одного корректного (непротиворечивого) состояния в другое.

**Изолированность** , или независимость – означает, что транзакция может оперировать только данными, находящимся в непротиворечивом состоянии, не имея доступа к промежуточным результатам других транзакций.

**Устойчивость** – устанавливает, что результаты успешно завершенной транзакции должны быть гарантированно занесены в базу данных независимо от возможных последовавших сбоев системы.

Кроме того, рассматривались возможности и условия для одновременного выполнения нескольких транзакций (протоколы параллельного выполнения) [8, 28, 16, 3] и обеспечения корректного завершения транзакций (протоколы восстановления) [34, 2]. В результате сложилась концепция сериализуемости расписаний выполнения транзакций, гарантирующая корректное параллельное выполнение.

Различные варианты протоколов обоих типов были практически реализованы в промышленных системах, таких как ГDB/2 [5], ORION [10], Starburst [32], Informix [6] и других.

Впоследствии, однако, обнаружились ограниченность классической модели транзакций. Эта

*Работа поддержана Российским Фондом Фундаментальных исследований (грант 95-01-00636) и АОЗТ Урбансофт (контракт 35/95).

ограниченность наиболее явно проявилась в следующих двух областях приложений баз данных: при выполнении транзакций в распределенных системах и в CAD/CAM приложениях, в которых выполнение транзакции требует длительного времени. В результате последующих исследований был предложен ряд новых моделей транзакций, не обладающих набором ACID свойств: это, во-первых, различные варианты вложенных транзакций, политранзакции и другие, применяемые в распределенных системах [1, 17], и, во-вторых, различные типы долгоживущих транзакций, например, саги [9].

Наиболее простой моделью такого типа, лежащей в основе большинства других, являются традиционные вложенные транзакции. Предполагается, что любая транзакция может порождать новые, вложенные транзакции, выполнение которых, вообще говоря, планируется независимо, но при этом все порожденные транзакции должны быть завершены раньше родительской. При этом происходит частичный отказ от принципов атомарности и независимости, поскольку выполнение порождающей транзакции зависит от выполнения порожденных.

Важным аспектом функционирования СУБД является возможность осуществлять восстановления состояния базы данных в случае аварийного завершения системы, а также возможность отката внесенных изменений. Наличие транзакций играет существенную роль в поддержке таких функций, однако для их успешного осуществления требуется наличие дополнительных структур, называемых журналами транзакций. В течение последних лет был разработан ряд алгоритмов для ведения таких журналов и осуществления откатов и восстановлений [21]. При поддержке в системе нестандартных моделей транзакций сложность ведения таких журналов существенно возрастает. Некоторые варианты подобных журналов описаны в работах [24, 22].

Как поддержка транзакций, так и ведение журналов, могут осуществляться на разных уровнях системы. Эффективность такой поддержки тем больше, чем на более низком уровне она осуществляется. Однако такого рода низкоуровневая поддержка имеет свои особенности в каждой конкретной системе.

# 2 Ограниченность традиционных транзакций

Нетрадиционные модели транзакций используются в наиболее часто в следующих областях применения СУБД: в распределенных системах и в системах реального времени, например, в CAD/CAM приложениях, при использовании CASE-средств, в издательских системах и т.п.

Типичным примером их использования являются мультибазы данных, то есть системы, интегрирующие множество существующих, возможно, разнородных, баз данных. Многие свойства традиционных транзакций либо оказываются в этих условиях не нужными, либо их становится трудно поддерживать.

Так, одним из основных требований в системах мультибаз данных является соблюдения принципа локальной автономии. Однако при его соблюдении становится сложно обнаружить конфликты между глобальными транзакциями, что, в свою очередь, усложняет глобальную сериализацию. Следование двухфазному протоколу завершения распределенных транзакций [12, 19], гарантирующему сериализуемость, снижает возможность параллельного выполнения, уменьшая пропускную способность системы.

Более того, локальная система имеет право приостановить, или отложить выполнение, или вообще отклонить глобальную транзакцию. Глобальная транзакция может стать долгоживущей и имеет большую вероятность оказаться ошибочной. Поддержка в таких условиях свойств атомарности и независимости потребует больших накладных расходов и также скажется на производительности системы.

Понятие долгоживущей транзакции, вообще говоря, не предполагает какую-то определенную модель. Так называют сложные транзакции, требующие доступа к большому числу элементов данных, и выполняющиеся в течении длительного времени. Такие транзакции обычно встречаются в системах автоматизированного проектирования, издательских системах и т.п. Для долгоживущих транзакций, во-первых, становится трудно реализовать принцип атомарности, поскольку из-за их большой продолжительности увеличивается вероятность их аварийного завершения. При этом необходимость отката, очевидная для коротких транзакций, приводит к потере большого объема проделанной работы. Во-вторых, становится трудно реализовать принцип независимости (изолированности). Поскольку эти транзакции имеют доступ к большим

объемам данных, их захват на все время выполнения необходимо вызовет длительные ожидания других транзакций (часы и дни).

Еще одним ограничением традиционной модели является невозможность организовать взаимодействие между различными параллельными деятельностями. Проектирование в CAD обычно осуществляется несколькими группами разработчиков, которым требуется взаимодействие, в частности, доступ к некоторым разделяемым данным. Это требование приходит в противоречие с независимостью традиционных транзакций.

В следующий разделах будут рассмотрены модели транзакций, снимающие многие из перечисленных ограничений.

# 3 Вложенные транзакции

Впервые модель вложенных транзакций была предложена в работе [23]. В этой модели любая транзакция может иметь несколько вложенных транзакций (подтранзакций), каждая из них, в свою очередь, может иметь собственные подтранзакции, таким образом, таким образом, транзакции в целом может быть поставлено в соответствие *дерево транзакций*. Корень этого дерева называется корневой транзакцией или *транзакцией верхнего уровня*. Все транзакции, располагающиеся на пути от корня до текущей подтранзакции, называются ее предками.

Транзакции верхнего уровня обладают полным набором ACID свойств традиционных транзакций. В случае аварийного завершения корневой транзакции производится ее откат таким образом, чтобы не создавать побочных эффектов для других транзакций в системе. Для подтранзакций гарантированно выполняется только свойство атомарности, они завершаются нормально или аварийно независимо от других транзакций системы. Зависимость нормального или аварийного завершения порожденных и родительских транзакций может быть суммирована следующим образом:

- при аварийном завершении порожденной транзакции вопрос о ее откате, рестарте или других действиях решается родительской транзакцией, никак не влияя на завершение последней;
- при аварийном завершении родительской транзакции все ее потомки подлежат откату.

При аварийном завершении подтранзакции родительской транзакцией может быть предпринято одно из следующих действий:

- проигнорировать аварийное завершение, в этом случае подтранзакция считается *несущественной*;
- инициировать рестарт аварийной подтранзакции;
- инициировать выполнение другой, *зависимой* подтранзакции, выполняющей альтернативные действия;
- наконец, в случае невозможности каким-либо образом компенсировать аварийное завершение подтранзакции, родительская транзакция сама завершается аварийно, производя откат всех своих потомков.

Протокол параллельного выполнения для вложенных транзакций базируется на стандартном двухфазном протоколе блокировок. Подтранзакции выполняются как независимые транзакции. Вопрос блокировок решается следующим образом: если данные заблокированы какой-либо транзакцией, ее потомки не имеют к ним доступа. Когда подтранзакция завершается, заблокированные ею данные передаются родительской транзакции, иначе говоря, *наследуются* ею. Блокировки снимаются после завершения транзакции верхнего уровня.

Описанная система блокировок дает возможность поддерживать независимость деревьев транзакций, обеспечивает безопасный доступ к данным.

Модель вложенных транзакций наиболее существенна для всех других нестандартных моделей, поскольку именно в ней была впервые применена идея структурирования транзакции в виде дерева, а также такие понятия, как несущественная транзакция и зависимая транзакция. Большинство нестандартных моделей транзакций представляет собой различные варианты и модификации вложенных транзакций.

# 4 Саги

Другой базовой моделью нестандартных транзакций является *сага*. Эта модель базируется на концепции *компенсационных транзакций* [11], и призвана решать проблемы, возникающие при выполнении долгоживущих транзакций.

Компенсационной транзакцией для транзакции $T$ является такая транзакция $C$, которая после завершения транзакции $T$ возвращает базу данных в то состояние, в котором она была до начала транзакции $T$. Сагой [9] называется долго-

живущая транзакция, которая состоит из последовательности относительно независимых шагов, в дальнейшем называемых подтранзакциями, $T_1$, $T_2$, $\ldots T_n$. Каждой подтранзакции $T_i$ ставится в соответствие ее компенсационная транзакция $C_i$. Гарантируется, что будет выполнена либо вся последовательность $T_1, T_2, \ldots T_n$, либо какая-нибудь последовательность $T_1, T_2, \ldots T_j, C_j, \ldots C_1$. То есть, выполнится либо вся сага целиком, либо, в случае аварийного завершения подтранзакции, все предшествующие подтранзакции будут компенсированы.

По завершении подтранзакции все заблокированные ею данные освобождаются, делаясь, таким образом, доступными другим параллельно выполняемым транзакциям. При аварийном завершении подтранзакции делается попытка ее повторить, если же по каким-либо причинам это невозможно, производится откат всей саги с помощью последовательного выполнения компенсационных транзакций.

Таким образом, в сагах нарушается принцип изолированности, поскольку промежуточные результаты становятся видимы другим транзакциям до полного завершения саги. Сериализуемость также может быть нарушена, однако сохраняются свойства атомарности и устойчивости.

Заметим, что благодаря нарушению изолированности могут быть частично решены проблемы, возникающие при выполнении долгоживущих транзакций, упомянутые в предшествовавших разделах. Также отметим, что использование саг имеет смысл только в том случае, если получаемые подтранзакции достаточно независимы, поскольку только в таких условиях достаточно велика вероятность ненарушения целостности данных. И, наконец, должна существовать возможность определить компенсационные транзакции для всех подтранзакций (не все действия могут быть компенсированы).

# 5 Модели, базирующиеся на вложенных транзакциях

В данном разделе описывается несколько моделей транзакций, являющихся конкретизациями вложенных транзакций.

## 5.1 Многоуровневые транзакции

Наиболее часто рассматриваемым типом вложенных транзакций являются многоуровневые транзакции [35]. Построение модели многоуровневых транзакций основано на том, что транзакции, как атомарные операции, могут иметь разный смысл на разных уровнях абстракции или на разных уровнях СУБД. В работе [35] описывается механизм поддержки многоуровневых транзакций и разрешение возникающей при этом вложенности. На рис. 1 приведен пример многоуровневой транзакции, осуществляющей перевод заданной суммы с одного счета на другой.

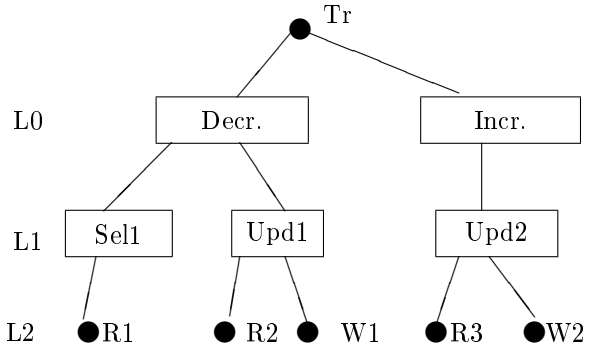


Рис. 1: Многоуровневая транзакция

На логическом уровне *(L0)* такая транзакция включает две операции: снятие суммы с одного счета и начисление на другой *(Incr* и *Decr)*. На уровне манипулирования кортежами *(L1)* снятие суммы со счета включает проверку наличия требуемого количества денег и обновление записи о счете, а начисление - просто обновление счета. И, наконец, на уровне страниц *(L2)* все эти операции преобразуются в последовательность операций чтения и записи.

Таким образом, на нижнем уровне мы имеем дерево вложенных транзакций, единственным новым условием для которого является его сбалансированность. Для отката многоуровневой транзакции на каждом уровне определяются соответствующие компенсационные транзакции, позволяющие производить последовательный поуровневый откат.

## 5.2 Модель DOM

Модель транзакций *DOM* [20] предназначена для поддержки различных приложений в распределенных системах, интегрирующих различные неоднородные компоненты, некоторые из которых, возможно, не являются базами данных (например, файловые системы). В модели поддерживаются два типа вложенных транзакций: *закрытые* и *открытые*. Закрытые вложенные транзакции соответствуют вложенным транзакциям, описанным

выше, а открытые не поддерживают изолированность транзакций верхнего уровня, делая их промежуточные результаты доступными для других транзакций.

Для любой транзакции может быть определена соответствующая ей компенсационная и/или зависимая транзакция, кроме того, любая подтранзакция может быть объявлена существенной или несущественной. В первом случае ее аварийное завершение необходимо влечет аварийное завершение родительской транзакции. Также в некоторых случаях может быть явно задан порядок выполнения подтранзакций. Эти средства позволяют специфицировать различные модели поведения транзакций и зависимостей между ними.

Ниже описывается ряд подтипов транзакций в модели DOM, характеризующихся тем, что в них способ выполнения порожденных транзакций может находиться в зависимости от состояния порождающей.

Основным нововведением является возможность для порожденной транзакции выполняться не в качестве потомка своей родительской транзакции, а в качестве потомка другой транзакции, называемой в этом случае *владельцем* в отличие от *создателя*. Во время выполнения порожденной (подчиненной) транзакции выполнение владельца приостанавливается.

Возможными конкретизациями являются *отложенные* и *отсоединенные* транзакции. Выполнение отложенных транзакций задерживается до завершения родительской (или до определенной в ней точки приостановки), после чего все они начинают выполняться как независимые, однако если в процессе их выполнения будут порождены новые транзакции, их выполнение будет задержано до завершения всех транзакций предыдущего цикла. После завершения всех циклов возможно завершение (или продолжение) порождающей транзакции.

В случае отсоединенных транзакций порожденная транзакция представляется своим собственным деревом и выполняется параллельно со своей истинной родительской транзакцией, если между ними не установлена *причинная зависимость*, и задерживается до ее завершения в другом случае.

## 5.3 Кооперативные транзакции

Кооперативные транзакции были предложены в [25] для использования в системах автоматизированного проектирования. Для таких систем наиболее ярко выражена потребность во взаимодействии между транзакциями, возможность разделять данные. При использовании традиционных транзакций такого рода взаимодействия ограничены требованием выполнять конфликтующие операции в строго определенном порядке, что гарантирует сериализуемость транзакций. В условиях же, когда транзакции выполняются в течении нескольких дней, невозможно и нет необходимости поддерживать их сериализуемость.

Для кооперативных транзакций была предложена их структуризация в виде дерева, называемого *иерархией кооперативных транзакций*. *Внешние узлы* этой иерархии представляют транзакции, ассоциированные с отдельными разработчиками. Внутренние узлы называются *группами транзакций*. Они включают в себя членов группы (потомков), которые могут совместно участвовать в выполнении одной из подзадач проектирования. Таким образом, *кооперативными* в данной модели называются транзакции, являющиеся потомками одного родителя в иерархии. Отметим, что в иерархии может быть больше трех уровней, т.е. допускается несколько уровней вложенности групп.

Поскольку транзакции в системах автоматизированного проектирования являются долгоживущими и открытыми, т.е. в начале выполнения не известны все операции, которые будет выполнять транзакция, иерархия является динамической и может быть изменена в процессц выполнения. В одну группу могут входить одновременно как ее подгруппы, так и отдельные кооперативные транзаккции.

В модели кооперативных транзакций не поддерживается сериализуемость, вместо этого в каждой группе транзакций определяются правила, по которым они могут взаимодействовать. Правила задаются в терминах конечных автоматов. Для каждой кооперативной транзакции определяется набор разрешенных операций и разрешенный порядок выполнения операций в связанных кооперативных транзакциях. Благодаря отсутствию изолированности и достигаемой тесной кооперации удается избегать большей чати прблем, характерных для долгоживущих транзакций.

## 5.4 Модель Flex

Модель Flex [18]была разработана для использования в системах мультибаз данных.

Модель позволяет достигать большей гибкости во время выполнения транзакций. Предполагается, что в процессе выполнения отдельной транзакции должен быть выполнен ряд *задач*. Модель позволяет пользователю для каждой задачи определить множество *функционально эквивалентных*

подтранзакций, успешное выполнение любой из которых достаточно для успешного выполнения задачи в целом. Flex-транзакция устойчива к сбоям в том смысле, что может быть успешно завершена даже в случае аварийного завершения ряда подтранзакций.

Кроме того, имеются возможности для определения *зависимостей* между вложенными транзакциями. Зависимости успешного и аварийного завершения определяют порядок выполнения подтранзакций, а *внешние зависимости* устанавливают зависимость выполнения подтранзакций от событий, не относящихся к транзакции. И, наконец, могут быть определены *компенсационные подтранзакции*, что позволяет достичь желаемой грануляции изолированности.

Flex-транзакции были применены при разработке языка логического программирования VPL [15].

# 6 Модели, базирующиеся на сагах

## 6.1 Расщепляемые транзакции

*Расщепляемые* транзакции [27] были разработаны для поддержки *незавершенных приложений*. Такие приложения характеризуются, во-первых, неопределенным временем выполнения, во-вторых, неопределенным поведением (заранее неизвестны действия, которые будут произведены), и, в-третьих, взаимодействием с параллельно выполняемыми деятельностями. Целью расщепления транзакции является разделение ее на две сериализуемые транзакции с разделение между ними ресурсов. При расщеплении данной транзакции на две транзакции  и $B$, из которых $A$ сериализуется раньше $B$, должны выполняться следующие условия:

1. $AWriteSet \cap BWriteSet \subseteq BWwriteLast$
2. $AReadsSet \cap BWriteSet = \emptyset$
3. $BReadSet \cap AWriteSet = ShareSet$

Первое свойство означает, что транзакция $B$ может модифицировать объекты, модифицированные перед тем транзакцией $A$, $A$ же не может изменять данные, модифицированные $B$. Второе свойство означает, что $A$ не может использовать результаты $B$, а третье - то что $B$, напротив, может использовать результаты $A$.

Все три свойства вместе гарантируют сериализуемость $A$ раньше $B$. В *независимом* случае, когда и $ShareSet$, и $BWriteLast$, они могут быть сериализованы в любом порядке.

## 6.2 Модель ConTract

Модель *ConTract* [29] разработана для поддержки долгоживущих деятельностей в различных нестандартных приложениях. Такая деятельность, или контракт, состоит из нескольких последовательно выполняемых шагов. Основное требование к контракту состоит в том, чтобы в случае аварийного завершения или сбоя системы его выполнение могло бы быть продолжено после восстановления системы, начиная с точки аварийного завершения. Как и в сагах, промежуточные результаты, полученные на каждом шаге, после его завершения становятся доступными параллельно выполняемым контрактам. Основным же отличием от контракта от саги является отсутствие компенсационных транзакций для каждого шага.

## 6.3 Модель системы ACTA

Система ACTA [4] представляет собой средство, дающее возможность формально определять взаимодействия между транзакциями и предполагаемые результаты их воздействия на базу данных. В рамках системы ACTA описываются саги и различные их расширения, наиболее существенные из которых перечислены ниже:

- саги, включающие подтранзакции, которые невозможно компенсировать;
- саги, содержащие вложенные саги;
- саги, включающие зависимые и альтернативные подтранзакции.

## 6.4 S-транзакции

Модель S-транзакций [7] была разработана для поддержки взаимодействий в международных банковских системах. В рассматриваемой области одним из важнейших является требование *локальной автономии*. S-транзакция, являясь по форме вложенной транзакцией, имеет ряд существенных отличий.

Во-первых, в следствие соблюдения принципа автономности отдельных компонент, подсистема, инициализировавшая S-транзакцию, не может однозначно определить, какие именно подсистемы будут участвовать в ее выполнении. Подсистемы, выполняющая подтранзакцию S-транзакции, может делать это любым образом, в случае же

аварийного эавершения она может послать эквивалентный запрос *альтернативной подсистеме*. Таким образом, дерево выполнения S-транзакции недетерминировано и может иметь разную структуру при разный выполнениях одной и той же транзакции.

Вторым важным отличием является то, что подобно сагам, для всех шагов S-транзакции могут быть определены компенсационные подтранзакции. По завершении подтранзакции ее результаты фиксируются в базе данных, и все блокировки снимаются. Таким образом, свойство изолированности поддерживается только для подтранзакций. Успешное или аварийное завершение S-транзакций зависит от аварийного или успешного завершения ее потомков, соответствующие условия задаются как *семантические ограничения*.

# 7 Политранзакции

*Политранзакции* [30] определяются как транзитивное замыкание всех транзакций, допустимых в независимой системе манипулирования данными. Эти транзакции проектировались для использования в распределенных системах с взаимозависимостями между базами данных, то есть таких систем, в которых данные из разных узлов могут быть связаны какими-либо ограничениями целостности.

Не предполагается, что все зависимости между транзакциями известны перед началом выполнения. Дерево политранзакции строится в соответствии со взаимозависимостями между базами данных и предикатами, описывающими целостность данных. Порожденные транзакции выполняются в соответствии с описанными схемами.

Эта модель освобождает от необходимости отслеживать глобальные ограничения целостности в процессе выполнения транзакции и от необходимости модифицировать приложения при изменениях во взаимозависимостях данных.

# 8 Взаимосвязь моделей транзакций

На рис. 2 представлена иерархия классов транзакций.

Заметим, что приведенная схема отражает, в основном, сходство и различия структуры транзакций в разных моделях, и, в меньшей степени, остальные характеристики моделей. Так, одним из базовых различий саг и вложенных транзакций является отсутствие/наличие иерархии вложенных транзакций, и это свойство (или его отсутствие) определяет порядок расположения моделй в иерархии. В меньшей степени учитываются такие характеристики, как наличие компенсационных транзакций, классификация подтранзакций и другие.

# 9 Заключение

За рамками данного обзора остался ряд важных направлений исследований в теории транзакций, в частносте те, которые связаны не только с разработкой новых моделей транзакций, но и с расширенем и обобщением самого понятия транзакций. Среди них следут отметить концепцию *потоков работ*, предложенную и разработанную в [33, 14, 31].

Под потоком работ понимается многозадачная деятельность, которая состоит из нескольких независимых задач, выполняемых в различных системах. Эти системы, как правило, управляются независимо, однако могут содержать связанные и, возможно, пересекающиеся данные. Кроме того, в поток работ включаются ограничения и критерии корректности для его правильного выполнения.

Концепция потоков работ была предложена для организации деятельности в распределенных системах мультибаз данных. Основным фактором, делающим невозможным применение в этих условиях нестандартных моделей транзакций, является необходимость поддержки *автономии* систем. В частности, в случае неуспешного завершения потока работ невозможно произвести откат успешно завершенной транзакции в локальной системе. Таким образом, при организации взаимодействия задач в потоке работ должны использоваться принципы, отличные от используемых в системах с подтранзакциями.

Дальнейшее развитие концепции потоков работ предполагает разработку различных моделей потоков, расширение возможностей описания взаимозависимостей задач и более тонкой настройки на потребности конкретных приложений.

Г

# Список литературы

[1] D. Agrawal and A. E. Abbadi. A Non-Restrictive Concurrency Control for Object Oriented Databases. In *Extending Data Base Technology. Proc. Int. Conf., EDBT-92*, volume

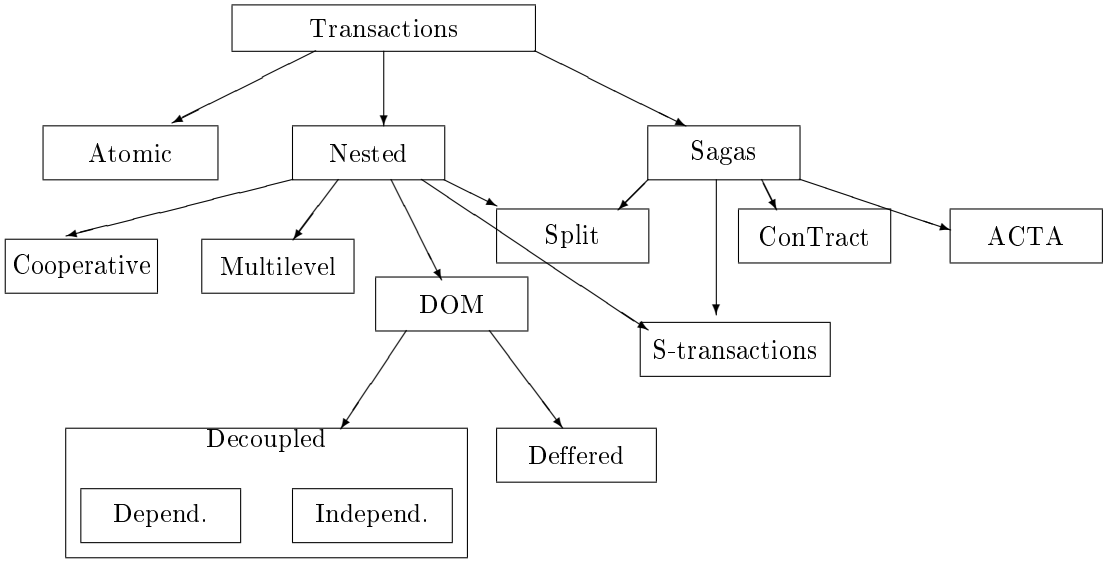


Рис. 2: Иерархия классов транзакций

580 of *Lect. Notes in Comp. Sci.*, pages 469–481, Vienna, Austria, March 1992.

[2] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[3] M. Casanova. *The Concurrency Control Problem of Database Systems, Lecture Notes in Computer Science*. Springer-Verlag, 1981.

[4] P. Chrysanthis and K. Ramamritham. ACTA: A framework for specifying and reasoning about transaction structure and behavior. In *Proceedings of ACM SIGMOD Conference*, 1990.

[5] R. Crus. Data recovery in IBM database 2. *IBM Systems Journal*, 23(2), 1984.

[6] R. Curtis. Informix-Turbo. In *IEEE Compcon Spring'88*, Feb.-March 1988.

[7] F. Eliassen, J. Veijalainen, and H. Tirri. Aspects of transaction modeling for interoperable information systems. Technical report, Report of the COST Project, 1988.

[8] K. Eswaran, J. Gray, R. Lorie, and I. Traiger. The notions of consistency and predicate locks in a database system. *Communications of ACM*, 19(11):624–633, November 1976.

[9] H. Garcia-Molina and K. Salem. SAGAS. In *Proceedings of ACM SIGMOD Conference*, pages 249–259, San Francisco, California, May 1987.

[10] J. Garza and W. Kim. Transaction Management in an Object-Oriented Database System. *Proceedings of ACM SIGMOD Conference*, 1988.

[11] J. Gray. The Transaction Concept: Virtues and Limitations. In *Proceedings of the 7th International Conference on Very Large Data Bases*, pages 144–154, Cannes, France, September 1981.

[12] J. N. Gray. *Operating Systems: An Advanced Course, Lecture Notes in Computer Science*. Springer-Verlag, New York, 1978.

[13] T. Haerder and A. Reuter. Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys*, 15(4), December 1983.

[14] W. Jin, L. Ness, M. Rusinkiewicz, and A. Sheth. Concurrency control and recovery of multidatabase workflows in telecommunication applications. In *Proceedings of ACM SIGMOD Conference*, pages 456–459, Washington, DC, May 1993.

[15] E. Kuehn, F. Puntigam, and A. Elmagarmid. Transaction specification in multidatabase systems based on parallel logic programming. *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, April, 1991.

[16] H. Kung and J. Robinson. On optimistic methods for concurrency control. *ACM TODS*, 6(2):213–226, June 1981.

[17] Y. Leu, A. Elmagarmid, and N. Boudriga. Specification and execution of transactions for advanced database applications. *Inf. Syst.*, 17(2):171–183, 1992.

[18] Y. Leu, A. Elmagarmid, M. Rusinkiewicz, and W. Litwin. Extending the transaction model in a multidatabase environment. In *Proceedings of the 16th International Conference on Very Large Databases*, Brisbane, Australia, August 1990.

[19] E. Levy, H. Korth, and A. Silberschatz. An optimistic commit protocol for distributed transaction management. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, May 1991.

[20] F. Manola and A. Buchmann. Functional/relational object-oriented model for distributed object management. Technical Report TM-0331-11-90-165, GTE Laboratories Inc., December 1990.

[21] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwartz. ARIES: A transaction recovery method supporting fine-granularity locking with partial rollbacks using write-ahead logging. *ACM Trans. on Database Systems*, 17(1):94–162, 1992.

[22] C. Mohan and B. Lindsay. Efficient commit protocols for the tree of distributed transactions. In *ACM Proceedings of 2-nd SIGACT/SIGOPS Symposium on Principles of Distributed Computing*, August 1983.

[23] J. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing*. PhD thesis, MIT Press, Cambridge, MA, 1985.

[24] J. E. B. Moss. Log-based recovery for nested transactions. In *vldb*, pages 427–432, Brighton, England, 1987.

[25] M. Nodine and S. Zdonik. Cooperative transaction hierarchies. *VLDB Journal*, 1(1):41–80, July 1992.

[26] M. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice-Hall, 1990.

[27] C. Pu, G. Kaiser, and N. Hutchinson. Split-transactions for open-ended activities. In *Proceedings of the 14th IEEE International Conference on Data Engineering*, 1988.

[28] D. Reed. *Naming and Synchronization in a Decentralized Computer System*. PhD thesis, MIT, Dep. of Electrical Engineering, 1978.

[29] A. Reuter. Contracts: A Means for Extending Control Beyond Transaction Boundaries. In *Presentation at Third International Workshop on High Performance Systems*, September 1989.

[30] M. Rusinkiewicz and A. Sheth. Polytransactions for managing interdependent data. *IEEE Data Engineering Bulletin*, 14(1), March 1991.

[31] M. Rusinkiewicz and A. Sheth. Specification and execution of transactional workflows. Technical Report UH-CS-93-11, Dept. of Computer Science, University of Houston, November 1993.

[32] P. M. Schwarz, W. Chang, J. C. Freytag, G. M. Lohman, J. McPherson, C. Mohan, and H. Pirahesh. Extensibility in the starburst database system. In *Proceedings of the International Workshop on Object-Oriented Database Systems*, Asilomar, CA, Sept. 1986. IEEE.

[33] A. Sheth and M. Rusinkiewicz. On transactional workflows. *IEEE Data Engineering Bulletin*, 16(2), June 1993.

[34] J. Verhofstad. Recovery techniques for database systems. *ACM Computing Surveys*, 10(2), June 1978.

[35] G. Weikum. Principles and realization strategies of multilevel transaction management. *ACM Transactions on Database Systems*, 16(1):132–180, March 1991.

## Сведения об авторе

Домбровская Генриэтта Ромуальдовна, окончила в 1985 г. математико-механический факультет Ленгосуниверситета, в настоящее время научный сотрудник НИИ математики и механики Санкт-Петербургского университета. Область научных интересов включает различные аспекты систем баз данных, в частности, перспективные модели данных, организация хранения данных, теория транзакций.

Г

# Advanced Transaction Models

**Henrietta Dombrowska**

This paper presents the survey of different advances transaction models proposed through the last ten years by different researchers. The main feature of all these transaction models that they do not obey the ACID properties. The problems due to which traditional transactions have limited applicability in advanced applications are described. The paper presents the classification of different advanced transaction models and describes some of them in detail. The conclusion makes a brief overview of related topics.