*Мы продолжали проповедовать Свободу человека. Но, забыв о Человеке, мы определили нашу Свободу как некую безнаказанность, при которой дозволены любые поступки, лишь бы они не причиняли вреда другому. А это лишено всякого смысла, ибо нет такого поступка, который не затрагивал бы другого человека.*

*Антуан де Сент-Экзюпери «Военный лётчик»*

# Ключ или отмычка

**Содержание**



## Введение

Любые рассуждения о достоинствах или недостатках тех или иных ключей должны опираться на требования предметной области и те ограничения, которые она налагает. Сравнения ключей должны в первую очередь, происходить в плоскости полного/неполного соответствия схемы базы данных предметной области, простоты или сложности достижения этого соответствия, требования введения дополнительных механизмов обеспечивающих данное соответствие. Если две схемы базы данных обладают разной степенью соответствия предметной области, то сравнение любых других их характеристик не будет корректным.

Схема базы данных создаётся на основе инфологической модели – формального описания предметной области. Как и любая другая модель, инфологическая модель отбрасывает те атрибуты сущностей, которые не играют в ней значимой роли. Например, при приёме на работу нового служащего нас может не интересовать его рост или цвет волос, и эти атрибуты могут быть отброшены, в то время как уровень образования или профессиональные навыки могут быть важны. Может так получиться, что среди несущественных атрибутов окажутся и те атрибуты, которые однозначно идентифицируют экземпляры данной сущности. В результате, сущность, в рамках инфологической модели, может не иметь атрибутов для формирования ключа-кандидата. Однако реляционная модель требует, чтобы каждый кортеж любого отношения имел уникальный идентификатор.

На реализации схемы базы данных могут сказаться ограничения, присущие той или иной реляционной СУБД. Например, СУБД может накладывать ограничение на размер и/или количество атрибутов входящих в отношение или индекс; различные СУБД могут поддерживать разные наборы типов атрибутов и т.п. Несомненно, эти особенности СУБД скажутся и на схеме базы данных. Важно и то, что различные реляционные СУБД по-своему реализуют или не реализуют совсем те или иные механизмы, например, поддержание проверок, каскадные модификации, триггера, хранимые процедуры и т.п. В результате реализация

логики взаимодействия с данными, для конкретной предметной области, может существенно различаться у разных СУБД. Может даже оказаться, что придерживаться какой-то одной стратегии выбора первичного ключа невозможно из-за сложности механизмов поддержания логики данных на конкретной СУБД. И, не смотря на то, что современные реляционные СУБД имеют достаточно развитые механизмы поддержания логики данных, тем не менее, этих механизмов может оказаться недостаточно или их реализация и поддержка будет слишком сложной и дорогой.

Проектирование реляционных баз данных включает в себя стадию определения функциональных зависимостей между атрибутами каждой сущности. На этой стадии, в частности, определяются ключи-кандидаты – такие наборы атрибутов, которые могут служить уникальным идентификатором каждого экземпляра данной сущности (кортежа отношения). Например, серия и номер паспорта являются уникальным идентификатором документа или, говоря другими словами, приведённые атрибуты определяют все прочие атрибуты данного документа, такие как, фамилию, имя, отчество владельца паспорта, место и дату выдачи документа и т.п.

Знание функциональных зависимостей необходимо для последующего выполнения процесса нормализации, который представляет собой декомпозицию исходных сущностей на основании определённых правил. Поскольку функциональные зависимости существуют объективно, вне зависимости от наших знаний предметной области, то их исследование также объективно необходимо для построения наиболее адекватной модели предметной области в виде схемы базы данных.

Неадекватность отражения предметной области в виде схемы базы данных приводит к тому, что информация, хранимая в базе данных, может быть рассогласована с информацией существующей в реальном мире. Можно рассмотреть простой пример. Допустим, что для идентификации отношения «Служащие» выбраны ключевые атрибуты паспорта: серия и номер. Тогда нам потребуется ответить на вопросы о том, что произойдёт, если служащий сменит паспорт или предъявит иной, допускаемый в данной ситуации законом, документ? Если смена паспорта потребует изменение ключевых атрибутов, то правильно ли были изначально определены сущности предметной области, их атрибуты и функциональные зависимости? Предположим, что ранее в платёжных документах отмечалась информация, соответствующая атрибутам старого паспорта. Если мы сменим значения атрибутов (серия и номер паспорта) на новые во всех сущностях (таблицах базы данных), то, очевидно, что документы, выданные до смены паспорта, будут искажены.

Искажения связаны с тем, что информация в документах хранимых в базе данных изменится, следуя требованиям ссылочной целостности. Однако информация в реальных документах сохранится в первоначальном виде. Теперь информация в базе данных не будет соответствовать реальной информации. Не изменять информацию в связанных сущностях тоже невозможно, поскольку в этом случае произойдёт нарушение ссылочной целостности. Ранее выданные документы будут ссылаться на служащего, которого уже не существует в базе данных.

Получается замкнутый круг: нельзя изменить атрибуты, так как они ранее были использованы в платёжных документах, а эти документы изменять нельзя, но, тем не менее, изменить атрибуты всё же надо, ибо реальная смена паспорта произошла. В чём состоит ошибка? В том, что паспорт был отождествлён с его владельцем, а, как позже выяснилось, у одного служащего может быть более одного паспорта. Таким образом, на этапе проектирования не было учтено, что служащие и паспорта представляют собой две разные сущности, связанные отношением один-ко-многим.

Приведённый пример показывает, что информационная неполнота схемы базы данных, её не соответствие предметной области может послужить причиной многих проблем, возникающих при эксплуатации. С полнотой схемы базы данных связана и проблема выбора ключей. Дискуссия о преимуществах тех или иных видов ключей ведётся уже не одно десятилетие, и она втянула на свою орбиту самых видных исследователей реляционной модели. Существует множество различных видов ключей и по каждому виду дано много определений. Наиболее полно перечень видов ключей представлен в работе Joe Celko [D&D], там же даны определения каждого вида ключа. В большой цитате из этой книги, приведённой ниже, даны определения различных ключей:

«Первичный ключ (primary key) – это ключ, который может быть выбран АБД (администратором базы данных) для представления таблицы. В оригинальной реляционной модели Кодда таблица должна иметь первичный ключ. Однако большинство теоретиков реляционной модели признают факт, что ключ есть ключ, и логически здесь нет ничего особенного относительно первичного ключа.

В SQL-машинах, тем не менее, первичный ключ может обладать особенными свойствами. Машины баз данных часто создают предопределённые соединения в специальных индексных структурах для первичного ключа. Первичный ключ также по умолчанию используется ссылочными ограничениями. Так могут быть другие специальные индексные структуры, которые поддерживают ограничения между таблицами. Таблицей, на которую ссылаются, и таблицей, которая ссылается.

Вторичный ключ (secondary key) или ключ-кандидат (candidate key) – комбинация столбцов, отличная от комбинации, составляющей первичный ключ. Эти ключи могут сопровождаться уникальными (UNIQUE) и не пустыми (NOT NULL) ограничениями в SQL. Существует различие в SQL между уникальным ограничением (UNIQUE) и ограничением первичного ключа. Ограничение первичного ключа всегда NOT NULL у всех его столбцов, в то время как ограничение UNIQUE позволяет одно и только одно значение NULL в каждом столбце, если не наложено иного ограничения.

Простой ключ (simple key) – ключ, содержащий только один атрибут. В идеале мы бы хотели самый короткий и самый простой из возможных типов данных, чтобы операции объединения (JOIN) производились быстрее. С этой точки зрения нам больше всего подходит тип INTEGER или другой числовой тип данных, который имеет аппаратную поддержку для операций над ним и сравнений.

Сложный (compound) или составной ключ (composite key) – ключ, состоящий более чем из одного атрибута.

Суррогатный ключ (surrogate key) – ключ создаваемый внутри базы данных, который не содержит в себе никакой информации. Суррогатный ключ часто используют вместо значимого сложного ключа, который является слишком громоздким, чтобы использоваться в реальной базе данных. Идея состоит в том, что система поддерживает его, и он никогда не показывается пользователю.

Искусственный ключ (artificial key) – ключ созданный внутри базы данных или пользователем с помощью некоторой процедуры, который не содержит информацию сам по себе. Искусственный ключ используется для создания идентификаторов строк, когда сущность должна быть описана полностью, чтобы однозначно идентифицировать конкретный элемент.

Интеллектуальный ключ (intelligent key) также называемый «значимый ключ» (meaningful key) или «естественный ключ» (natural key); ключ, в который включены значимые атрибуты и, таким образом, он содержит информацию.

Естественный ключ (natural key) – это частный случай интеллектуального ключа, который тоже является естественным и неизменяемым в приложении.

Например, идентификация области по её физическим координатам или человека по его ДНК или отпечаткам пальцев. Главной его характеристикой является то, что, по сущности, вы можете определить значение ключа.

Майк Паккард (Mike Packard) описал различие между интеллектуальным и естественным ключами, отмечая, что интеллектуальный ключ содержит перегружаемые (overloaded) значения, в то время как естественный ключ содержит единственное значение.

Суперключ (super key) – сложный ключ с большим числом столбцов, чем необходимо для того, чтобы быть уникальным идентификатором. Это не всегда плохо, так как избыточность может оказаться полезной пользователю.

Перекрывающиеся ключи (overlapping keys) – сложные ключи, которые имеют один или более общих столбцов».

Основные баталии велись по вопросу выбора между суррогатными и интеллектуальными ключами. Возможно, если бы др. Е. Кодд определил свою позицию по данному вопросу более чётко, то споры бы утратили остроту, но этого не произошло. Вот как об этом пишет Крис Дейт (Chris Date) [CD WENE] «в статье про RM/T Кодд также впервые использует идею суррогатов - т.е. определяемых системой идентификаторов. (Снова эта идея подается только в связи с предлагаемыми семантическими расширениями, хотя нет никаких оснований, не использовать ее в базовой модели, и в пользу этого имеется много веских аргументов). Однако, к сожалению, в статье утверждается, что суррогаты должны быть скрыты от пользователей - очевидное нарушение приведенного ранее в этой статье определения реляционной базы данных, в котором говорится, что все данные в базе данных должны быть доступны (авторизованным) пользователям. На самом деле, можно было привести тот аргумент, что сокрытие суррогатов нарушает собственный Информационный Принцип Кодда, который устанавливает, что вся информация в базе данных должна явно представляться в терминах отношений и никак иначе».

Обсуждению вопроса выбора между интеллектуальными и суррогатными ключами и посвящена данная статья. Безусловно, выбор в пользу интеллектуального ключа требует очень хорошего знания предметной области. Сам интеллектуальный ключ может состоять из набора атрибутов различных типов в разных сочетаниях. В отличие от этого суррогатный ключ представляет собой некоторый универсальный атрибут, как правило, целочисленного типа, который не зависит ни от предметной области, ни, тем более, от структуры отношения, которое он идентифицирует. Эта универсальность суррогатного ключа и позволяет называть данный атрибут не ключом, но отмычкой.

**Почему выбирают суррогатные ключи**

Устойчивость суррогатных ключей

Наиболее весомым аргументом в пользу выбора суррогатных ключей часто называют проблему неустойчивости интеллектуальных ключей и, соответственно, большей устойчивости суррогатных ключей. Интеллектуальный ключ жёстко привязан к предметной области, и все флуктуации предметной области могут задевать значения, хранимые в интеллектуальном первичном ключе. И это действительно так. Здесь возможны две ситуации. Первая состоит в том, что одно значение, хранимое в интеллектуальном ключе, замещается другим значением. В такой ситуации, возможно, потребуется выполнить каскадные (CASCADE) обновления или заменить значения во внешних ключах на значение, назначенное по умолчанию (DEFAULT), пустое (NULL) значение или вообще запретить обновление (NO ACTION). Все эти ситуации описаны и закреплены в стандарте SQL 92. Так при объявлении внешнего ключа можно указать требуемые реакции на

две основные операции: обновление и удаление. В следующем примере поле FLD2 ссылается на поле FLD1 той же таблицы, то есть является внешним ключом:

```
CREATE TABLE EXAMPLE_TABLE1 (
FLD1   CHAR(10) NOT NULL PRIMARY KEY,
FLD2   CHAR(10) REFERENCES EXAMPLE_TABLE1 (FLD1)
…);
```

При удалении кортежа, все ссылающиеся на него другие кортежи, будут теперь вместо ссылки иметь значение NULL, но при обновлении атрибута FLD1 все, ссылающиеся на данный кортеж, другие кортежи также примут новое значение. СУБД, поддерживающие требование стандарта, полностью снимают с разработчика базы данных проблему поддержания ссылочной целостности при операциях обновления первичных ключей и удаления кортежей. Безусловно, операции, сопровождающиеся каскадными изменениями, будут выполняться дольше операций, не требующих каскадных изменений. Применение суррогатных ключей позволяет избежать каскадных изменений (UPDATE). Действительно в данном случае изменяется атрибут, не входящий в первичный ключ, а, следовательно, в ссылочных отношениях делать изменения не потребуется. Против каскадных удалений суррогатные ключи бессильны. Удаление кортежа в одном отношении потребует всех связанных кортежей того же и/или других отношений.

С другой стороны, изменения значений ключевых полей требуется не так часто. В примере с паспортными данными, который рассматривался в начале статьи показано, что получение нового паспорта не обязательно сопряжено с изменением значений ключевых полей. Новый паспорт реально ссылается на того же человека, что и старый паспорт. Мало того, возможны ситуации, когда человек имеет одновременно оба паспорта, например, считалось, что старый паспорт потерян, но он был найден спустя некоторое время после того, как уже был выдан новый паспорт. Если бы атрибуты серии и номера паспорта были включены в сущность «Служащие» и по ним бы проводилась идентификация служащего, то проблему двух паспортов решить было бы не так просто. Но разумное разделение сущностей «Служащие» и «Документы» исключает потребность в каскадных обновлениях.

Другим примером аналогичной ситуации может служить изменение названий, выбранных в качестве первичного ключа. Предположим, что в базе данных существует таблица «Фирмы-разработчики», которая имеет в качестве первичного ключа атрибут «Краткое название». Пусть в этой таблице один из кортежей имел ключ со значением «AT&T», но через какое-то время данная фирма была переименована в «Lucent». На первый взгляд нам потребуется сделать изменения не только в первичном ключе данной таблицы, но провести каскадные изменения внешних ключей, имеющих в своём составе данное поле. Под такие изменения, частности попадает и таблица «Позиции заказа», которая связывает заказ с моделями микросхем различных фирм-разработчиков. Однако такое изменение недопустимо, вследствие того, что изменится содержание отчётных документов, таких как, оплаченные заказы. Какой выход можно предложить в данной ситуации? Надо добавить новое поле к таблице «Фирмы-разработчики», и это поле должно ссылаться на эту же таблицу по её первичному ключу. Назовём это поле «Старое название». Теперь можно внести новый кортеж, где в качестве названия указать фирму «Lucent», и в поле «Старое название» записать значение «AT&T». Это решение не только не потребует каскадных изменений внешних ключей, но и сохранит отчётные документы в первозданном виде.

Применение суррогатного ключа в подобной ситуации не спасло бы положения, поскольку изменение значения в поле «Краткое название» таблицы «Фирмы-разработчики» привело бы к тому, что автоматически изменились бы и отчётные документы.

Безусловно, изменение значений первичного ключа, связанное с ошибками ввода потребует проведения каскадных изменений во внешних ключах. Но это едва ли не единственный случай, когда подобные изменения действительно необходимы и оправданы. Но, с другой стороны, это хороший стимул для того, чтобы продумать систему мер от таких ошибок, в частности, автоматизировать ввод важнейшей информации, входящей в интеллектуальные ключи. В конце концов, суррогатные ключи тоже не защищают от ошибок ввода, они просто делают исправления менее болезненными.

Вторая ситуация, связанная с изменениями интеллектуального ключа, состоит в том, что меняется тип атрибута, входящего в ключ. Например, возможна ситуация, когда вместо целого атрибута потребовалось использовать строковый атрибут, или размера атрибута оказалось недостаточно для сохранения значения и атрибут требуется увеличить. Такие ситуации возможны и они требуют гораздо более сложных операций, чем простое каскадное изменение значений во внешних ключах. Прежде всего, потребуется изменить структуру внешних ключей, которые ссылаются на первичный ключ, в котором произошли изменения. Операция по изменению нескольких отношений, в свою очередь, требует сохранения и переформатирования хранимой информации.

Таким образом, изменение структуры первичного ключа превращается в сложную операцию по модификации части базы данных. Применение суррогатных ключей в данной ситуации позволяет значительно упростить решение задачи, поскольку потребуется изменить поля только одного отношения, не задевая внешних ключей других отношений. Возникает резонный вопрос о том, можно ли решение об использовании суррогатных ключей считать наилучшим из всех возможных решений. Но для ответа на данный вопрос необходимо сначала разобраться в том, является ли данное решение адекватным проблеме. Проблема изменения структуры первичного ключа может быть вызвана двумя факторами. Первый фактор – ошибка проектирования. Второй фактор – реальные изменения, происходящие в предметной области.

*Ошибки проектирования*

Ошибки проектирования, в результате которых в последствии приходится изменять структуру первичного ключа, являются достаточно распространённым явлением. В этой связи интересно рассмотреть ещё один фрагмент из работы Joe Celko [D&D], где приводится разбор примера предложенного К. Дейтом. «Крис Дейт и другие сторонники реляционных баз данных приводят доводы в пользу искусственных ключей и против интеллектуальных, что изменения в данных входящих в интеллектуальный ключ будут разрушать базу данных (см. глава 30, «Не включайте информацию в первичный ключ!» в RELATIONAL DATABASE WRITINGS 1989-1991 by Chris Date, Addison-Wesley, 1992, ISBN 0-201-54303-6). Использованная в одном из примеров Дейта Автомобильная компания Zitzi применяет часть номеров от 0000 до 4999 для запчастей, которые покупаются у внешних поставщиков, и номера от 5000 до 9999 для запчастей, сделанных у себя. Однажды компания купит 5001 запчасть у внешнего поставщика, и, цитирую Дейта: «Любая программа, которая полагалась на факт, что покупные запчасти имеют номера меньше чем 5000, потерпит неудачу». Я думаю, что это хороший пример его точки зрения. Автомобильная Компания Zitzi положила не ключевые атрибуты в их первичный ключ. Поставщики, несомненно, могут меняться, поэтому источник запчасти не должен быть включён в этот ключ. Возможно, номер запчасти должен относиться к сборке или подпроцессу сборки, которой запчасть принадлежит. Это согласуется с промышленным стандартом для кодов частей самолётов. Эти коды являются составными, они становится всё длиннее и длиннее по мере того, как мы углубляемся в более мелкие части. Идеально, когда некоторый физический элемент используется в более чем одном подпроцессе сборки, ключ одинаково заканчивается в обоих случаях. Например, xxxxxx-005 и yyyyyy-005 могут

означать: «винт номер пять используется в Frammistat» и «винт номер пять используется в Freppometer», которые есть раздельные номерные части, но тот же самый физический элемент».

Какой вывод можно сделать из этого примера? Вполне очевидно, что при проектировании базы данных К. Дейтом были сделаны ошибочные допущения, которые привели к ситуации, когда необходимо изменить атрибуты первичного ключа. Какое из двух решений является более правильным: введение суррогатного ключа или приведение базы данных в состояние адекватное требованиям предметной области? Может ли суррогатный ключ решить проблему несоответствия схемы базы данных своей предметной области?

Ответы на эти вопросы не столь просты, как того, наверное, хотелось бы. И, к сожалению, некоторые проектировщики баз данных пользуются «простым» решением, основанном на применении суррогатного ключа, вместо тщательного исследования предметной области. Решение, предлагаемое К. Дейтом для процитированного выше примера, состоит во введении суррогатного ключа и не требует дополнительного исследования предметной области. Но предположим, что интеллектуальный ключ всё-таки существует. Существование интеллектуального ключа является объективным фактором, который не зависит от наших представлений о предметной области. Не менее объективным фактором являются и функциональные зависимости, в частности, зависимости неключевых атрибутов от интеллектуального ключа. Но в таком случае, решение, основанное на введении суррогатного ключа, порождает транзитивную зависимость:

суррогатный ключ $\Rightarrow$ интеллектуальный ключ $\Rightarrow$ неключевые атрибуты

(здесь знак $\Rightarrow$ обозначает термин «определяет»)

Наличие транзитивной зависимости нарушает третью нормальную форму (3НФ). Таким образом, бездумное использование суррогатных ключей, приводит к нарушению нормализации, а, следовательно, и к аномалиям. При этом проектировщики будут не готовы к появлению аномалий, поскольку они не исследовали функциональные зависимости между атрибутами. Следовательно, аномалии не будут предотвращены, и могут повлиять на операции с хранимой в базе данных информацией.

Трудно представить, что К. Дейт, автор множества работ в области реляционной модели, призывал к денормализации базы данных. Но такой результат отчасти становится неизбежным, если следовать его совету, а не заниматься тщательным исследованием предметной области. Отсюда можно сделать и другой вывод, что использование суррогатных ключей, как средства исправления ошибок проектирования, не только не решает проблему, но может серьёзно усугубить её решение. Или, говоря другими словами, суррогатные ключи не являются способом понижения сложности предметной области и простота их использования весьма обманчива. Как следствие, применение суррогатных ключей в данном случае, равнозначно попытке страуса избежать опасности, спрятав голову в песок.

Хотелось бы отметить тот любопытный факт, что сторонники суррогатных ключей часто используют грубые ошибки проектирования в качестве доказательства правильности своей позиции. Типичным примером этого положения может служить, например, учебное пособие по MS SQL 7.0 Ю. Тихомирова [ЮТ MSS]. В разделе «Первичные ключи. Что выбрать в качестве первичных ключей для каждой из этих таблиц?» он пишет: «Рассмотрим таблицу Authors. Среди её столбцов очевидным кандидатом на первичный ключ является name. Авторов всегда можно различить по имени. Однако использовать этот столбец в качестве первичного ключа достаточно проблематично по нескольким причинам. Во-первых, значения в столбце name состоят из имени и фамилии автора». А чуть ниже, в этом же разделе, следует такая сентенция «И эта комбинация будет удовлетворять нас,

пока таблица не разрастётся до такой степени, что в ней не появятся авторы с одинаковыми именами и фамилиями». Для читателя так и осталось загадкой, как можно различать авторов по имени и фамилии, если существуют авторы с одинаковыми именами и фамилиям. Наверное, было бы правильнее при постановке задачи сказать, что сущность Authors не имеет ключей-кандидатов в списке атрибутов, рассматриваемых Ю. Тихомировым, а, следовательно, суррогатный ключ был выбран на безальтернативной основе. Но такой выбор никак нельзя соотнести с необходимостью использования суррогатных ключей при наличии альтернативы в виде интеллектуальных ключей. В результате же столь неудачного доказательства складывается впечатление, что автор данной книги никогда не посещал публичных библиотек, не искал книги по каталогам. К сожалению, подобные примеры далеко не редкость.

*Изменения предметной области*

Предметная область является частью реального мира и, кроме того, она и сама может быть достаточно сложной системой, способной к развитию, а, следовательно, к изменениям. Не замечать и не учитывать этого нельзя. Однако изменения предметной области предполагают и изменения в базах данных, схемы которых моделируют предметную область, её сущности и атрибуты этих сущностей. Изменения сущностей и их атрибутов приводят, в конечном итоге, к необходимости модификации схемы баз данных. При этом базы данных могут содержать значительные объёмы информации, и изменения в таких базах данных достаточно сложны. Это вполне оправдывает попытки найти некоторое универсальное средство, способное минимизировать затраты на перестройку баз данных. Рассмотрение данного вопроса, хотелось бы начать с самой предметной области и её связей с окружающим миром.

Предметная область состоит из сущностей, которые, как правило, можно разделить на две категории. К первой категории относятся сущности, изменение которых происходит в рамках предметной области. Ко второй категории относятся сущности, изменение которых происходит вне предметной области. Можно рассмотреть в качестве примера систему управления предприятием. Предприятие может в достаточно широких пределах менять свою организационную структуру управления, учётную политику и многие другие параметры. Эти структуры и параметры находят своё отражение и в виде таблиц реляционных баз данных. Вместе с перечисленными изменениями возможны и изменения в схемах баз данных. С другой стороны, предприятие не существует обособленно, оно активно контактирует с другими предприятиями и государственными органами. И эти внешние по отношению к данному предприятию организации могут нуждаться в некоторой структурированной информации, связанной с деятельностью данного предприятия. Предоставляемая информация тоже может менять свою структуру, что, безусловно, скажется и на структурах информации представленных в базах данных.

Возможное изменение структур информации в рамках предметной области должно контролироваться проектировщиком ещё при разработке баз данных. В этом нет ничего сверхъестественного. Для того чтобы можно было предполагать будущие изменения надо опять же хорошо представлять себе предметную область. Любая сложная система базируется на относительно небольшом количестве базовых процессов, которые всегда остаются неизменными, поскольку они определяют существо предметной области. Однако представления этих процессов могут меняться в достаточно широких пределах не только для различных групп пользователей, но, с течением времени, для одной и той же группы пользователей. И здесь, безусловно, важно уметь отличать информацию о процессах от информации о представлении этих процессов.

Известный стандарт ANSI/SPARC устанавливал трёхуровневую модель: физическая модель, концептуальная модель и, наконец, уровень представлений. Такое разделение является важным, прежде всего, в методологическом плане. Информация о базовых процессах не может меняться по объективным причинам, независящим от чьей-то воли. Фактически изменение этой информации сопряжено с изменением базовых процессов. А изменение базовых процессов, в свою очередь, означает изменение существа самой предметной области. В то время как представления этой информации или формы обслуживания основных процессов являются динамичными. Основные ошибки проектирования, как правило, связаны с тем, что проектировщики не делают разделения между этими видами информации, что в результате приводит к необходимости сложных изменений при изменении той или иной отчётной формы. Подобного рода ошибки появляются вследствие недостаточного знания проектировщиками своей предметной области. При тщательном разделении информации структуры, связанные с представлением информации могут дополняться новыми атрибутами или могут удалять атрибуты, ставшие ненужными. Однако эти операции практически никогда не затрагивают структуры ключевых полей, а, следовательно, не приводят к тем сложностям, с которыми связана операция по изменению структуры ключевых полей.

В качестве иллюстрации, к изложенному выше, можно привести примеры из области автоматизации управления предприятием. Служащие предприятия идентифицируются по табельному номеру, они заключают договора, которые тоже идентифицируются по своему номеру. Вся основная работа со служащими ведётся относительно их договора: переводы с одного места работы на другое, изменение окладов, режимов работ и т.п. А также относительно их табельного номера: определение категории льгот для данного служащего, поощрения и взыскания, послужной список и порядок повышения квалификации. Соответственно такие атрибуты, как табельный номер, номера договоров, номера отделов, идентификаторы режимов работ и т.п. по своей сути остаются неизменными. Это важное требование ещё и потому, что информация о каждом служащем, работавшем на предприятии должна сохраняться десятки лет, так, чтобы её можно было восстановить по первому требованию. С другой стороны такая информация, как образование, состав семьи, адрес местожительства, документы и т.п., является вспомогательной, и она может меняться произвольно часто. Однако эти изменения не могут отразиться на основных информационных структурах. Аналогичные рассуждения можно привести и по другим сущностям, например, таким оборудование, материалы, технологические потоки и операции. Но, рассуждая таким образом, можно придти к закономерному выводу о том, что изменчивость структуры первичных ключей внутренних (для конкретной предметной области) информационных структур является проблемой тех проектировщиков, которые по каким-то субъективным причинам не исследовали свою предметную область должным образом.

Перейдёт теперь к информации, изменение которой происходит вне предметной области, но эти изменения отражаются на схемах баз данных. Внешняя, по отношению к предметной области, сфера неподконтрольна проектировщикам и с изменениями, происходящими здесь, просто приходится считаться. При этом, однако, надо отметить два важных момента. Внешняя информация менее подвержена изменениям, по сравнению с внутренней, для конкретной предметной области, информацией. Продолжая рассмотрение системы автоматизации управления предприятием, можно отметить, что формы и способы взаимодействия предприятия с государством меняются значительно реже, чем взаимодействие между подсистемами внутри предприятия. И это закономерно, так как изменение связей государство – предприятие не может быть локальным, оно коснётся не одного, а многих предприятий. Однако столь масштабные изменения могут происходить только при весьма значительных аргументах, поскольку они требуют много средств и времени. С другой стороны, подобные изменения не могут

носить косметический характер именно в силу своих высоких затрат, и они связаны, как правило, с существенными переменами внешней сферы изменяющими, в том числе, и конкретную предметную область. Такие изменения, как правило, невозможно «спрятать» с помощью суррогатных ключей.

Второй момент, который необходимо отметить в связи с изменениями внешней сферы, состоит в том, что в хорошей системе взаимодействие системы с внешним миром осуществляется через интерфейсный слой. Соответственно все изменения внешнего мира влияют в основном на этот интерфейс и значительно реже на существо предметной области. Соответственно, для любой относительно сложной системы необходимо потратить определённые усилия на то, чтобы сделать этот интерфейс максимально гибким. Эти усилия окупятся, когда система сможет пережить внешние потрясения без каких-либо существенных модификаций.

**Связи между таблицами на основе суррогатных ключей**

Существует мнение, что в силу компактности суррогатных ключей, связь, построенная на их основе, наиболее эффективна. Но перед тем как обсуждать этот вопрос, хотелось бы рассмотреть проблему связи на основе суррогатных ключей более обобщённо. В этой связи, очень показательно высказывание К. Дейта о существенности [CD AC]: «Теперь (наконец!) я поясню смысл понятия существенности. Конструкция, связанная с данными, существенна, если ее утрата вызывает потерю информации - вполне точно, я понимаю это так, что перестанет достигаться некоторое отношение. Например, в реляционном варианте базы данных отделов и сотрудников все конструкции данных в этом смысле существенны. Аналогично, в начальной сетевой версии все конструкции (все строки, все столбцы и все связи) тоже существенны. Но в пересмотренном сетевом варианте строки и столбцы продолжают быть существенными, а связь - нет. Нет такой информации, которую можно было бы получить из сети и нельзя было бы получить только из строк и столбцов; вообще отсутствует логическая потребность в связи.

Замечание: Некоторые люди думают, что имеет место обратная ситуация - существенна именно связь, а не внешний ключ. Но это противоречит тому, что поскольку некоторые строки и столбцы должны быть существенными и ничего больше не требуется, то зачем нам нужно что-то еще?

Теперь (наконец) я могу пояснить, в чем состоит принципиальное различие между реляционной базой данных и базой данных другого рода, скажем, сетевой. В реляционной базе данных единственной существенной конструкцией данных является отношение. В других базах данных должна присутствовать по меньшей мере одна дополнительная существенная конструкция данных (такая, как существенная связь). Если бы это было не так, то база данных была бы по существу реляционной с явной демонстрацией некоторых путей доступа. (Не требуется, чтобы пользователи явно применяли эти пути доступа; единственный вопрос состоит в том, почему мы показываем эти пути доступа, а другие не делают этого.) И именно эти дополнительные существенные конструкции данных в основном (но не полностью) приводят к сложности нереляционных баз данных».

Данная цитата весьма интересна, если рассматривать её с точки зрения применения суррогатных ключей, как альтернативной связи между отношениями, вместо естественной связи, построенной на интеллектуальных ключах. Если предположить, что первичный ключ делают суррогатным с целью упрощения связи, и он по определению не несёт в себе никакой полезной информации, то, призывая к использованию суррогатных ключей («Не включайте информацию в первичный ключ!»), Крис Дейт фактически призывает… к отходу от реляционной модели. Как будет показано далее, использование суррогатных вместо интеллектуальных ключей действительно усложняет базу данных и запросов.

*Усложнение базы данных*

Усложнение базы данных можно рассматривать с двух позиций. Во-первых, усложняются структуры базы данных, во-вторых, усложняется логика поддержания достоверности информации, хранимой в базе данных.

Усложнение структуры базы данных вполне закономерно. Предположим, что сущность имеет интеллектуальный ключ, но помимо него был введён ещё и суррогатный ключ. Ввод дополнительного атрибута в каждую таблицу сам по себе усложняет структуру базы данных. Но, чтобы не нарушать 3НФ (о чём говорилось ранее) и ограничения предметной области необходимо поддерживать уникальность и интеллектуального, и суррогатного ключей одновременно. Соответственно, количество структур, с помощью которых поддерживается уникальность (как правило, уникальных индексов), увеличивается. И так по каждому отношению, обладающему интеллектуальным ключом.

Усложнение логики базы данных не столь очевидно и на этом вопросе стоит остановиться более подробно. Интеллектуальный ключ, являясь внешним ключом, содержит полезную информацию, и эта информация может быть использована в рамках ссылочного отношения, содержащего данный внешний ключ. Например, есть общероссийский классификатор товаров, где каждый вид товаров или товарная группа (классификатор имеет иерархическое строение) обладают уникальным номером. Предположим, что у нас есть отношение, которое фиксирует только продуктовые товары, ссылаясь при этом на общероссийский классификатор. При использовании интеллектуального внешнего ключа можно на поле внешнего ключа наложить дополнительное ограничение диапазона значений. Теперь проектировщики могут быть спокойны за то, что в отношение продуктовых товаров не попадёт иной товар, поскольку код другого товара будет вне диапазона значений, отведённого для продуктовых товаров. То есть, при попытке ввода товара, не относящегося к продуктовой группе, произойдёт нарушение ограничения наложенного на внешний ключ. Можно ли сделать подобное ограничение на суррогатных ключах? Нет, без введения дополнительных механизмов, нельзя, и вот почему. При создании отношения «Продуктовые товары» ещё не известно, какие значения будут присвоены суррогатному ключу у тех или иных категорий товаров, и, тем более, неизвестно будут ли продуктовые товары представлены диапазоном значений или некоторой произвольной последовательностью номеров, так, что между номерами продуктовых товаров окажутся номера совсем иных товаров. Как следствие, в случае использования суррогатных ключей придётся реализовывать какие-то сложные логические конструкции, которые бы поддерживали достоверность данных в отношении «Продуктовые товары».

В качестве другого примера использования значений внешнего ключа можно рассмотреть машиностроительную базу данных. Здесь довольно много информации представлено в виде справочников, в том числе, есть и справочники различных коэффициентов. Теперь представим, что в базе данных хранится некоторая сборочная единица, при расчёте которой использован некоторый коэффициент, описанный в одной из справочных таблиц.

```
TABLE COEFFICIENTS (
COEF            DECIMAL(8.2) NOT NULL PRIMARY KEY,
DESCRIPTION   VARCHAR(255) NOT NULL,
        …);
TABLE NODES (
PART_NO        CHAR(15) NOT NULL PRIMARY KEY,
COEF             DECIMAL(8.2) NOT NULL REFERENCES COEFFICIENTS,
FIELD_A         INTEGER NOT NULL,
FIELD_B         DECIMAL(9.5) NOT NULL,
…
```

```
CHECK   ((FIELD_A * COEF) < FIELD_B),
        …);
```

Очевидно, что в случае использования интеллектуальных ключей, логика поддержания достоверности информации реализуется просто и эффективно. Реализовать подобное с помощью суррогатных ключей опять же не столь легко, так как нам потребуется сначала найти нужный коэффициент в таблице коэффициентов прежде, чем его можно будет использовать в выражении CHECK. Подобных примеров полезного использования значений интеллектуальных внешних ключей можно привести достаточно много из самых разных предметных областей.

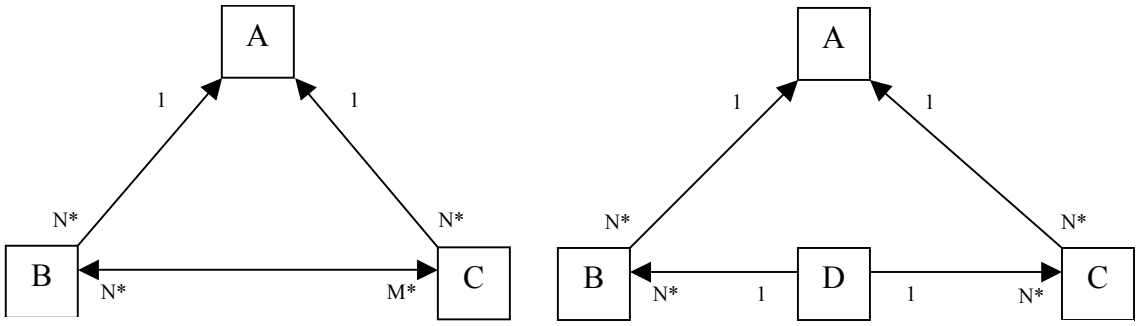
*Коллизии*

Усложнение логики базы данных, построенной на суррогатных ключах, связано с тем, что в таких базах данных необходимо преодолевать коллизии, вызванные применением суррогатных ключей. Существо коллизий состоит в том, что базы данных, построенные на суррогатных ключах, не обеспечивают той же достоверности информации, что и базы данных, построенные на интеллектуальных ключах, без введения дополнительных механизмов. При этом данные механизмы не всегда просты, а их введение далеко не всегда очевидно.

Определим коллизии как аномалии данных, которые возникают при введении суррогатных ключей. Ниже рассмотрены две разновидности таких коллизий, и они проиллюстрированы примером из реальной предметной области. Но возможные коллизии не исчерпываются перечисленными примерами даже в случае бинарных связей между отношениями. При повышении арности связей между отношениями могут быть и иные, более сложные, коллизии. Вопрос исследования коллизий является очень интересным и требует отдельного рассмотрения, которое возможно будет представлено в виде отдельной работы.

Итак, определение коллизии базируется на утверждении, что $K^s$ не тождественен $K^i$, где $K^i$ – представляет собой интеллектуальный ключ некоторого отношения, а $K^s$ – представляет собой суррогатный аналог.

Рассмотрим Δ-схему




*A Концептуальная модель* *B. Физическая модель*

*Рис. 1. Дельта-схема*

Из схемы, представленной на рис. 1. видно, что отношения B и C ссылаются на отношение A. Связи между отношениями A, B и C являются детерминированными, то есть, внешние ключи, ссылающиеся на отношение A, входят составной частью в первичные ключи отношений B и C. Пусть интеллектуальный первичный ключ отношения A состоит из одного или более атрибутов и обозначается $K_a$. Интеллектуальный первичный ключ отношения B $K_b$ будет состоять из внешнего ключа $K_a$ и одного или более атрибутов отношения B, обозначаемых, как $\Delta K_b$. Аналогично, интеллектуальный первичный ключ отношения C $K_c$ будет состоять из внешнего ключа Ka и одного или более атрибутов отношения

C, обозначаемых, как $\Delta K_c$. Тогда интеллектуальные ключи отношений B и C можно записать в виде:

$$K_b \supset K_a \bigcup \lambda K_b$$

$$K_c \supset K_a \bigcup \lambda K_c$$

тогда интеллектуальный ключ отношения D запишется в виде:

$$K_d \supset K_b \bigcup K_c = (K_a \bigcup \lambda K_b) \bigcup (K_a \bigcup \lambda K_c) = K_a \bigcup \lambda K_b \bigcup \lambda K_c$$

Теперь введём суррогатные ключи для отношений B и C, такие, что

$$K_b^s \Rightarrow K_b$$

$$K_c^s \Rightarrow K_c$$

где знак => означает однозначное соответствие, то есть, каждому уникальному значению суррогатного ключа соответствует уникальная комбинация значений атрибутов интеллектуального ключа. Тогда ключ отношения D в схеме, построенной на суррогатных ключах, можно записать в виде:

$$K_d \supset K_b^s \bigcup K_c^s$$

Предположим, что следующие значения атрибутов, входящих в ключи отношений B и C, являются допустимыми:

$K_{b1}^s \Rightarrow K_{b1} : \alpha_1, \beta_1, \beta_2, \ldots$ и $K_{b2}^s \Rightarrow K_{b2} : \alpha_2, \beta_1, \beta_2, \ldots$

$K_c^s \Rightarrow K_c : \alpha_1, \gamma_1, \gamma_2, \ldots$

здесь $\alpha_1$ является значением атрибута внешнего ключа, ссылающегося на первичный ключ отношения A; $\beta_i$ – значения атрибутов, входящих в интеллектуальный первичный ключ отношения B; $\gamma_i$ – значения атрибутов, входящих в интеллектуальный ключ отношения C.

В этом случае ключ отношения D в схеме, построенной на интеллектуальных ключах, может принять только одно значение, а именно:

$$K_d : \alpha_1, \beta_1, \beta_2, \ldots \gamma_1, \gamma_2, \ldots$$

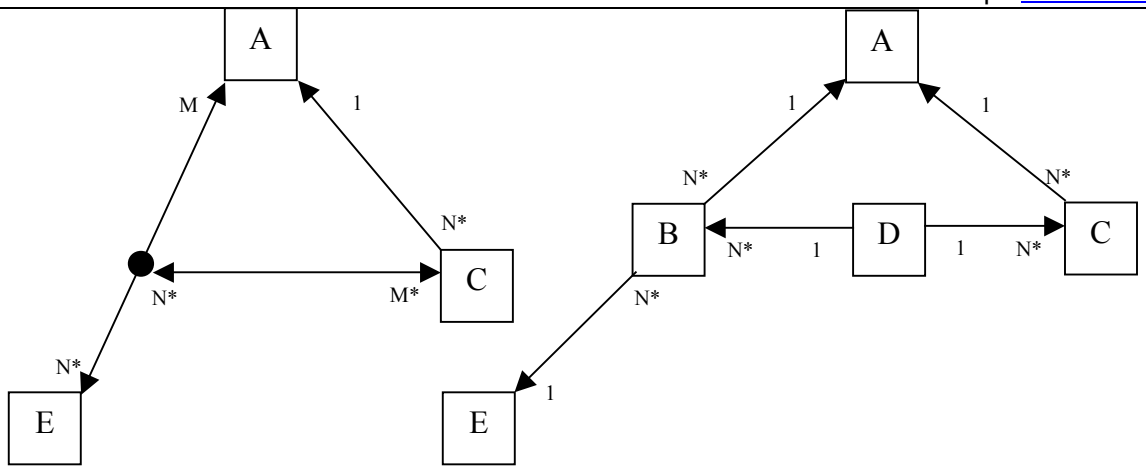
Однако ключ отношения D в схеме, построенной на суррогатных ключах, может принять два значения:

$$K_d^s : K_{b1}^s, K_c^s$$

$$K_d^s : K_{b2}^s, K_c^s$$

Реально же это означает, что второй ключ является некорректным, поскольку возникает коллизия по атрибуту $\alpha$.

Δ-коллизия является не единственной коллизией возможной при введении суррогатных ключей. Можно рассмотреть другой пример с P-коллизией. P-схема является развитием Δ-схемы. Пусть отношение B, из рассмотренного ранее примера, реализует связь многие-ко-многим между отношениями A и E. Так, как показано на рис. 2.




*A Концептуальная модель* *B. Физическая модель*

*Рис. 2. P-схема*

Теперь интеллектуальный ключ отношения B представим в виде:

$$K_b \supset K_a \bigcup K_e$$

а суррогатный ключ сохранит прежний вид:

$$K_b^s \Rightarrow K_b$$

Интеллектуальный ключ отношения D в этом случае примет вид:

$$K_d \supset K_b \bigcup K_c = (K_a \bigcup K_e) \bigcup (K_a \bigcup \lambda K_c) = K_a \bigcup K_e \bigcup \lambda K_c$$

В то время как суррогатный ключ снова останется неизменным:

$$K_d \supset K_b^s \bigcup K_c^s$$

Определим допустимые комбинации значений атрибутов, входящих в ключи отношений B и C:

$K_{b1}^s \Rightarrow K_{b1} : \alpha_1, \delta_1$ и $K_{b3}^s \Rightarrow K_{b3} : \alpha_2, \delta_1$

$K_c^s \Rightarrow K_c : \alpha_1, \gamma_1, \gamma_2, \ldots$

здесь $\sigma_i$ – значения атрибута внешнего ключа отношения B, ссылающегося на отношение E, и входящие в интеллектуальный ключ отношения B.

P-коллизию можно сформулировать следующим образом. В отношении D каждому значению ключа $K_e$ из отношения E может быть поставлено только одно значение ключа $K_c$ отношения C. Ключ отношения D, в схеме на интеллектуальных ключах, по-прежнему может принимать только одно значение, а именно:

$$K_d : \alpha_1, \delta_1, \gamma_1, \gamma_2, \ldots$$

Следовательно, наложенное ограничение не может быть нарушено. Однако в схеме, построенной на суррогатных ключах, ключ отношения D может принимать два значения:

$$K_d^s : K_{b1}^s, K_c^s$$

$$K_d^s : K_{b3}^s, K_c^s$$

Здесь оба варианта являются допустимыми, но совместно они нарушают наложенное ограничение, поскольку одному и тому же значению ключа $K_e(\sigma_1)$ дважды соответствует одно и то же значение $K_c(\alpha_1,\gamma_1,\gamma_2,\ldots)$. Другими словами, если не устранена Δ-коллизия, то она может быть усугублена ещё и P-коллизией.

Для иллюстрации коллизий, рассмотрим фрагмент предметной области, представленный на рис. 3. Более подробно данная предметная область описана в приложении.

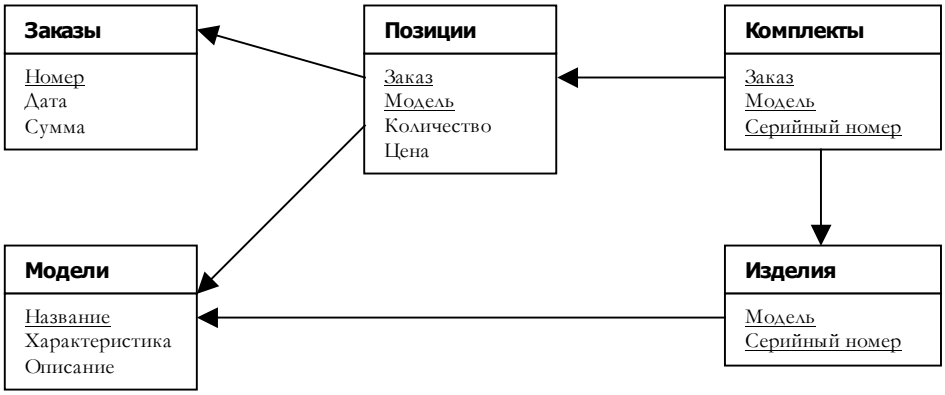



*Рис.3. Использование интеллектуальных ключей*

Из приведённого фрагмента схемы базы данных видно, что модели идентифицируются по своему названию, изделия имеют составной ключ, состоящий из названия модели и серийного номера изделия (серийные номера у разных моделей могут совпадать). Ключом заказов является их номер. Позиции заказа исполняют роль связи многие-ко-многим между моделями и заказами, и, кроме того, характеризуют количество изделий конкретной модели в конкретном заказе, а также цену данной модели. Отношение «Комплекты» играют роль связи многие-ко-многим между отношениями «Позиции» и «Изделия». Благодаря отношению «Комплекты» можно узнать, какими изделиями укомплектована та или иная позиция заказа. При этом несколько заказов могут быть укомплектованы одним изделием. Это возможно потому, что покупатель может вернуть изделие после факта продажи, если оно его не устроило по каким-либо параметрам или оказалось бракованным. Бракованные изделия возвращаются изготовителю и после того, как они приведены в соответствие с техническими требованиями, они могут быть снова пущены в продажу. Таким образом, одно и то же изделие может укомплектовать более одного заказа.

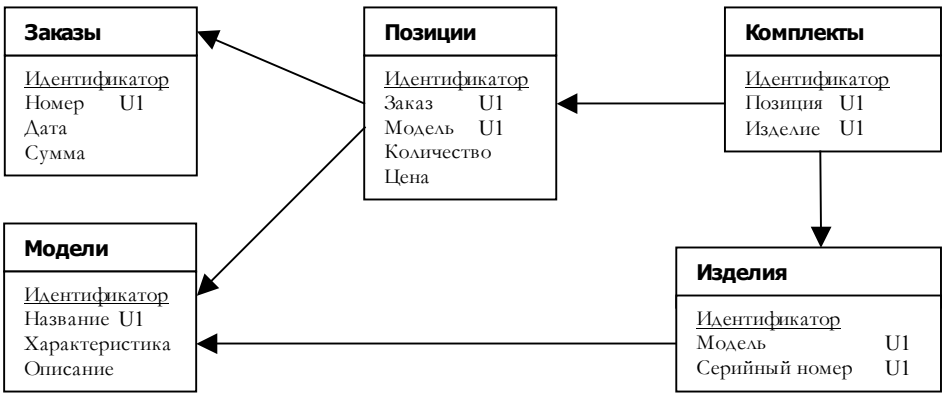



*Рис.4. Использование суррогатных ключей*

При использовании суррогатных ключей получается схема представленная на рис. 4. Для соответствия предметной области необходимо поддерживать уникальность тех полей, которые образуют интеллектуальные ключи (поля, помеченные символами «Ux»), помимо уникальности суррогатных ключей, которые выступают в качестве первичных ключей. Но даже в этом случае схемы баз данных, представленные на рис. 3. и рис. 4. различаются. Схема на суррогатных ключах не обеспечивает достоверности информации, в отличие от схемы на интеллектуальных ключах, то есть имеет коллизии.

В случае с интеллектуальными ключами первичный ключ отношения «Комплекты» образуется из двух внешних ключей: «Заказ» + «Модель» и «Модель» + «Серийный номер». При этом атрибут «Модель» существует в единственном числе. Когда происходит комплектация заказа, то атрибут «Модель» получается из соответствующего кортежа отношения «Позиции». Теперь в соответствие полученной модели необходимо поставить серийный номер конкретного изделия. Для указанной модели можно подставить нужный серийный номер из отношения «Изделия». Таким образом, можно корректно укомплектовать все позиции заказа. Теперь можно рассмотреть коллизии, которые возможны в модели построенной на суррогатных ключах.

P-Коллизия

В модели, построенной на суррогатных ключах, можно многократно ввести одно и то же изделие в рамках одного заказа, что является недопустимым. Для того чтобы убедиться в этой возможности, надо посмотреть на отношение «Комплекты». Как видно из схемы, представленной на рис. 4, ничто не препятствует введению одного изделия в несколько различных позиций одного и того же заказа.

Δ-Коллизия

Δ-коллизия проявляется в том, что при использовании схемы, построенной на суррогатных ключах, можно заданную в заказе модель укомплектовать изделием, относящимся к совсем иной модели. Это тоже легко увидеть из схемы, представленной на рис. 4, если ещё раз проанализировать отношение «Комплекты».

Возможно, имеет смысл ещё раз пояснить суть отношения «Комплекты». Это отношение реализует связь многие-ко-многим между отношениями «Позиции» и «Изделия». Соответственно отношение «Комплекты» образуется за счёт внешних ключей этих двух отношений и является полностью ключевым в обеих схемах. При этом в случае схемы с интеллектуальными ключами коллизии исключены, но в схеме с суррогатными ключами они вполне возможны. Это обстоятельство приводит к тому, что «тождественная» схема на суррогатных ключах перестаёт соответствовать требованиям предметной области, в отличие от схемы, построенной на интеллектуальных ключах.

Априори понятно, что наличие коллизий связано с тем, что в отношении «Комплекты» отсутствует понятие модели. Однако ввести атрибут «модель» в это отношение не так просто. Для того чтобы модель в отношении «Комплекты» была корректной, она должна ссылаться на модель в отношении «Позиции». Но атрибут «Модель» в отношении «Позиции» не обладает свойством уникальности. Поэтому помимо атрибута «Модель» придётся экспортировать и атрибут «Заказ» в отношение «Комплекты». Аналогичные рассуждения можно привести к отношению «Изделия». Из этого отношения потребуется экспортировать атрибуты «Модель» и «Серийный номер». Тогда отношение «Комплекты» будет иметь атрибуты:

- Идентификатор;
- Модель;
- Позиция;
- Серийный номер.

В результате получено то же отношение, что и в схеме, построенной на интеллектуальных ключах, что со всей очевидностью показывает, что интеллектуальные ключи являются единственной схемой, обеспечивающей достоверность информации. Безусловно, есть и другой путь. Можно использовать суррогатные ключи и поддерживать достоверность с помощью дополнительных механизмов, например, триггеров. Однако это приведёт к ещё большему усложнению логики базы данных.

Потерю достоверности информации в подобных ситуациях отследить не так просто, поскольку данный факт наличия коллизий в схеме базы данных, основанной на суррогатных ключах, не столь очевиден. Надо отметить, что приведённые выше фрагменты схем баз данных достаточно просты, в реальной ситуации количество отношений и сложность связей между ними могут быть значительно выше. Преодолеть потерю достоверности информации в схеме, основанной на суррогатных ключах, без удаления суррогатных ключей или введения вспомогательных механизмов.

*Эффективность связей между отношениями*

Когда говорят об эффективности связей основанной на суррогатных ключах, то подразумевают два основных момента. Первый связан с тем, что суррогатные ключи компактны. Второй момент состоит в том, что суррогатный ключ, как правило, содержит числовую информацию, которую процессоры обрабатывают наиболее эффективно.

В отличие от суррогатных ключей, интеллектуальные ключи имеют тот размер, который они должны иметь. Размер интеллектуального ключа определяется предметной областью, но не возможностями того или иного процессора. Если в какой-то таблице используется атрибут кода страны, то в случае интеллектуального ключа он будет 2-х или 3-х символьным, в то время как суррогатный ключ будет определяться СУБД и возможностями конкретного процессора, и он сегодня может быть 32 бита или 64 бита. В этом частном случае интеллектуальный ключ может оказаться меньше, чем суррогатный ключ. В другом случае интеллектуальный ключ может состоять из нескольких строковых атрибутов, каждый из которых будет, к примеру, более десятка символов. Соответственно размер интеллектуального ключа окажется намного больше, чем размер суррогатного ключа.

Тем не менее, использование интеллектуальных ключей часто ограничивают, мотивируя такое решение тем, что их размер, как правило, больше, чем размер суррогатного ключа. Для значительного количества задач такое оправдание выглядит не очень убедительно. Действительно, можно рассмотреть такие наиболее распространённые задачи, как автоматизация бухгалтерии, отдела кадров, складской учёт, учёт оборудования и т.п., и убедиться в том, что здесь интеллектуальные ключи основных сущностей имеют формат целого числа. Так, номер счетов в бухгалтерии типичное целое число; табельный номер служащего, также представим в виде целого числа, как, впрочем, и номер договора со служащим; номенклатурные номера товаров и инвентарные номера оборудования тоже представимы целыми числами. Таким образом, заявления о том, что интеллектуальные ключи обязательно имеют больший размер, а, следовательно, работа с ними происходит медленнее, не всегда соответствуют действительности. Тем более странными кажутся попытки наряду с простым и компактным интеллектуальным ключом поддерживать дополнительный суррогатный ключ, который не имеет какой-либо информационной ценности.

Помимо этого, надо отметить, что во всём мире давно и успешно применяют уникальные аббревиатуры. Это касается не только стран, городов и аэропортов, но и денежных единиц, единиц измерений, химических элементов, а также других устойчивых понятий. Например, табель рабочего времени, форма которого

утверждёна государственными органами, задаёт сокращения, которые соответствуют причинам не выхода на работу. Применение суррогатных ключей вместо устойчивых аббревиатур свидетельствует, по всей видимости, о слабом знании словаря предметной области.

Современное делопроизводство, управление и учёт широко используют общепринятые сокращения, различного вида коды, и этот процесс необратим, поскольку позволяет с меньшими затратами осуществлять автоматизацию основных операций. Примером тому служит повсеместное внедрение штрихкодового кодирования, включая двумерное штриховое кодирование, которое позволяет эффективно управлять материальными потоками и вести их непрерывный учёт. Аналогичные системы сегодня применяются, и в системах безопасности и для учёта рабочего времени (пластиковые карты со штрихкодом, магнитной дорожкой или микросхемой, вместо обычных пропусков), и т.п. Нет сомнений в том, что формализация основных потоков контроля и управления, взаимосвязей между этими потоками, как в рамках предметной области, так и вне неё, будет продолжаться. Не замечать этого процесса нельзя, а не пользоваться его результатами не разумно. Отрадно и то, что автоматизированный ввод информации решает проблему ошибок ввода.

Возвращаясь к эффективности связей между отношениями, надо отметить, что не существует прямой зависимости между эффективностью и размером ключевых атрибутов. Можно вернуться к схемам баз данных представленных на рис. 3 и рис. 4. Для того чтобы получить исчерпывающую информацию о том, какими изделиями был укомплектован заказ с номером 777 в схеме, построенной на интеллектуальных ключах, достаточно выполнить простой запрос, не требующий соединения таблиц:

```
SELECT MODEL, SERIAL_NO FROM COMPLETE_SET WHERE ORD_NO = 777;
```

Для достижения той же информативности в схеме, построенной на суррогатных ключах, придётся написать гораздо более сложный запрос, требующий соединения всех таблиц схемы:

```
SELECT M.NAME, PR.SERIAL_NO FROM ORDERS O
        INNER JOIN POSITIONS P ON P.ORDER_ID = O.ID
        INNER JOIN COMPLETE_SET CS ON CS.POSITION_ID = P.ID
        INNER JOIN PRODUCTS PR ON PR.ID = CS.PRODUCTS_ID
        INNER JOIN MODELS M ON M.ID = PR.MODEL_ID
WHERE O.ORD_NO = 777;
```

Можно не сомневаться в том, что первый запрос будет выполнен гораздо быстрее, чем второй. Аналогичная ситуация будет наблюдаться и в случае запроса противоположного приведённому выше. Пусть требуется узнать номер заказа, по которому проходило конкретное изделие:

```
SELECT ORD_NO FROM COMPLETE_SET
WHERE MODEL = “Celeron 500” AND SERIAL_NO = “XXXYYYZZZ”;
```

для схемы на интеллектуальных ключах. Для схемы на суррогатных ключах, опять нужно выполнить соединение всех таблиц:

```
SELECT O.ORD_NO FROM ORDERS O
        INNER JOIN POSITIONS P ON P.ORDER_ID = O.ID
        INNER JOIN COMPLETE_SET CS ON CS.POSITION_ID = P.ID
        INNER JOIN PRODUCTS PR ON PR.ID = CS.PRODUCTS_ID
        INNER JOIN MODELS M ON M.ID = PR.MODEL_ID
WHERE M.NAME = “Celeron 500” AND PR.SERIAL_NO = “XXXYYYZZZ”;
```

Потребность в очень большом числе слияний при использовании суррогатных ключей объяснима, она связана с отсутствием какой-либо полезной информации во внешних ключах.

Не смотря на то, что операция слияния таблиц является широко распространённой, она требует значительных ресурсов. Это можно проиллюстрировать на примере обратных запросов. Запрос к схеме, построенной на интеллектуальных ключах, выполнятся эффективно, поскольку оба атрибута, «Модель» и «Серийный номер» составляют внешний ключ. СУБД, как правило, по внешним ключам автоматически создаётся индекс. С помощью этого индекса и будут найдены требуемые кортежи.

При исполнении запроса к схеме базы данных, построенной на суррогатных ключах, действия будут иные. Сначала при помощи уникального индекса будет найдена требуемая модель в таблице моделей. Далее произойдёт загрузка в память нужного кортежа этой таблицы. Из этого кортежа будет извлечён идентификатор модели. Теперь необходимо по найденному идентификатору и заданному серийному номеру выполнить поиск в уникальном индексе, который был создан на атрибутах «Model_ID» и «Serial_No». Кортеж, соответствующий критерию поиска подгружается в память и из него извлекается значение атрибута «Product_ID». По этому атрибуту в индексе, образованном внешним ключом таблицы комплектующих, надо найти все соответствующие кортежи и загрузить их в память. Из этих кортежей извлекаются значения атрибутов позиций заказов. По выбранным атрибутам с помощью уникального индекса, образованного первичным ключом таблицы позиций, выбираются кортежи, из которых получают идентификаторы заказов. И вот, наконец, можно перейти к таблице заказов и узнать номера заказов, по которым проходило заданное изделие.

Обычно, для ускорения операций объединения используются кластерные индексы, то есть информация в таблице упорядочивается по значению первичного ключа. Имеет смысл отметить, что запросы, поступающие от пользователей, никогда не задают поиск информации по атрибуту, образующему, суррогатный ключ, по той простой причине, что значения этого атрибута неизвестны пользователю. Таким образом, эффективность поиска по сравнению с использованием естественных ключей снижается, так как СУБД сначала должна найти по вторичному индексу значение кластерного индекса, а затем уже по этому значению сам требуемый кортеж. Если использовать в качестве кластерного индекса вторичный индекс, то снижается эффективность объединений таблиц.

Вопросы производительности всегда были в центре внимания разработчиков программного обеспечения, поэтому и вопросы эффективности обработки запросов рассматриваются в данной работе достаточно подробно. Методику тестирования и полученные результаты можно найти в приложении.

**Пересекая границы базы данных**

Применение интеллектуальных ключей оправдывает себя и в том случае, если необходимо синхронизировать информацию в нескольких базах данных. Поскольку интеллектуальные ключи являются составной частью предметной области, то они сохраняют своё значение безотносительно к тому, в каком количестве баз данных (или одной распределённой базе данных) реализована предметная область.

В отличие от интеллектуальных ключей, суррогатные ключи вырабатываются каждой базой данных (или каждым узлом распределённой базы данных) самостоятельно. Поэтому при переносе информации из одной базы данных в другую, «старые» суррогатные ключи, взятые из исходной базы данных, могут конфликтовать с суррогатными ключами в приёмной базе данных. Вернёмся к схеме, представленной на рис. 4. Предположим, что выполняется репликация отношения «Заказы» из одной базы данных в другую базу данных, аналогичную по своей структуре. Если переносить полный кортеж, включая атрибут суррогатного ключа, то возможен конфликт по уникальности ключа в приёмной базе данных,

поскольку может оказаться, что кортеж с тем же значением суррогатного ключа в отношении «Заказы» приёмной базы данных уже существует.

С другой стороны, если не включать атрибут суррогатного ключа в схему репликации, то появляется проблема с репликацией отношения «Позиции». Каждая позиция должна быть связана со своим заказом, а поскольку первичный ключ из отношения «Заказы» при репликации исчезает, то теряется и связь между заказом и его позициями.

Опасность представляет собой и вариант перезаписывания значений суррогатного ключа в приёмной базе данных. В этом случае атрибут кортежа отношения «Позиции» может сохранить своё значение, которое было справедливо в исходной базе данных, но будет неверным в приёмной базе данных. Например, некоторая позиция заказа ссылалась на заказ, имеющий значение суррогатного первичного ключа равное XXX. После переноса заказа в приёмную базу данных, ему был присвоен номер YYY, но позиция по-прежнему ссылается на заказ XXX. Произойдёт либо нарушение ссылочной целостности, либо, что ещё хуже, позиция заказа станет принадлежать совсем другому заказу. Второй вариант вполне возможен в случае, если информация переносится из меньшей базы данных в большую, например, из базы данных филиала в базу данных центрального офиса.

Перенос данных с перезаписью значений атрибутов суррогатного ключа ставит ещё и проблему идентификации одной и той же записи в разных базах данных. Поскольку значения первичных суррогатных ключей в разных базах данных будут различаться, то непосредственная связь по ним будет невозможна. Для поддержания связи необходимо будет иметь дополнительную таблицу. В ней будет происходить фиксация информации о том, из какой именно базы данных, из какой таблицы и с каким значением атрибута суррогатного ключа был этот кортеж в исходной базе данных, и с каким значением атрибута суррогатного ключа он был занесён в приёмную базу данных. Поддержание таблиц синхронизации необходимо, так как информация в кортежах, которые были реплицированы, может меняться и эти изменения должны быть корректно перенесены в другие базы данных. Но решение, изложенное выше, приемлемо только для одноуровневой звездообразной схемы репликации.

При большем количестве уровней вместе с основным кортежем, придётся передавать и кортежи таблицы синхронизации исходной базы данных, в противном случае, конечная база данных не будет иметь представления о том, с каким значением атрибута суррогатного ключа был сохранён кортеж в той базе данных, где эта информация впервые была введена. То есть, приёмной базе данных надо будет фиксировать не только занесение нового кортежа, но и то, как был зафиксирован этот кортеж в исходной базе данных. Это требование серьёзно осложняет применение данного метода репликации, основанного на перезаписи первичных ключей.

При двусторонней репликации таблицы синхронизации должны быть в обеих (исходной и приёмной) базах данных. После передачи кортежа исходная база данных должна получить информацию о том, с каким значением атрибута суррогатного ключа этот кортеж был зарегистрирован в приёмной базе данных. И здесь процесс фиксации кортежа должен сопровождаться раскруткой его истории прохождения по базам данных для исключения циклических ссылок.

Другим решением проблемы репликации на основе суррогатных ключей может стать выделение диапазонов значений суррогатных ключей для одноимённых отношений в каждой базе данных. Однако в этом случае кто-то должен заниматься распределением и перераспределением диапазонов в случае появления новых баз данных, а также отслеживать возможные выходы за пределы диапазона установленные для той или иной таблицы баз данных. В результате

снова происходит усложнение логики, как отдельно взятой базы данных, так и схемы репликации в целом.

В отличие от столь сложных решений репликация на основе интеллектуальных ключей не требует никаких дополнительных затрат. Это объясняется тем, что интеллектуальные ключи являются уникальными в пределах предметной области, а не отдельно взятой базы данных, о чём уже говорилось в начале раздела. С другой стороны, нельзя не учитывать того, что сегодня централизованные системы становятся слишком сложными и громоздкими и существует очевидная тенденция перехода к распределённым системам, но использование суррогатных ключей в распределённых системах сопряжено с проблемами, перечисленными выше.

**Заключение**

На основании изложенного выше, можно сделать вывод о том, что использование суррогатных ключей оправдано в двух случаях:

- отсутствие интеллектуального ключа;
- ограничение на размер первичного ключа.

Отсутствие интеллектуального ключа возможно в случае, когда предметная область не включает в себя атрибуты, которые позволяют идентифицировать кортеж. С другой стороны, атрибуты, которые позволяют идентифицировать каждый кортеж отношения, могут быть слишком велики и их совокупный размер может превысить ограничение на размер ключа у конкретной СУБД.

Иногда называют третий случай, когда оправдано применение суррогатных ключей: необходимость унифицировано идентифицировать кортежи любого отношения базы данных. Однако в данной ситуации лучше использовать уникальный индекс на вспомогательном, как правило, целочисленном атрибуте.

Пресловутая устойчивость суррогатных ключей связана в основном с просчётами при проектировании и не может служить оправданием повышению сложности базы данных и запросов. Это тем более очевидно, если учесть тот факт, что модификации схем баз данных или отдельных отношений в хорошем проекте представляют собой исключительное события, в то время как взаимодействие с базами данных ведётся повседневно в оперативном режиме. Стоит ли повседневную работу приносить в жертву гипотетически возможным изменениям.

В отличие от суррогатных ключей интеллектуальные ключи позволяют получать более простые и элегантные структуры баз данных. В силу своей информативности интеллектуальные ключи делают проще и эффективнее запросы, исключая избыточные соединения. Помимо этого, схемы, построенные на интеллектуальных ключах, устойчивы к появлению коллизий, а значит, эти ключи лучше обеспечивают достоверность информации в базе данных. Эти ключи требуют тщательного изучения предметной области и детального знания атрибутов сущностей, а также функциональных зависимостей между атрибутами. Однако этот процесс изучения имеет свои положительные стороны, позволяя получать продуманные и устойчивые к внешним изменениям схемы баз данных. Учитывая компактность многих интеллектуальный ключей, не следует ожидать увеличения размеров баз данных при их использовании.

Применение суррогатных ключей вызвано определённым несовершенством современных СУБД, в частности, способом хранения информации по кортежам. В такой ситуации одна и та же информация многократно дублируется в различных структурах. Так атрибуты первичного ключа хранятся не только в самом отношении, но, как правило, входят и в уникальный индекс, который автоматически создаётся для каждого первичного ключа. Если какое-то отношение ссылается на другое отношения, то внешний ключ опять же сохраняется не только

в ссылочном отношении, но и в индексе, построенном на внешнем ключе. При связи между отношениями один-ко-многим значения первичного ключа могут многократно повторяться во внешнем ключе. Таким образом, одни и те же значения из атрибутов первичного ключа многократно представляются в базе данных.

Давая определение термину отношения, Е. Кодд писал [ЕК РМ]: «Термин *отношение* используется здесь в его общепринятом математическом смысле. Для заданных множеств $S_1$, $S_2$, ..., $S_n$ (не обязательно различных) R является отношением на этих n множествах, если представляет собой множество кортежей степени n, у каждого из которых первый элемент взят из множества $S_1$, второй - из множества $S_2$ и т.д. Мы будем называть $S_j$ *j-тым доменом R*». Это определение позволяет представить отношение как совокупность атрибутов. В таком случае, ключевые атрибуты, на основании которых происходит связь между отношениями, могут разделяться несколькими отношениями, находящимися в связи. Внутреннее устройство этих атрибутов является их внутренним делом и неизвестно отношениям, которые разделяют атрибуты. Следовательно, их можно организовать наиболее удобным образом, например, в виде иерархической структуры, повышающей скорость поиска затребованных значений. В свою очередь, организация атрибутов в виде индекса позволяет иметь только одно представление информации в базе данных. (Более подробно эта схема хранения рассмотрена в письмах по проектированию, посвящённых проекту СУБД).

При данной схеме хранения информации потребности в суррогатных ключах не возникает, поскольку здесь не потребуются ни каскадные модификации, при изменении значения или удалении первичного ключа, ни сложных манипуляций с базой данных при изменении структуры атрибутов, составляющих первичный ключ. А поскольку такая схема хранения не противоречит реляционной модели, предложенной Е. Коддом, то можно сделать вывод о том, что суррогатные ключи являются привнесёнными в эту модель с целью попытки компенсации недостатков конкретных СУБД. Фактически к той же мысли можно придти, анализируя высказывание К. Дейта о существенности, о чём уже говорилось ранее.

**Список литературы**


| | | |
|---|---|---|
| [CD AC] | C.J. Date | «An Analysis of Codd's Contribution to the Great Debate», Intelligent Enterprise, May 11, 1999, Volume 2, Number 7 |
| [CD WENE] | C.J. Date | «When's an extension not an extension?», Intelligent Enterprise, June 1, 1999, Volume 2, Number 8 |
| [D&D] | Joe Celko | «Data & Databases: Concepts in Practice», The Morgann Kaufman Series in Data Management Systems |
| [ЮТ MSS] | Тихомиров | Ю.В. «Microsoft SQL Server 7.0» СПб.:БХВ – Санкт-Петербург, 1999. |
| [ЕК РМ] | Е.Ф. Кодд | «Реляционная модель данных для больших совместно используемых банков данных», СУБД №1, 1995. |

Некоторые статьи К. Дейта можно найти на http://www.citforum.ru в переводе Сергея Кузнецова.