Введение в системы управления базами данных
А.Ю.Пушников

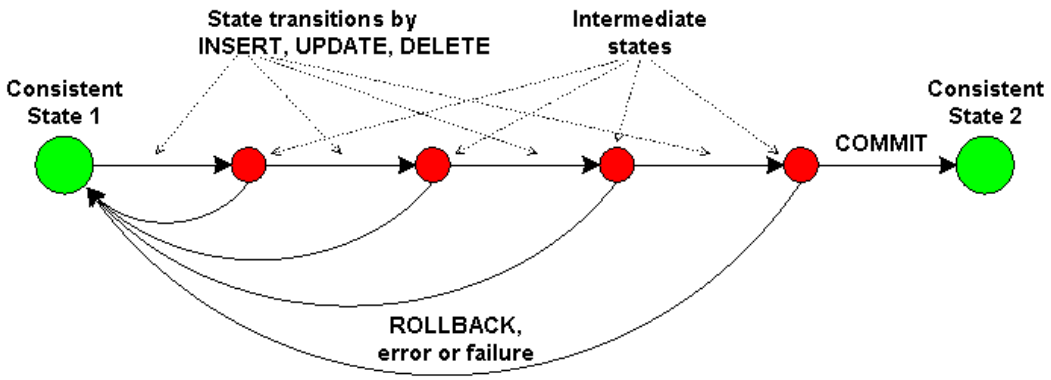
Данное учебное пособие было опубликовано в 1999 г. в двух частях издательством Башкирского государственного университета. Выходные данные: УДК 519.6, ББК 32.973-018.2 Пушников А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 108 с. - ISBN 5-7477-0350-1. В электронной версии - полный текст книги

Web-страница: http://www.citforum.ru/database/dblearn/index.shtml

ТРАНЗАКЦИИ И ЦЕЛОСТНОСТЬ БАЗ ДАННЫХ

В данной и в последующих главах изучается фундаментальное понятие транзакции. Это понятие не входит в реляционную модель данных, т. к. транзакции рассматриваются не только в реляционных СУБД, но и в СУБД других типов, а также и в других типах информационных систем.

Транзакция — это неделимая, с точки зрения воздействия на СУБД, последовательность операций манипулирования данными. Для пользователя транзакция выполняется по принципу "*все или ничего*", т.е. либо транзакция выполняется целиком и переводит базу данных из одного *целостного состояния* в другое *целостное состояние*, либо, если по каким-либо причинам, одно из действий транзакции невыполнимо, или произошло какое-либо нарушение работы системы, база данных возвращается в исходное состояние, которое было до начала транзакции (происходит откат транзакции). С этой точки зрения, транзакции важны как в многопользовательских, так и в однопользовательских системах. В однопользовательских системах транзакции — это логические единицы работы, после выполнения которых база данных остается *в целостном состоянии*. Транзакции также являются *единицами восстановления* данных после сбоев - восстанавливаясь, система ликвидирует следы транзакций, не успевших успешно завершиться в результате программного или аппаратного сбоя. Эти два свойства транзакций определяют атомарность (неделимость) транзакции. В многопользовательских системах, кроме того, транзакции служат для обеспечения *изолированной* работы отдельных пользователей - пользователям, одновременно работающим с одной базой данных, кажется, что они работают как бы в однопользовательской системе и не мешают друг другу.



Пример нарушения целостности базы

Для иллюстрации возможного нарушения целостности базы данных рассмотрим следующий пример:

Пример 1. Пусть имеется система, в которой хранятся данные о подразделениях и работающих в них сотрудниках. Список подразделений хранится в таблице DEPART(Dep_Id, Dep_Name, Dept_Kol), где Dept_Id - идентификатор подразделения, Dept_Name - наименование подразделения, Dept_Kol - количество сотрудников в подразделении. Список сотрудников хранится в таблице PERSON(Pers_Id, Pers_Name, Dept_Id), где Pers_Id - идентификатор сотрудника, Pers_Name - имя сотрудника, Dept_Id - идентификатор подразделения, в котором работает сотрудник:

Dept_Id	Dept_Name	Dept_Kol
1	Кафедра алгебры	3
2	Кафедра программирования	2

Таблица 1 DEPART

Pers_Id	Pers_Name	Dept_Id
1	Иванов	1
2	Петров	2
3	Сидоров	1
4	Пушников	2
5	Шарипов	1

Таблица 2 PERSON

Ограничение целостности этой базы данных состоит в том, что поле Dept_Kol не может заполняться произвольными значениями — это поле должно содержать количество сотрудников, реально числящихся в подразделении.

С учетом этого ограничения можно заключить, что вставка нового сотрудника в таблицу *не может быть выполнена одной операцией*. При вставке нового сотрудника необходимо одновременно увеличить значение поля Dept_Kol:

- Шаг 1. Вставить сотрудника в таблицу PERSON: INSERT INTO PERSON (6, Муфтахов, 1)
- Шаг 2. Увеличить значение поля Dept_Kol: UPDATE DEPART SET Dept=Dept+1 WHERE Dept_Id=1

Если после выполнения первой операции и до выполнения второй произойдет сбой системы, то реально будет выполнена только первая операция и база данных остается в нецелостном состоянии.

Понятие транзакции

Определение 1. ***Транзакция*** — это последовательность операторов манипулирования данными, выполняющаяся *как единое целое* (все или ничего) и переводящая базу данных *из одного целостного состояния в другое целостное состояние*.

Транзакция обладает четырьмя важными свойствами, известными как ***свойства АСИД (ACID - Atomicity, Consistency, Isolation, Durability)***:

- ***(А) Атомарность***. Транзакция выполняется как атомарная операция - либо выполняется вся транзакция целиком, либо она целиком не выполняется.
- ***(С) Согласованность***. Транзакция переводит базу данных из одного согласованного (целостного) состояния в другое согласованное (целостное) состояние. Внутри транзакции согласованность базы данных может нарушаться.
- ***(И) Изоляция***. Транзакции разных пользователей не должны мешать друг другу (например, как если бы они выполнялись строго по очереди).
- ***(Д) Долговечность***. Если транзакция выполнена, то результаты ее работы должны сохраниться в базе данных, даже если в следующий момент произойдет сбой системы.

Транзакция обычно начинается автоматически с момента присоединения пользователя к СУБД и продолжается до тех пор, пока не произойдет одно из следующих событий:

- Подана команда COMMIT WORK (зафиксировать транзакцию).
- Подана команда ROLLBACK WORK (откатить транзакцию).
- Произошло отсоединение пользователя от СУБД.
- Произошел сбой системы.

Команда COMMIT WORK завершает текущую транзакцию и автоматически начинает новую транзакцию. При этом гарантируется, что результаты работы завершенной транзакции фиксируются, т. е. сохраняются в базе данных.

Замечание. Некоторые системы требуют подать явную команду BEGIN TRANSACTION для того, чтобы начать новую транзакцию.

Команда ROLLBACK WORK приводит к тому, что все изменения, сделанные текущей транзакцией, откатываются, т. е. отменяются так, *как будто их вообще не было*. При этом автоматически начинается новая транзакция.

При отсоединении пользователя от СУБД происходит автоматическая фиксация транзакций.

При сбое системы происходят более сложные процессы. Кратко суть их сводится к тому, что при последующем запуске системы происходит анализ выполнявшихся до момента сбоя транзакций. Те транзакции, для которых была подана команда COMMIT WORK, но *результаты работы которых не были занесены в базу данных* выполняются снова (накатываются). Те транзакции, для которых не была подана команда COMMIT WORK, откатываются. Более подробно восстановление после сбоев рассматривается далее.

Свойства АСИД транзакций не всегда выполняются в полном объеме. Особенно это относится к свойству И (изоляция). В идеале транзакции разных пользователей не должны мешать друг другу, т.е. они должны выполняться так, чтобы у пользователя создавалась иллюзия, что он в системе один. Простейший способ обеспечить абсолютную изолированность состоит в том, чтобы выстроить транзакции в очередь и выполнять их строго одну за другой. Очевидно, при этом теряется эффективность работы системы. Поэтому реально одновременно выполняется несколько транзакций (смесь транзакций). Различается несколько уровней изоляции транзакций. На низшем уровне изоляции транзакции могут реально мешать друг другу, на высшем они полностью изолированы. За большую изоляцию транзакций приходится платить большими накладными расходами системы и замедлением работы. Пользователи или администратор системы могут по своему усмотрению задавать различные уровни всех или отдельных транзакций. Более подробно изоляция транзакций рассматривается в следующей главе.

Свойство Д (долговечность) также не является абсолютными свойством, т. к. некоторые системы допускают вложенные транзакции. Если транзакция Б запущена внутри транзакции А, и для транзакции Б подана команда COMMIT WORK, то фиксация данных транзакции Б является условной, т. к. внешняя транзакция А может откатиться. Результаты работы внутренней транзакции Б будут окончательно зафиксированы только если будет зафиксирована внешняя транзакция А.

Ограничения целостности

Свойство (С) - согласованность транзакций определяется наличием понятия согласованности базы данных.

Определение 2. ***Ограничение целостности*** — это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

Примерами ограничений целостности могут служить следующие утверждения:

Пример 2. Возраст сотрудника не может быть меньше 18 и больше 65 лет.

Пример 3. Каждый сотрудник имеет уникальный табельный номер.

Пример 4. Сотрудник обязан числиться в одном отделе.

Пример 5. Сумма накладной обязана равняться сумме произведений цен товаров на количество товаров для всех товаров, входящих в накладную.

Как видно из этих примеров, некоторые из ограничений целостности являются ограничениями реляционной модели данных. Пример 3 представляет ограничение, реализующее целостность сущности. Пример 4 представляет ограничение, реализующее ссылочную целостность. Другие ограничения являются достаточно произвольными утверждениями (примеры 2 и 5). Любое ограничение целостности является *семантическим* понятием, т. е. появляется как следствие определенных свойств объектов предметной области и/или их взаимосвязей.

Определение 3. База данных находится в ***согласованном (целостном) состоянии***, если выполнены (удовлетворены) все ограничения целостности, определенные для базы данных.

В данном определении важно подчеркнуть, что должны быть выполнены не все вообще ограничения предметной области, а только те, которые *определены* в базе данных. Для этого необходимо, чтобы СУБД *обладала* развитыми средствами поддержки ограничений целостности. Если какая-либо СУБД не может отобразить все необходимые ограничения предметной области, то такая база данных хотя и будет находиться в целостном состоянии с точки зрения СУБД, но это состояние не будет правильным с точки зрения пользователя.

Таким образом, согласованность базы данных есть формальное свойство базы данных. База данных не понимает "смысла" хранимых данных. "Смыслом" данных для СУБД является весь набор ограничений целостности. Если все ограничения выполнены, то СУБД считает, что данные корректны.

Вместе с понятием целостности базы данных возникает понятие ***реакции системы на попытку нарушения целостности***. Система должна не только проверять, не нарушаются ли ограничения в ходе выполнения различных операций, но и должным образом реагировать, если операция приводит к нарушению целостности. Имеется два типа реакции на попытку нарушения целостности:

1. *Отказ* выполнить "незаконную" операцию.
2. Выполнение *компенсирующих* действий.

Например, если система знает, что в поле "Возраст_Сотрудника" должны быть целые числа в диапазоне от 18 до 65, то система отвергает попытку ввести значение возраста 66. При этом может генерироваться какое-нибудь сообщение для пользователя.

В противоположность этому, в примере 1 система допускает вставку записи о новом сотруднике (что приводит к нарушению целостности базы данных), но *автоматически* производит компенсирующие действия, изменяя значение поля Dept_Kol в таблице DEPART.

Работу системы по проверке ограничений можно изобразить на следующем рисунке:

Рисунок 1

В некоторых случаях система может не выполнять проверку на нарушение ограничений, а сразу выполнять компенсирующие операции. Действительно, в примере 1 при вставке или удалении сотрудника проверку производить *не нужно*, т. к. результаты ее известны заранее - ограничение *обязательно* будет нарушено. В этом случае необходимо сразу приступать к компенсированию возникшего нарушения.

Классификация ограничений целостности

Ограничения целостности можно классифицировать несколькими способами:

- По способам реализации.
- По времени проверки.
- По области действия.

Классификация ограничений целостности по способам реализации

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

- Декларативная поддержка ограничений целостности.
- Процедурная поддержка ограничений целостности.

Определение 4. ***Декларативная поддержка ограничений целостности*** заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД)

определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

Например, следующий оператор создает таблицу PERSON и определяет для нее некоторые ограничения целостности:

```
CREATE TABLE PERSON
  (Pers_Id INTEGER PRIMARY KEY,
  Pers_Name CHAR(30) NOT NULL,
  Dept_Id REFERENCES DEPART(Dept_Id) ON UPDATE CASCADE ON DELETE CASCADE);
```

После выполнения оператора для таблицы PERSON будут объявлены следующие ограничения целостности:

- Поле Pers_Id образует потенциальный ключ отношения.
- Поле Pers_Name не может содержать null-значений.
- Поле Dept_Id является внешней ссылкой на родительскую таблицу DEPART, причем, при изменении или удалении строки в родительской таблице *каскадно* должны быть внесены соответствующие изменения в дочернюю таблицу.

Средства декларативной поддержки ограничений описаны в стандарте SQL и более подробно рассматриваются ниже.

Определение 5. ***Процедурная поддержка ограничений целостности*** заключается в использовании триггеров и хранимых процедур.

Не все ограничения целостности можно реализовать декларативно. Примером такого ограничения может служить требование из примера 1, утверждающее, что поле Dept_Kol таблицы DEPART должно содержать количество сотрудников, реально числящихся в подразделении. Для реализации этого ограничения необходимо создать триггер, запускающийся при вставке, модификации и удалении записей в таблице PERSON, который корректно изменяет значение поля Dept_Kol. Например, при вставке в таблицу PERSON новой строки, триггер увеличивает на единицу значение поля Dept_Kol, а при удалении строки - уменьшает. Заметим, что при модификации записей в таблице PERSON могут потребоваться даже более сложные действия. Действительно, модификация записи в таблице PERSON может заключаться в том, что мы переводим сотрудника из одного отдела в другой, меняя значение в поле Dept_Id. При этом необходимо в старом подразделении уменьшить количество сотрудников, а в новом - увеличить.

Кроме того, необходимо защититься от неправильной модификации строк таблицы DEPART. Действительно, пользователь может попытаться модифицировать запись об отделе, введя неверное значение поля Dept_Kol. Для предотвращения подобных действий необходимо создать также триггеры, запускающиеся при вставке и модификации записей в таблице DEPART. Триггер, запускающийся при удалении записей из таблицы DEPART не нужен, т. к. уже имеется ограничение ссылочной целостности, каскадно удаляющее записи из таблицы PERSON при удалении записи из таблицы DEPART.

По сути, наличие ограничения целостности (как декларативного, так и процедурного характера) *всегда* приводит к созданию или использованию некоторого программного кода, реализующего это ограничение. Разница заключается в том, где такой код хранится и как он создается.

Если ограничение целостности реализовано в виде триггеров, то этот программный код является просто телом триггера. Если используется декларативное ограничение целостности, то возможны два подхода:

1. При декларировании (объявлении) ограничения текст ограничения хранится в виде некоторого объекта СУБД, а для реализации ограничения используются встроенные в СУБД функции, и тогда этот код представляет собой внутренние функции ядра СУБД.
2. При декларировании ограничения СУБД автоматически генерирует триггеры, выполняющие необходимые действия по проверке ограничений.

Примером использования функций ядра для проверки декларативных ограничений является автоматическая проверка уникальности индексов, соответствующих потенциальным ключам отношений. В качестве другого примера можно привести поддержку ссылочной целостности средствами СУБД ORACLE. Ограничение ссылочной целостности является в ORACLE объектом базы данных, хранящим формулировку этого ограничения. Проверка ограничения выполняется функциями ядра ORACLE со ссылкой на этот объект. Ограничение целостности в этом случае *нельзя модифицировать иначе, как используя декларативные операторы* создания и модификации ограничений.

Примером генерации новых триггеров для реализации декларативных ограничений, может служить система Visual FoxPro. Триггеры, автоматически сгенерированные Visual FoxPro при объявлении ограничений ссылочной целостности, можно посмотреть и даже внести в них изменения, так чтобы они могли выполнять некоторые дополнительные действия.

Если система не поддерживает ни декларативную поддержку ссылочной целостности, ни триггеры (как, например, FoxPro 2.5), то программный код, следящий за корректностью базы данных, приходится размещать в пользовательском приложении (такой ведь код все равно необходим!). Это сильно затрудняет разработку программ и не защищает от попыток пользователей напрямую внести некорректные данные в базу данных. Особенно сложно становится в том случае, когда имеется сложная база данных и множество различных приложений, работающих с ней (например, к базе данных торгового предприятия может обращаться несколько приложений, таких как "Складской учет", "Прием заказов", "Главный бухгалтер" и т.п.). Каждое из таких приложений должно содержать один и тот же код, отвечающий за поддержание целостности базы данных. Особенно весело становится разработчику, когда необходимо внести изменения в логику поддержания целостности. Приходится заменять во всех программах одни и те же места, перекомпилировать все приложения и распространять по рабочим местам новые версии.

Классификация ограничений целостности по времени проверки

По времени проверки ограничения делятся на:

- Немедленно проверяемые ограничения.
- Ограничения с отложенной проверкой.

Определение 6. ***Немедленно проверяемые ограничения*** проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Например, проверка уникальности потенциального ключа проверяется в момент вставки записи в таблицу. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности, обычно откатывается.

Определение 7. ***Ограничения с отложенной проверкой*** проверяется в момент фиксации транзакции оператором COMMIT WORK. Внутри транзакции ограничение может не выполняться. Если в момент фиксации транзакции обнаруживается нарушение ограничения с отложенной проверкой, то транзакция откатывается. Примером ограничения, которое не может быть проверено немедленно является ограничение из примера 1. Это происходит оттого, что транзакция, заключающаяся во вставке нового сотрудника в таблицу PERSON, состоит не менее чем из двух операций - вставки строки в таблицу PERSON и обновления строки в таблице DEPART. Ограничение, безусловно, неверно после первой операции и становится верным после второй операции.

Классификация ограничений целостности по области действия

По области действия ограничения делятся на:

- Ограничения домена
- Ограничения атрибута
- Ограничения кортежа
- Ограничения отношения
- Ограничения базы данных

Ограничения домена

Определение 8. ***Ограничения целостности домена*** представляют собой ограничения, накладываемые только на допустимые значения домена. Фактически, ограничения домена обязаны являться частью определения домена (см. определение домена в гл. 2).

Например, ограничением домена "Возраст сотрудника" может быть условие "Возраст сотрудника не менее 18 и не более 65".

Проверка ограничения. Ограничения домена сами по себе не проверяются. Если на каком-либо домене основан атрибут, то ограничение соответствующего домена становится ограничением этого атрибута.

Ограничения атрибута

Определение 9. ***Ограничение целостности атрибута*** представляют собой ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене. Ограничение атрибута в точности совпадают с ограничениями соответствующего домена. Отличие ограничений атрибута от ограничений домена в том, что ограничения атрибута *проверяются*.

Если логика предметной области такова, что на значения атрибута необходимо наложить дополнительные ограничения, помимо ограничений домена, то такие ограничения переходят в следующую категорию.

Проверка ограничения. Ограничение атрибута является *немедленно проверяемым* ограничением. Действительно, ограничение атрибута не зависит ни от каких других объектов базы данных, кроме домена, на котором основан атрибут. Поэтому никакие изменения в других объектах не могут повлиять на истинность ограничения.

Ограничения кортежа

Определение 10. ***Ограничения целостности кортежа*** представляют собой ограничения, накладываемые на допустимые значения *отдельного* кортежа отношения, и *не являющиеся* ограничением целостности атрибута. Требование, что ограничение относится к *отдельному* кортежу отношения, означает, что для его проверки *не требуется* никакой информации о других кортежах отношения.

Пример 6. Атрибут "Возраст сотрудника" в таблице "Спецподразделение", может иметь дополнительное ограничение "Возраст сотрудника не менее 25 и не более 45", помимо того, что этот атрибут уже имеет ограничение, определяемое доменом - "Возраст сотрудника не менее 18 и не более 65".

Приведенное ограничение кортежа, по сути, является дополнительным ограничением на значения *одного* атрибута. В этом случае допустимы два решения. Можно объявить новый домен "Возраст сотрудника спецподразделения" и тогда ограничение кортежа становится ограничением домена и атрибута, либо рассматривать это ограничение именно как ограничение кортежа. Оба решения имеют свои положительные и отрицательные стороны.

Замечание. Тут имеются некоторые возможности для оптимизации. Формально, при изменении значения данного атрибута необходимо проверить два ограничения - ограничение атрибута и ограничение кортежа. Но в данном случае ограничение кортежа сильнее ограничения атрибута и достаточно проверить только ограничение кортежа. Разумно построенная СУБД могла бы выявлять такие случаи и уменьшать лишнюю работу.

Пример 7. Для отношения "Сотрудники" можно сформулировать следующее ограничение: если атрибут "Должность" принимает значение "Директор", то атрибут "Зарплата" содержит значение не менее 1000$.

Это ограничение связывает два атрибута одного кортежа.

Пример 8. В накладной можно установить следующую взаимосвязь атрибутов - "Цена*Количество=Сумма", связывающую атрибуты "Цена", "Количество", "Сумма".

Данный пример кажется неестественным, т. к. сумма является явно избыточным атрибутом, значение которого просто выводятся из значений других атрибутов. Поэтому кажется, что лучше хранить только два базовых атрибута "Цена" и "Количество", а сумму вычислять во время выполнения запросов по мере необходимости. Так, собственно, требует реляционная теория,

стремящаяся свести избыточность к минимуму. В практике, однако, дело обстоит сложнее. Например, каждая строка реальной накладной может содержать следующие данные о товаре:

Наименование атрибута	**Описание атрибута**	**Базовый ли атрибут**	**Формула для вычислимого атрибута**
Name	Наименование товара	Да	
N	Количество	Да	
P1	Учетная цена товара	Да	
S1	Учетная сумма на все количество		S1 = N*P1
PerSent	Процент наценки на единицу товара	Да	
P2	Наценка на единицу товара		P2 = P1*PerSent/100
S2	Сумму наценки на все количество		S2 = N*P2
P3	Цену товара с учетом наценки		P3 = P1+P2
S3	Сумму на все количество с учетом наценки		S3 = N*P3
NDS	Процент НДС	Да	
P4	Сумма НДС на единицу товара		P4 = P2*NDS/100
S4	Сумма НДС на все количество		S4 = N*P4
P5	Цена товара с НДС		P5 = P3+P4
S5	Сумма на все количество с НДС		S5 = N*P5

Таблица 3 Атрибуты товара

Базовыми, т. е. требующими ввода данных являются всего 5 атрибутов (выделены серым цветом). Все остальные атрибуты вычисляются по базовым. Нужно ли хранить в отношении только базовые атрибуты, или желательно хранить все атрибуты, пересчитывая значения вычислимых атрибутов каждый раз при изменении базовых?

Решение 1. Пусть в отношении решено хранить *только базовые атрибуты*.

Достоинства решения:

- Структура отношения полностью неизбыточна.
- Не требуется дополнительного программного кода для поддержания целостности кортежа.
- Экономится дисковое пространство.
- Уменьшается трафик сети.

Недостатки решения:

- В бухгалтерии для формирования проводок используются, как правило, не базовые, а вычислимые атрибуты. Одни и те же формулы используются во многих местах, поэтому все операторы отбора данных будут содержать одинаковые фрагменты кода с одними и теми же формулами. Имеется риск в разных местах вычислять одни и те же данные по *разным формулам*.
- При изменении логики вычислений (что бывает довольно часто при изменении законодательства), необходимо изменить одни и те же фрагменты кода *во всех местах*, где они встречаются. Это сильно затрудняет модификацию приложений.
- Если возникает нерегламентированный запрос, то человек, формулирующий запрос должен *помнить* все эти формулы.

Решение 2. Предположим, что в отношении решено хранить *все атрибуты*, в том числе и вычислимые.

Достоинства решения:

- Код, поддерживающий целостность кортежа (и содержащий формулы для вычислимых атрибутов), хранится в одном месте, например в триггере, связанном с данным отношением.
- При изменении логики вычислений изменения в формулы требуется внести только в одном месте (в триггере).
- Запросы к базе данных содержат меньше формул и поэтому более просты.
- Легче формулировать нерегламентированные запросы, т. к. в запросе используются атрибуты, имеющие для бухгалтера конкретный смысл.

Недостатки решения:

- При изменении логики расчета надобность в некоторых атрибутах может исчезнуть, зато может появиться потребность в новых атрибутах. Это потребует *перестройки структуры* отношения, что является весьма болезненной операцией для работающей системы.
- Структура отношения становится более *сложной и запутанной*.
- *Увеличивается объем* базы данных.
- *Увеличивается трафик* сети.

Как видим, оба решения имеют свои достоинства и недостатки. Важно то, что программный код, содержащий эти формулы, не исчезает ни в каком из этих решений (да и куда он денется, раз такова предметная область!). Только в одном случае код хранится в одном месте, а в другом может быть "размазан" по всему приложению.

На самом деле данный пример сильно упрощен, т. к. еще одной неприятной особенностью наших бухгалтерий является то, что все расчеты должны вестись с определенной точностью, а именно - до копеек. Возникает проблема округления, а это еще более усложняет формулы для

расчетов цен. Простой пример - вычисление НДС содержит операцию деления, следовательно может приводить к бесконечным дробям типа 15,519999… Такую дробь необходимо округлить до 15.52. Если продается одна единица товара, то это не страшно, но если продается несколько единиц товара, то сумму НДС на все количество можно считать по разным формулам:

1. S4 = N* ROUND(P2*NDS/100) - СНАЧАЛА округлить при вычислении НДС на единицу товара, а ПОТОМ умножить на все количество, или
2. S4 = ROUND(N*P2*NDS/100) - СНАЧАЛА умножить на все количество, а ПОТОМ округлить до требуемого знака.

Лирическое отступление. Автор, как математик по образованию (к.ф.-м.н.), считает, что верной, безусловно, является первая формула. Действительно, вычисляя по первой формуле, мы получим *одну и ту же сумму* НДС независимо от того, продали мы одну партию товара, содержащую 50 единиц, или продали 50 партий по одной единице в каждой. При вычислениях по второй формуле сумма НДС в партии, состоящий из 50 единиц товара *отличается* от суммы НДС по 50 партиям по одной единице товара в каждой. Разработав несколько складских программ, автор получал разные ответы на этот вопрос в разных бухгалтерия, разные ответы на этот вопрос в одной бухгалтерии в разное время, и разные ответы на этот вопрос в разных налоговых инспекциях. В конечном итоге автор пришел к грустному выводу, что для того, чтобы стать бухгалтером, его способностей и образования недостаточно.

Другие решения. Можно попытаться использовать и другие решения, для облегчения разработки и сопровождения в данном случае. Например, можно хранить в базовом отношении только базовые атрибуты, а для работы бухгалтерии использовать заранее подготовленные представления (динамические отношения, задаваемые оператором SQL). Тогда логика расчетов будет храниться в одном SQL-операторе, определяющем это представление. Другим вариантом может быть сохранение формул в виде хранимых процедур и функций базы данных.

Проверка ограничения. К моменту проверки ограничения кортежа должны быть проверены ограничения целостности атрибутов, входящих в этот кортеж.

Ограничение кортежа является *немедленно проверяемым* ограничением. Действительно, ограничение кортежа не зависит ни от каких других объектов базы данных, кроме атрибутов, входящих в состав кортежа. Поэтому никакие изменения в других объектах не могут повлиять на истинность ограничения.

Ограничения отношения

Определение 11. ***Ограничения целостности отношения*** представляют ограничения, накладываемые только на допустимые значения *отдельного* отношения, и *не являющиеся* ограничением целостности кортежа. Требование, что ограничение относится к отдельному отношению, означает, что для его проверки не требуется информации о других отношениях (в том числе не требуется ссылок *по внешнему ключу* на кортежи *этого же* отношения).

Пример 9. Ограничение целостности сущности (см. гл. 3), задаваемое потенциальным ключом отношения, является ограничением отношения, т.к. для его проверки необходимо иметь информацию обо всех кортежах отношения (более точно, обо всех занятых в данный момент значениях потенциального ключа).

Пример 10. Ограничение целостности, определяемые наличием функциональных, многозначных зависимостей и зависимостей соединения, являются ограничениями отношения.

Пример 11. Предположим, что в отношении PERSON (см. пример 1) задано следующее ограничение - в каждом отделе должно быть не менее двух сотрудников. Это ограничение можно сформулировать так - количество строк с одинаковым значением Dept_Id должно быть не меньше 2.

Замечание. Для того чтобы ввести в действие (объявить) это ограничение, необходимо, чтобы в отношение уже были вставлены некоторые кортежи.

Пример 12. Ограничение целостности, определяемое требованием, что некоторая таблица должна быть не пуста, являются ограничениями отношения.

Проверка ограничения. К моменту проверки ограничения отношения должны быть проверены ограничения целостности кортежей этого отношения.

Ограничение отношения может быть как *немедленно проверяемым* ограничением, так и ограничением *с отложенной проверкой*.

Ограничение отношения, являющееся ограничением потенциального ключа (пример 9) является немедленно проверяемым ограничением.

Ограничение, определенное наличием функциональной зависимости атрибутов, также является немедленно проверяемым ограничением.

Ограничения же, определенные многозначной зависимостью или зависимостью соединения, являются ограничениями с отложенной проверкой. Действительно, эти ограничения требуют, чтобы кортежи вставлялись и удалялись *целыми группами* (см. гл. 7). Это невозможно сделать, если выполнять проверку после каждой одиночной вставки или удаления кортежа.

Ограничение в примере 11 кажется немедленно проверяемым. Действительно, можно сразу после вставки или удаления кортежа проверить, выполняется ли ограничение, и, если оно не выполняется, то откатить операцию. Но, однако, в этом случае, невозможно вставить ни один новый кортеж для нового отдела. В новый отдел необходимо вставить сразу не менее двух сотрудников. Таким образом, это ограничение с отложенной проверкой.

Ограничение из примера 12 имеет смысл проверять только при удалении кортежей из отношения. Это ограничение может быть как немедленно проверяемым, так и отложенным.

Ограничения базы данных

Определение 12. ***Ограничения целостности базы данных*** представляют ограничения, накладываемые на значения двух или более связанных между собой отношений (в том числе отношение может быть связано само с собой).

Пример 13. Ограничение целостности ссылок (см. гл. 3), задаваемое внешним ключом отношения, является ограничением базы данных.

Пример 14. Ограничение на таблицы DEPART и PERSON из примера 1 является отношением базы данных, т. к. оно связывает данные, размещенные в различных таблицах.

Проверка ограничения. К моменту проверки ограничения базы данных должны быть проверены ограничения целостности отношений.

Ограничение базы данных может быть как *немедленно проверяемым* ограничением, так и ограничением *с отложенной проверкой*.

Ограничение отношения, являющееся ограничением внешнего ключа, может быть как немедленно проверяемым ограничением, так и отложенным ограничением. Действительно, в простейшем случае, если кортеж X отношения A должен ссылаться на кортеж Y отношения B, то проверку ограничения ссылочной целостности можно производить *сразу* после любой из операций вставки, обновления или удаления в любом из отношений A или B. В более сложном случае, предположим, что кортеж X отношения A должен ссылаться на кортеж Y отношения B, а кортеж Y отношения B должен в свою очередь ссылаться на кортеж X отношения A (циклическая ссылка). Очевидно, что сразу после вставки кортежа X отношение A ссылочная целостность *обязательно* нарушена, т.к. кортежа Y *еще нет* в отношении B. Проверку ссылочной целостности можно провести только после завершения транзакции, состоящей из последовательности операций:

1. вставки кортежа X в отношение A с нулевой ссылкой на отношение B,
2. вставки кортежа Y отношение B со ссылкой на кортеж X отношения A,
3. исправления ссылки в кортеже X с NULL на ссылку на кортеж Y.

Ограничение, приведенное в примере 1, может быть только ограничением с отложенной проверкой.

Реализация декларативных ограничений целостности средствами SQL

Общие принципы реализации ограничений средствами SQL

Стандарт SQL *не предусматривает* процедурных ограничений целостности, реализуемых при помощи триггеров и хранимых процедур. В стандарте SQL отсутствует понятие "триггер", хотя триггеры имеются во всех промышленных СУБД SQL-типа. Таким образом, реализация ограничений средствами конкретной СУБД обладает большей гибкостью, нежели с использованием исключительно стандартных средств SQL.

Стандарт SQL позволяет задавать декларативные ограничения следующими способами:

- Как ограничения домена.
- Как ограничения, входящие в определение таблицы.
- Как ограничения, хранящиеся в базе данных в виде независимых утверждений (assertion).

Допускаются как немедленно проверяемые, так и ограничения с отложенной проверкой. Режим проверки отложенных ограничений можно в любой момент изменить так, чтобы ограничение проверялось:

1. После исполнения *каждого оператора*, изменяющего содержимое таблицы, к которой относится данное ограничение.
2. При завершении *каждой транзакции*, включающей операторы, изменяющие содержимое таблиц, к которым относятся данное ограничение.
3. В *любой промежуточный момент*, если пользователь инициирует проверку.

При определении ограничения указывается тип проверки ограничения - является ли это ограничение ***неоткладываемым*** (***NOT DEFERRED***) или может быть ***откладываемым*** (***DEFERRED***). Во втором случае можно задать процедуру по умолчанию: *проверять немедленно* или *проверять по завершении транзакции*. Таким образом, можно определить потенциально откладываемое ограничение, которое по умолчанию проверяется немедленно. В любой момент режим проверки такого ограничения можно изменить на отложенный и наоборот. Режим проверки может быть изменен для одного ограничения или сразу для всех потенциально откладываемых ограничений. Если ограничение определено как неоткладываемое, то тип такого ограничения *изменить нельзя* и ограничение всегда проверяется немедленно.

Элементы процедурности все же присутствуют в стандарте SQL в виде так называемых ***действий, исполняемых по ссылке*** (***referential triggered actions***). Эти действия определяют, что будет происходить при изменении значения родительского ключа, на который ссылается некоторый внешний ключ. Эти действия можно задавать независимо для операций обновления (ON UPDATE) или для операций удаления (ON DELETE) записей в родительском отношении. Стандартом SQL определяется 4 типа действий, исполняемых по ссылке:

- ***CASCADE***. Изменения значения родительского ключа автоматически приводят к таким же изменениям связанного с ним значения внешнего ключа. Удаление кортежа в родительском отношении приводит к удалению связанных с ним кортежей в дочернем отношении.
- ***SET NULL***. Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения NULL.

- ***SET DEFAULT***. Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения, принятые по умолчанию для этих ключей.
- ***NO ACTION***. Значения внешнего ключа не изменяются. Если операция приводит к нарушению ссылочной целостности (появляются "висящие" ссылки), то такая операция не выполняется.

Как видно, действия, исполняемые по ссылке, фактически являются встроенными в СУБД триггерами. Действия типа CASCADE, SET NULL и SET DEFAULT являются компенсирующими операциями, вызывающимися при попытке нарушить ссылочную целостность.

Синтаксис ограничений стандарта SQL

Понятие ограничения используется во многих операторах определения данных (DDL).

Ограничение check::=
CHECK *Предикат*

Ограничения таблицы ::=
[**CONSTRAINT** *Имя ограничения*]
{
{**PRIMARY KEY** (*Имя столбца*.,..)}
| {**UNIQUE** (*Имя столбца*.,..)}
| {**FOREIGN KEY** (*Имя столбца*.,..) **REFERENCES** *Имя таблицы* [(*Имя столбца*.,..)]
[*Ссылочная спецификация*]}
| { *Ограничение check* }
}
[*Атрибуты ограничения*]

Ограничения столбца::=
[**CONSTRAINT** *Имя ограничения*]
{
{**NOT NULL**}
| {**PRIMARY KEY**}
| {**UNIQUE**}
| {**REFERENCES** *Имя таблицы* [(*Имя столбца*)] [*Ссылочная спецификация*]}
| { *Ограничение check* }
}
[*Атрибуты ограничения*]

Ссылочная спецификация::=
[**MATCH** {**FULL** | **PARTIAL**}]
[**ON UPDATE** {**CASCADE** | **SET NULL** | **SET DEFAULT** | **NO ACTION**}]
[**ON DELETE** {**CASCADE** | **SET NULL** | **SET DEFAULT** | **NO ACTION**}]

Атрибуты ограничения::=
{**DEFERRABLE** [**INITIALLY DEFERRED** | **INITIALLY IMMEDIATE**]}
| {**NOT DEFERRABLE**}

Ограничение типа CHECK. Ограничение типа CHECK содержит предикат, могущий принимать значения TRUE, FALSE и UNKNOWN (NULL). Примеры предикатов различного вида приведены в главе 5. Ограничение типа CHECK может быть использовано как часть описания домена, таблицы, столбца таблицы или отдельного ограничения целостности - ASSERTION. Ограничение считается нарушенным, если предикат ограничения принимает значение FALSE.

Пример 15. Пример ограничения типа CHECK:

CHECK (Salespeaple.Salary IS NOT NULL) OR (Salespeaple.Commission IS NOT NULL)

Данное ограничение утверждает, что каждый продавец должен иметь либо ненулевую зарплату, либо ненулевые комиссионные.

Пример 16. Еще пример ограничения типа CHECK:

CHECK EXIST(SELECT * FROM Salespeaple)

Данное ограничение утверждает, что список продавцов не может быть пустым.

Ограничения таблицы и ***ограничения столбца***. Ограничения таблицы и ограничения столбца таблицы входят как часть описания соответственно таблицы или столбца таблицы. Ограничение таблицы может относиться к нескольким столбцам таблицы. Ограничение столбца относится только к одному столбцу таблицы. Любое ограничение столбца можно описать как ограничение таблицы, но не наоборот.

Ограничения таблицы или столбца могут иметь наименования, при помощи которого в дальнейшем можно отменять это ограничение или менять время его проверки.

Ограничение PRIMARY KEY. Ограничение PRIMARY KEY для таблицы или столбца означает, что группа из одного или нескольких столбцов образуют потенциальный ключ таблицы. Это означает, что комбинация значений в PRIMARY KEY должна быть уникальной для каждой строки таблицы. Дублированные значения или значения, содержащие NULL, будут отвергнуты. Для одной таблицы может быть определено единственное ограничение PRIMARY KEY. В терминах стандарта SQL это называется первичным ключом таблицы.

Ограничение UNIQUE. Ограничение UNIQUE для таблицы или столбца означает, что группа из одного или нескольких столбцов образуют потенциальный ключ таблицы, в котором допускаются значения NULL. Это означает, что две строки, содержащие одинаковые и не равные NULL-значения, считаются нарушающими уникальность и не допускаются. Две строки, содержащие NULL-значения *считаются различными* и допускаются. Для одной таблицы может быть определено несколько ограничений UNIQUE.

Замечание. С точки зрения реляционной модели данных, ограничение типа UNIQUE не определяет потенциальный ключ, т.к. потенциальный ключ не должен содержать NULL-значений.

Ограничения FOREIGN KEY и REFERENCES. Ограничение FOREIGN KEY… REFERENCES… для таблицы и ограничение REFERENCES… для столбца определяют внешний ключ таблицы. Ограничение REFERENCES… для столбца определяет простой внешний ключ, т.

е. ключ, состоящий из одной колонки. Ограничение FOREIGN KEY… REFERENCES… для таблицы может определять как простой, так и сложный внешний ключ, т. е. ключ, состоящий из нескольких колонок таблицы. Столбец или группа столбцов таблицы, на которую ссылается внешний ключ, должна иметь ограничения PRIMARY KEY или UNIQUE. Столбцы, на которые ссылается внешний ключ, должны иметь тот же тип данных, что и столбцы, входящие в состав внешнего ключа. Таблица может иметь ссылку на себя. Ограничение внешнего ключа нарушается, если значения, присутствующие во внешнем ключе, не совпадают со значениями соответствующего ключа родительской таблицы ни для одной строки из родительской таблицы. Операции, приводящие к нарушению ограничения внешнего ключа, отвергаются. Как должны совпадать значения внешнего ключа и ключа родительской таблицы, а также, какие действия необходимо выполнить при изменениях ключей в родительской таблице, описаны ниже в ссылочной спецификации.

Ограничение NOT NULL. Ограничение NOT NULL столбца не допускает появления в столбце NULL-значений.

Ссылочная спецификация. Ссылочная спецификация определяет характеристики внешнего ключа таблицы.

Предложение MATCH {FULL | PARTIAL}. Предложение MATCH FULL требует полного совпадения значений внешнего и первичного ключей. Предложение MATCH PARTIAL допускает частичное совпадение значений внешнего и первичного ключей. Предложение MATCH может быть также пропущенным. Для случая MATCH PARTIAL в дочерней таблице могут появиться строки, имеющие значения внешнего ключа, *неуникально* совпадающие со значениями родительского ключа. Т. е. одна строка дочерней таблицы может иметь неуникальные ссылки не несколько строк родительской таблицы. Это очень сильно отличается от реляционной модели данных, и это отличие провоцируется допущением NULL-значений. Чтобы рассмотреть различные варианты совпадений внешнего и родительского ключей, рассмотрим следующий пример.

Пример 17. Пусть имеется две таблицы:

X	*Y*
1	*Aa*
1	*Bb*
2	*Cc*
2	*Dd*
3	*Ee*
3	*Ff*

Таблица 4 таблица A (Родительская)

Z	X	Y
1	1	Aa
2	1	Null
3	Null	Cc
4	Null	Null
5	4	Gg

Таблица 5 Таблица B (Дочерняя)

Таблица A имеет первичный ключ (X, Y). Таблица B имеет первичный ключ Z, и внешний ключ (X, Y), ссылающийся на первичный ключ таблицы A. Различные варианты совпадения строк дочерней таблицы B со строками родительской таблицы A приведены ниже:

	Колонки X и Y таблицы B допускают null-значения	**Колонки X и Y таблицы B не допускают null-значений**
MATCH отсутствует	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - допустима, не совпадает ни с чем. 3 строка - допустима, не совпадает ни с чем. 4 строка - допустима, не совпадает ни с чем. 5 строка - не допустима.	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - не допустима. 3 строка - не допустима. 4 строка - не допустима. 5 строка - не допустима.
MATCH FULL	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - не допустима. 3 строка - не допустима. 4 строка - допустима, не совпадает ни с чем. 5 строка - не допустима.	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - не допустима. 3 строка - не допустима. 4 строка - не допустима. 5 строка - не допустима.
MATCH PARTIAL	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - допустима, неуникально совпадает с 1 и 2 строками таблицы A. 3 строка - допустима, уникально совпадает с 3 строкой таблицы A. 4 строка - допустима, не совпадает ни с чем. 5 строка - не допустима.	1 строка - допустима, совпадает с 1 строкой таблицы A. 2 строка - не допустима. 3 строка - не допустима. 4 строка - не допустима. 5 строка - не допустима.

Предложение MATCH игнорируется, если все столбцы внешнего ключа имеют ограничения NOT NULL.

Предложения ON UPDATE и ON DELETE. Предложения ON UPDATE и ON DELETE определяют действия, исполняемые по ссылке. Действия, исполняемые по ссылке, в основном описаны выше в этой главе. Сложности в понимании того, как выполняются эти действия, возникают если установлено MATCH PARTIAL и колонки, входящие в состав внешнего ключа, допускают NULL-значения. Подробно эти действия с учетом возможных сложностей описаны в [9].

Атрибуты ограничения. Атрибуты ограничения определяют, в какой момент проверяются ограничения. Ограничение может быть определено как NOT DEFERRABLE (неоткладываемое) или DEFERRABLE (откладываемое). Если атрибуты ограничения не указаны, то по умолчанию принимается NOT DEFERRABLE.

Если ограничение определено как NOT DEFERRABLE (***неоткладываемое***), то ограничение всегда проверяется *сразу после выполнения каждого оператора* INSERT, UPDATE или DELETE, которые могут привести к нарушению ограничения.

Если ограничение определено как DEFERRABLE (***откладываемое***), то ограничение *может иметь два режима проверки* - немедленно после выполнения операции или в конце транзакции. Режим проверки может быть изменен в любой момент внутри транзакции командой SET CONSTRAINTS. При определении ограничения можно указать начальный режим проверки INITIALLY DEFERRED (начально отложенное) или INITIALLY IMMEDIATE (начально немедленно проверяемое).

Синтаксис операторов SQL, использующих ограничения

Стандарт SQL описывает следующие операторы, в которых может быть использованы ограничения:

- CREATE DOMAIN - создать домен
- ALTER DOMAIN - изменить домен
- DROP DOMAIN - удалить домен
- CREATE TABLE - создать таблицу
- ALTER TABLE - изменить таблицу
- DROP TABLE - удалить таблицу
- CREATE ASSERTION - создать утверждение
- DROP ASSERTION - удалить утверждение
- COMMIT WORK - зафиксировать транзакцию
- SET CONSTRAINTS - установить момент проверки ограничений

CREATE DOMAIN *Имя домена* [**AS**] *Тип данных*

[**DEFAULT** *Значение по умолчанию*]
[*Имя ограничения*] *Ограничение check* [*Атрибуты ограничения*]

Этот оператор задает домен, на основе которого можно определять колонки таблиц. Т. к. имя колонки, которая будет основана на этом домене заранее неизвестно, то в ограничении CHECK

домена для ссылки на значение этого домена используется ключевое слово VALUE. В конкретной таблице СУБД заменит слово VALUE на имя колонки таблицы.

Пример 18. Приведенный ниже оператор создает домен Salary на основе целочисленного типа данных, причем значения из этого домена не могут принимать неположительные значения (но могут принимать значение NULL!). По умолчанию это ограничение проверяется немедленно, но может быть и отложенным:

```
CREATE DOMAIN Salary AS integer CHECK (VALUE > 0) DEFERRABLE INITIALLY IMMEDIATE
```

ALTER DOMAIN *Имя домена*
{**SET DEFAULT** *Значение по умолчанию*}
| {**DROP DEFAULT**}
| {**ADD** [*Имя ограничения*] *Ограничение check* [*Атрибуты ограничения*]}
| {**DROP CONSTRAINT** *Имя ограничения*}

Этот оператор изменяет имеющийся домен. Стандарт запрещает вносить несколько изменений одной командой ALTER DOMAIN. Т. е. если требуется удалить ограничение CHECK и добавить значение по умолчанию, то придется выполнить два оператора ALTER DOMAIN.

DROP DOMAIN *Имя домена* **CASCADE** | **RESTRICT**

Этот оператор уничтожает имеющийся домен. Если указана опция RESTRICT, то домен не уничтожается, если имеются ссылки на него из столбцов таблиц. Если указана опция CASCADE, то происходят следующие действия:

- Тип данных домена передается столбцам, основанным на этом домене.
- Если столбец не имеет значения по умолчанию, а для домена значение по умолчанию определено, то оно становится значением по умолчанию для столбца.
- Все ограничения домена становятся ограничениями столбца.

CREATE TABLE *Имя таблицы*
({*Определение столбца* | [*Ограничение таблицы*]}.,..)

Определение столбца::=
Имя столбца {*Имя домена* | *Тип данных* [*Размер*]}
[*Ограничение столбца…*]
[**DEFAULT** *Значение по умолчанию*]

Этот оператор (синтаксис приведен не полностью - пропущены опции создания временных таблиц) создает таблицу базы данных. В таблице обязано быть не менее одного определения столбца. В таблице может быть определено несколько ограничений (в том числе и ни одного).

Каждый столбец должен иметь имя и быть определен на некотором типе данных или на некотором домене. Ограничения домена становятся ограничениями столбца. Кроме того, столбец может иметь дополнительные ограничения. Если домен имеет значение по умолчанию и в определении столбца определено значение по умолчанию, то значение для столбца перекрывает значение для домена.

Пример 19.

```
CREATE TABLE Salespeaple
(Salespeaple_Id Id_Nums PRIMARY KEY,
Fam CHAR(20) NOT NULL,
Im CHAR(15),
BirthDate DATE,
Salary Salary_Domain DEFAULT 1000,
City_Id INTEGER REFERENCES City ON UPDATE CASCADE ON DELETE RESTRICT,
District_Id INTEGER,
CONSTRAINT AltKey UNIQUE(Fam, Im, BirthDate),
CHECK (City_Id IS NOT NULL OR District_Id IS NOT NULL),
FOREIN KEY District_Id REFERENCES District ON UPDATE CASCADE ON DELETE RESTRICT)
```

Этот оператор создает таблицу Salespeaple с колонками (Salespeaple_Id, Fam, Im, BirthDate, Salary, City_Id, District_Id) и следующими ограничениями:

- Колонка Salespeaple_Id наследует все ограничения домена Id_Nums. Кроме того, эта колонка образует первичный ключ таблицы (следовательно, не допускает NULL-значений).
- Колонка Fam не допускает NULL-значений.
- Колонка Salary наследует все ограничения домена Salary_Domain. Кроме того, эта колонка имеет значения по умолчанию1000.
- Колонка City_Id является внешним ключом, ссылающимся на первичный ключ таблицы City. При изменении первичного ключа в таблице City соответствующие значения внешнего ключа в таблице Salespeaple будут каскадно изменены. При удалении строки из таблицы City будет выполняться проверка, нет ли ссылок на удаляемую строку из таблицы Salespeaple. Если такие ссылки найдутся, то операция удаления в таблице City будет отвергнута.
- Колонка District_Id также является внешним ключом, ссылающимся на первичный ключ таблицы District. Этот внешний ключ, в отличие от предыдущего, определен как ограничение таблицы. Действия, определенные по ссылке аналогичны предыдущим.
- Колонки (Fam, Im, BirthDate) образуют альтернативный ключ таблицы. Это ограничение имеет наименование AltKey.
- Колонки City_Id и District_Id не могут одновременно принимать NULL-значения (хотя каждая из них по отдельности допускает использование NULL-значений).

ALTER TABLE *Имя таблицы*
{**ADD** [**COLUMN**] *Определение столбца*}
| {**ALTER** [**COLUMN**] *Имя столбца* {**SET DEFAULT** *Значение по умолчанию* | **DROP DEFAULT**}}
| {**DROP** [**COLUMN**] *Имя столбца* **RESTRICT** | **CASCADE**}
| {**ADD** *Ограничение таблицы*}
| {**DROP CONSTRAINT** *Имя ограничения* **RESTRICT** | **CASCADE**}

Этот оператор позволяет изменять имеющуюся таблицу. В таблице можно удалять или добавлять колонки и/или ограничения. Кроме того, для колонок можно менять значение по умолчанию.

DROP TABLE *Имя таблицы* **CASCADE** | **RESTRICT**

Этот оператор позволяет удалять имеющуюся таблицу. Вместе с таблицей удаляются и ограничения, определенные для этой таблицы.

Если указан параметр RESTRICT, то таблица удаляется только если нет никаких ссылок на эту таблицу в других ограничениях или представлениях.

Если указан параметр CASCADE, то удаляются также и все объекты, ссылающиеся на эту таблицу.

CREATE ASSERTION *Имя утверждения*
Ограничение check
[*Атрибуты ограничения*]

Этот оператор позволяет создавать утверждения - т. е. ограничения, не являющиеся частью определения домена, колонки или таблицы.

Предикат CHECK, входящий в определение утверждения, может быть достаточно сложным и содержать ссылки на несколько таблиц.

Пример 20.

```
CREATE ASSERTION Check_Pay CHECK (Salespeaple.Salary IS NOT NULL) OR
 (Salespeaple.Commission IS NOT NULL) DEFERRABLE INITIALLY IMMEDIATE
```

DROP ASSERTION *Имя утверждения*

Этот оператор позволяет удалять имеющееся утверждение.

COMMIT WORK

Этот оператор фиксирует транзакцию. При этом проверяются все отложенные до конца транзакции ограничения. Если одно из ограничений не выполняется, то транзакция откатывается.

SET CONSTRAINT {*Имя ограничения*... | **ALL**}
{**DEFERRED** | **IMMEDIATE**}

Этот оператор позволяет во время выполнения транзакции менять момент проверки всех (ALL) или некоторых ограничений. Этот оператор действует только на ограничения, определенные как DEFERRABLE (потенциально откладываемые). Если ограничение A находилось в состоянии IMMEDIATE (немедленно проверяемое), то оператор SET CONSTRAINT A DEFERRED переводит его в состояние DEFERRED (с отложенной проверкой) и тогда все операции, потенциально могущие нарушить это ограничение, будут выполняться без проверки. Проверка будет произведена в конце транзакции или в момент подачи команды SET CONSTRAINT A IMMEDIATE.

Выводы

Транзакция — это неделимая, с точки зрения воздействия на СУБД, последовательность операций манипулирования данными, выполняющаяся по принципу "***все или ничего***", и переводящая базу данных из одного ***целостного состояния*** в другое целостное состояние.

Транзакция обладает четырьмя важными свойствами, известными как ***свойства АСИД***:

- ***(А) Атомарность***.
- ***(С) Согласованность***.
- ***(И) Изоляция***.
- ***(Д) Долговечность***.

База данных находится в ***согласованном состоянии***, если для этого состояния выполнены все ***ограничения целостности***.

Ограничение целостности — это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

Ограничения целостности классифицируются несколькими способами:

- ***По способам реализации***.
- ***По времени проверки***.
- ***По области действия***.

По способам реализации различают:

- ***Декларативную*** поддержку ограничений целостности - средствами языка определения данных (DDL).
- ***Процедурную*** поддержку ограничений целостности - посредством триггеров и хранимых процедур.

По времени проверки ограничения делятся на:

- ***Немедленно проверяемые ограничения***.
- ***Ограничения с отложенной проверкой***.

По области действия ограничения делятся на:

- ***Ограничения домена***.
- ***Ограничения атрибута***.
- ***Ограничения кортежа***.
- ***Ограничения отношения***.
- ***Ограничения базы данных***.

Стандарт языка SQL поддерживает только декларативные ограничения целостности, реализуемые как:

- Ограничения домена.
- Ограничения, входящие в определение таблицы.
- Ограничения, хранящиеся в базе данных в виде независимых утверждений (assertion).

Проверка ограничений допускается как после выполнения каждого оператора, могущего нарушить ограничение, так и в конце транзакции. Во время выполнения транзакции можно изменить режим проверки ограничения.