Московский государственный технический университет им. Н.Э. Баумана

На правах рукописи

Мин Тхет Тин

МЕТОДИКА ФОРМИРОВАНИЯ РЕЛЯЦИОННЫХ ТАБЛИЦ НА ОСНОВЕ ИНФОРМАЦИИ ТАБЛИЧНОГО ВИДА

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель -
д.т.н., доцент, Брешенков А.В.

Москва 2014 г.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ

БД – база данных

РБД – реляционная база данных

РМД - реляционная модель данных

СУБД – система управления БД

ИТВ – информация табличного вида

РТ – реляционные таблицы

ЭТ – электронные таблицы

ПК – первичный ключ

ПО – программное обеспечение

ОС – операционная система

ВВЕДЕНИЕ

В настоящее время трудно переоценить значение компьютерных информационных систем. А коль скоро базы данных (БД) являются ядром информационных систем, в полной мере это относится и к БД. Это детально и убедительно доказывается в соответствующей научно-популярной и технической литературе. В частности об этом говорят специалисты в области БД [1−23]. Более того, в паспорте специальности 05.13.11 (Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей) отмечается:

− необходимость разработки и исследования в области программных средств организации и управления обработкой данных и знаний;

− необходимость создания прикладного математического обеспечения, программных средств автоматизации разработки программ;

− актуальность разработки программных средств обработки данных и знаний в ВМ, ВК и КС;

− актуальность разработки методы проектирования систем управления базами данных (СУБД) и базами знаний (СУБЗ), в том числе распределенными СУБД и СУБЗ.

Даже из незначительной информации, которую можно получить из названия диссертации, можно сделать вывод о том, что она посвящена решению названных проблем. Действительно, речь идет о преобразовании информации в данные, кодировании информации в виде данных. Естественно, для этого потребуются модели информации табличного вида и модели таблиц данных. Кроме того необходима разработка математического,

информационного, программного, методического и других видов обеспечений, ориентированных на преобразование информации в данные.

Собственно понятие информации − глобальное и охватывает все сферы человеческой деятельности от вербального общения между людьми до работы в Интернете [25]. Далеко не всякую информацию можно представить в виде данных. Ведь данные − это информация, представленная в регламентированном виде. К сожалению, не всю информацию можно строго регламентировать. Поэтому в работе рассматривается информация табличного вида (ИТВ). Представление такого рода информации близко к представлению данных в БД, и поэтому в принципе процесс преобразования ИТВ в формат БД можно формализовать. Но ИТВ по ряду признаков существенно отличается от данных. Суть проблемы в том и состоит, чтобы разработать способы, алгоритмы, методику и средства, которые позволят исключить эти отличия.

Для начала необходимо дать хотя бы неформальные определения ИТВ и представление данных в БД. ИТВ − это информация, которая воспринимается пользователем как двумерные таблицы.

По сути, это могут быть ведомости, прайс-листы, словари, списки и многое другое. Главная задача разработчиков такого рода таблиц − это обеспечение минимальной субъективной сложности восприятия информации. При этом:

− данные в столбцах могут не совпадать по типу;

− возможно дублирование записей;

− допустимо отражение семантики данных посредством цвета, фона, шрифта и т.п.;

− допустимо использование подзаголовков и внутренних заголовков [26];

− допустимо повторное использование заголовков и подзаголовков (суть их определяется посредством места в таблице, цвета, фона, шрифта и т.п.).

В этом случае никакой речи о регламентации информации не может быть и речи. Проблема и состоит в преобразовании нерегламентированной информации к регламентированному виду. А регламентированный вид − это формат БД.

По форме ИТВ может быть представлено на бумаге, в формате текстовых редакторов, в формате текстовых процессоров, в формате электронных таблиц и во многих других форматах. В связи с этим возникает вторая проблема – проблема преобразования форматов.

Естественно задаться вопросом, а нужны ли преобразования ИТВ в таблицы БД. Собственные исследования, работа с экспертами, участие в разработках показали, что, с одной стороны, БД исключительно редко создаются на пустом месте (чаще всего имеются значительные объемы информации вида ИТВ), а, с другой стороны, к настоящему времени накопилось множество ИТВ, которое просто необходимо обрабатывать средствами современных систем управления базами данных (СУБД).

И, к сожалению, в настоящее время нет теоретических и практических разработок, которые могли бы в полном объеме решить проблемы преобразования ИТВ в формат БД. Как сказал один из основоположников теории проектирования БД Дейт К. Дж.: “проектирование БД – это скорее искусство, чем наука” [12]. А преобразование ИТВ в формат БД − это важнейшая проблема, стоящая перед проектировщиками БД. Причем эта проблема еще менее исследованная область знаний по сравнению с традиционными методами проектирования БД.

Достаточно большой объем работы в области проектирования БД на основе использования ИТВ проделал Брешенков А.В. [4−7, 26−37]. Однако, несмотря на глубокую теоретическую и практическую проработку проблемы, в работах Брешенкова А.В. не рассматриваются некоторые задачи, которые необходимо решать в процессе преобразования ИТВ в формат БД. В частности:

− рассмотрены не все возможные виды подзаголовков в ИТВ;

− не рассмотрены гибридные подзаголовки;

− в качестве атрибутов, которые входят в первичный ключ, анализировалось не более 3-х;

− связи между таблицами рассмотрены для ключевых полей, включающих только один атрибут;

− не проанализировано одно из требований минимальности первичного ключа: никакая часть первичного ключа не должна быть уникальной;

− не проведено детальное исследование по поводу выявления внешних ключей в ИТВ.

Все это, конечно, не умаляет научные и практические достоинства работ [4−7, 26−37]. Более того, эти работы послужили хорошей базой для дальнейших разработок и, в частности, разработок, выполненных в диссертации.

Естественен вопрос – какую концептуальную модель БД использовать в качестве целевой при преобразовании ИТВ? К числу наиболее популярных концептуальных моделей БД относятся реляционные модели. Это связано с их наглядностью, простотой восприятия и реализации, наличием формального аппарата для представления и обработки данных. Несмотря на применение предшествующих моделей данных (иерархической и сетевой) и, несмотря на бурное развитие новых моделей данных (постреляционной, многомерной и объектно-ориентированной), реляционные БД (РБД) в настоящее время занимают в мире лидирующее положение и подавляющее большинство существующих БД и СУБД построены в соответствии с реляционным подходом [38]. Как указывают даже сторонники новых концептуальных моделей данных: “Реляционные данные существуют уже около 30 лет. За это время вспыхивало несколько революций, которые должны были положить конец реляционным хранилищам. Конечно, ни одна из этих революций не состоялась, и ни одна из них ни на йоту не поколебала позиции реляционных БД” [44].

За последние годы выполнен значительный объем научных исследований, посвященных проектированию реляционных баз данных (РБД). Среди них можно назвать работы Е. Ф. Кодда [19-23], К. Дж. Дейта [11-13, 39-44], Гэри Хансена, Джэймса Хансена [45], Ульмана Дж. [47-48], Чена Р. Р. [177], Райана Стивенса, Рональда Плю [154], Дэйва Энсора [112], Тихомирова Ю.В. [48-49], Григорьева Ю.А., Ревункова Г.И. [50-51], Карповой Т.С. [53], Брешенкова А.В. [4-7, 26-37, 53] и других.

Однако, несмотря на значительные успехи в области проектирования РБД большинство задач проектирования формализовать не удается. Это, в частности, отмечает и Дейт К. Дж. [13]. Такое положение вещей связано в большинстве случаев с тем, что в процессе проектирования БД в основном используются не сами данные или информация, преобразуемая в данные, а предполагаемые схемы отношений. В процессе эксплуатации БД может оказаться, что предположения ошибочны и схемы отношений сформированы ошибочно. “Действительно, не зная содержимого таблиц, а только отталкиваясь от их схемы отношений, далеко не всегда возможен правильный и оптимальный выбор ключевых полей, выявление функциональных зависимостей, решение вопросов нормализации, обоснованное формирование связей между таблицами. Ведь все эти вопросы решаются неформально на основании предполагаемого содержимого таблиц с данными, которых еще нет” [53]. В связи с этим БД приходится нередко перепроектировать, что связано с большими издержками различного характера.

Ситуация кардинально меняется, когда в качестве исходных сведений для проектирования используются существующая информация.

Ее анализ можно формализовать и в конечном итоге принять лучшие решения. В особенности это касается ИТВ.

С другой стороны, значительная часть информации, в том числе и информация табличного вида, находится вне баз данных и даже вне ЭВМ,

хотя потребители этой информации, как правило, очень заинтересованы в возможности использования многочисленных возможностей БД [5,6].

Эти два положения и определяют актуальность разработки способов, алгоритмов, методики и средств формирования реляционных таблиц на основе существующих ИТВ.

Конечно, традиционная методология проектирования РБД является мощным, а на сегодня лучшим подходом к разработке реляционных БД, использующих в качестве концептуальной модели реляционную модель данных. Поэтому было бы просто неразумно в данной работе не использовать инструмент создания целостных, непротиворечивых и неизбыточных систем.

С другой стороны, неразумно не воспользоваться фактами повсеместного использования ИТВ и не решать задачи проектирования БД на основе анализа имеющейся информации. В связи с этим и возникает проблема теоретических и практических разработок, ориентированных на разработку способов, алгоритмов, методики и средств формирования реляционных таблиц на основе информации табличного вида.

Проблема заключается в отсутствии полного комплекса способов, алгоритмов, методики и средств, ориентированных на преобразование ИТВ в реляционные таблицы (РТ).

В работе предлагаются методика преобразование ИТВ в РТ, в основе которого лежат модели ИТВ и РБД, способы и алгоритмы решения проектных задач, соответствующие лингвистические и программные средства.

Предметом исследования являются модели, способы и методика проектирования РБД на основе использования ИТВ, а также компоненты математического, лингвистического, информационного и программного обеспечений методики преобразования ИТВ в реляционные таблицы.

Целью работы является разработка в рамках методики теоретических и практических основ формирования таблиц РБД на базе ИТВ, улучшение

качественных и количественных характеристик существующих средств и алгоритмов решения задач формирования РБД на основе ИТВ для:

− обработки всех возможных видов подзаголовков в ИТВ;

− исключения гибридных подзаголовков;

− использования в качестве первичных ключей произвольного числа атрибутов;

− исключения повторяющихся заголовков и подзаголовков;

− организации связей между таблицами по нескольким атрибутам;

− анализа требования минимальности первичного ключа: никакая часть первичного ключа не должна быть уникальной;

− выявления внешних ключей в ИТВ.

В соответствии с поставленной целью в диссертации решаются следующие задачи:

1. Анализ традиционных методик и моделей, используемых при проектировании РБД.
2. Уточнение моделей ИТВ и РБД, моделей таблиц ИТВ и РТ.
3. Анализ проблем, возникающих в ходе преобразования ИТВ в РТ.
4. Определение состава алгоритмов и средств для обеспечения эффективного решения задач преобразования ИТВ в РТ.
5. Разработка способа преобразования заполненных таблиц ИТВ в реляционные таблицы.
6. Разработка способа назначения ключевых полей в заполненных таблицах ИТВ.
7. Разработка алгоритмов, реализующих указанные способы.
8. Разработка методики преобразования ИТВ в РТ.
9. Программная реализация методики преобразования ИТВ в РТ.

При разработке формальных моделей и методики в диссертации использовались реляционная алгебра, реляционное исчисление, исчисление предикатов, теория множеств, теория алгоритмов.

Структура работы соответствует списку перечисленных задач, содержит описание разработанных способов, средств и алгоритмов, которые задействованы в методике преобразования заполненных таблиц ИТВ в реляционные таблицы.

В первой главе выполнен аналитический обзор традиционного подхода формирования реляционных таблиц, сформулированы его достоинства и недостатки. Уточнено понятие информации табличного вида. Сформулированы мотивы разработки способа преобразования ИТВ в РТ. Выполнен анализ проблем разработки методики в рамках методологии проектирования РБД на основе существующей информации табличного вида. Выполнена постановка задачи разработки методики формирования реляционных таблиц на основе существующей ИТВ. Определен состав алгоритмов и средств, разрабатываемых в рамках методики проектирования РТ на основе существующей информации табличного вида.

Во второй главе разработан способ преобразования таблиц ИТВ к реляционному виду. В рамках способа рассмотрены алгоритмы приведения атрибутов заполненных таблиц к единому типу, исключения дублирования записей в текстовых формах представления ИТВ, избавления от сложных атрибутов и исключения подзаголовков, исключения гибридных подзаголовков, исключения дублирования имен заголовков и подзаголовков.

В третьей главе разработан способ назначения ключевых полей в заполненных реляционных таблицах, который по сравнению с традиционными способами позволяет снизить трудоемкость и достигнуть наилучшего решения при выполнении данной проектной процедуры.

В четвертой главе разработана методика формирования реляционных таблиц на основе информации табличного вида. В рамках методики задействованы модели, способы и алгоритмы, разработанные в предыдущих главах диссертации.

В заключении представлены основные результаты работы.

В приложениях приводятся программная реализация методики формирования реляционных таблиц на основе информации табличного вида, акт внедрения результатов диссертационной работы.

Научную новизну работы определяет концепция и теоретические основы формирования РТ на базе ИТВ, которые воплощены в методику проектирования РТ.

Новые научные результаты, выносимые на защиту:

- Исследована проблема проектирования реляционных БД с использованием информации табличного вида.
- Расширена модель РТ.
- Построена модель таблиц ИТВ.
- Предложен новый способ преобразования ИТВ в реляционные таблицы.
- Предложен способ назначения ключевых полей в заполненных таблицах ИТВ.
- Разработана методика формирования реляционных таблиц на основе информации табличного вида.

Обоснованность научных положений, рекомендаций и выводов, изложенных в работе, определена корректным использованием современного математического аппарата. Достоверность положений и выводов диссертации подтверждена положительными результатами внедрения в учебный процесс МГТУ им. Н.Э. Баумана.

Научные результаты, полученные в диссертации, доведены до практического использования в учебном процессе. Они представляют интерес в области проектировании РБД. Методика, способы и алгоритмы, а также программные средства могут быть использованы при решении задач проектирования РБД на основе использования ИТВ.

Содержание отдельных разделов и диссертации в целом было изложено и получило одобрение:

- на Российских научно-технических конференциях и семинарах (2011 − 2014 г.г.);
- на заседании кафедры “Компьютерные системы и сети” МГТУ им. Н.Э. Баумана.

Совокупность научных положений, идей и практических результатов исследований составляет оригинальное направление в области проектирования реляционных баз данных в различных областях человеческой деятельности.

По результатам выполненных исследований опубликовано 11 научных работ.

Диссертационная работа состоит из введения, четырех глав и заключения, опубликованных на 149-и страницах машинописного текста, содержит 77 рисунков, 23 таблицы, список литературы из 100 наименований и 2 приложений.

1. АНАЛИЗ ПРОБЛЕМ РАЗРАБОТКИ МЕТОДИКИ ФОРМИРОВАНИЯ РЕЛЯЦИОННЫХ ТАБЛИЦ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИИ ТАБЛИЧНОГО ВИДА

В первой главе выполнен краткий анализ подхода **формирования** реляционных таблиц в рамках современной методологии проектирования реляционных баз данных, сформулированы его достоинства и недостатки. Определено понятие ИТВ и сформулированы мотивы разработки методики преобразования ИТВ в реляционное представление. Сформулированы мотивы и проблемы разработки методики формирования реляционных таблиц на основе использования ИТВ. Выполнен анализ применимости современных теоретических и практических разработок для решения проблемы преобразования таблиц. Выполнена постановка задачи разработки способа формирования РТ на основе использования ИТВ.

1.1 Аналитический обзор традиционного подхода формирования реляционных таблиц в контексте решаемой проблемы

1.1.1 Основы современной методологии проектирования реляционных баз данных

Проектирование РБД в соответствии с традиционной методологией включает в себя 4 этапа [11−13]. Как правило, выделяют следующие этапы проектирования: формулировка и анализ требований, инфологческое

проектирование, датологическое проектирование, физическое проектирование.

Формулировка и анализ требований связаны с определением целей разработки, выделением информационных потоков, экспертированием и многими другими мероприятиями, связанными с анализом предметной области. Главной задачей этого этапа является формулировка требований к разрабатываемой БД. В связи с этим этот этап заканчивается техническим заданием (ТЗ) на разрабатываемую БД.

При наличии ИТВ ТЗ в значительной степени уже сформулировано. Действительно, в общих чертах определены состав данных и их структура, известно смысловое назначение информации, имеются сведения о связях между сущностями. В связи с этим процесс формирования ТЗ при наличии ИТВ упрощается.

Инфологическое проектирование ориентировано на построение модели предметной области. В частности, выделяются сущности, формируются и оптимизируются локальные представления, назначаются первичные и внешние ключи, строятся диаграммы сущность − связь, таблицы приводятся к нормальным формам, формируются связи между таблицами. На этом этапе проектирования РБД разработчик абстрагируется от инструментальных средств их разработки. От таких как Oracle, Clarion, Access и других. В этом и состоит одно из основных достоинств теории проектирования РБД – разделение логического и физического уровней разработок [45-47] .

Именно на этом этапе проектирования РБД в наибольшей мере проявляется достоинство подхода формирования РТ на основе использования ИТВ. Оно обусловлено тем, что проектные задачи решаются не на основе анализа гипотетических схем отношений, а на основе анализа реальной информации, а это обеспечивает возможность формализации проектных процедур [53].

Датологическое проектирование связано с построением модели предметной области в терминах языков инструментальных средств разработки БД. При этом в качестве модели данных используются сами данные. Разработка датологической модели осуществляется на базе инфологической модели.

При проектировании РТ на основе существующих ИТВ достигается слияние инфологического и датологического этапов проектирования.

Физическое проектирование позволяет привязать датологическую модель к среде хранения. На этом этапе осуществляется выбор носителя данных, выбор внутренних форматов их хранения, выбор методов доступа к данным, методов сжатия данных, реализуются меры по безопасности данных [1].

На этом этапе несущественно, использовалась ли ИТВ при проектировании РТ.

1.1.2 Реляционная модель данных

РМД рассматривается практически во всех работах, посвященных проектированию РБД. В частности она обсуждается в работах [4−9], [11−13], [19−23]. Она является целевой моделью в контексте темы диссертации. В связи с этим ее необходимо обсудить. К числу удачных определений основной компоненты РМД можно отнести определение, представленное в работе [37]. По определению авторов “реляционная модель данных (РМД) некоторой предметной области представляет набор отношений, изменяющихся во времени”. Для однозначного понимания терминологии и сути данной работы оправдано привести таблицу элементов реляционной модели данных (таблица 1.1), которая представлена в работе [37].

Таблица 1.1

Элемент реляционной модели	Форма представления
Отношение	Таблица

Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значения поля в записи
Первичный ключ	Один из нескольких атрибутов
Тип данных	Тип значений элементов таблицы

Несмотря на это детальное представление элементов РМД в таблицу можно добавить описательный атрибут – свойства и ограничения атрибута, внешний ключ – атрибут, используемый для обеспечения уникальности записей и для связи между таблицами.

Основным понятием РМД является отношение, которое представляет собой подмножество декартового произведения доменов $D_1, D_2, \dots, D_k$ вида:

$D = D_1 \times D_2 \times \dots \times D_k$, где

$D_1 = (d_{11}, d_{12}, \dots, d_{1i}, \dots, d_{1m1})$

$D_2 = (d_{21}, d_{22}, \dots, d_{2i}, \dots, d_{2m2})$

...

$D_k = (d_{k1}, d_{k2}, \dots, d_{ki}, \dots, d_{kmn})$

Домен – множество элементов, типы которых могут не совпадать.

Отношение R представляется следующим образом:

$R \subseteq D = D_1 \times D_2 \times \dots \times D_k$ [1].

Следует отметить то, что в таблице 1.1 описаны скорее элементы РТ (составной части РМД), а не РМД. РМД отражает связи между таблицами. А в рамках темы данной работы как раз и представляет основной интерес именно РТ.

РТ соответствует отношению из *k* атрибутов, должна удовлетворять следующим свойствам:

- каждая строка представляет собой кортеж из *k* значений, принадлежащим *k* столбцам;
- каждый кортеж содержит точно одно значение (соответствующего типа) для каждого атрибута;
- порядок столбцов фиксирован *(1, 2, …, k);*
- порядок строк произволен;
- любые две строки различаются хотя бы одним элементом;
- строки и столбцы могут обрабатываться в любой последовательности, определяемой применяемыми операциями обработки;
- атрибуты не должны дублироваться [18-23].

Как правило, ИТВ, свойства которой рассмотрены во введении, не удовлетворяет большинству из перечисленных требований. В изменении такого положения вещей и состоит основная задача работы.

1.1.3. Ключевые поля и обеспечение целостности данных

В работах, посвященных теории проектирования РБД, в частности в [11-13], дается определение ключевых полей, обосновывается их необходимость, формулируются требования к ним и определяются свойства внешних ключей.

Ряд специалистов в области реляционных баз данных не включают в качестве требований к РТ наличие первичных ключей. Это оправданно лишь отчасти.

Действительно. Если эту проблему рассматривать в теоретическом аспекте, то отдельные первичные и внешние ключи могут быть назначены в процессе нормализации таблиц. Если эту проблему рассматривать в практическом аспекте, большинство инструментальных средств, ориентированных на проектирование РБД, позволяют описывать реляционные таблицы без указания первичных и внешних ключей.

В связи с этим ключевые поля могут быть просто не назначены, а это недопустимо. Ведь одно из центральных требований к РТ звучит следующим образом: любые две строки таблицы должны различаться хотя бы одним элементом. А в этом и состоит одно из назначений первичных ключей.

К основным требованиям к первичным ключам относятся их уникальность и минимальность.

В соответствии с требованием уникальности первичного ключа ни одно значение, соответствующее атрибуту первичного ключа не должно повторятся. Если первичный ключ включает в себя несколько атрибутов, то это требование звучит следующим образом – ни один кортеж значений соответствующий всем атрибутам, входящим в первичный ключ, не должен дублироваться.

В соответствии с требованием минимальности первичного ключа, с одной стороны, суммарная длина атрибутов, входящих в первичный ключ, должна быть минимальной, а, с другой стороны, ни один атрибут, входящий в первичный ключ, не должен быть уникальным (в том смысле, что его значения могут повторяться).

При наличии только схемы отношения непросто выбрать атрибуты, удовлетворяющие данным условиям. При этом легко ошибиться, т.к. после заполнения таблицы реальная ситуация может отличаться от предполагаемой.

Используя ИТВ, разработчик РБД имеет возможность принимать решение на основе анализа существующих данных. Кроме того, процесс поиска уникальных и минимальных ключей можно формализовать.

Кроме первичных ключей в РБД широко используются внешние ключи. [15, 16]. Такие ключи используются для связи между таблицами. Определяют внешние ключи следующим образом: если в отношении R1 значения какого-либо атрибута, не входящие в его первичный ключ, равны значениям первичного атрибута отношения R, то такой атрибут называют

внешним ключом для отношения R. В данной работе реализован способ поиска такого рода атрибутов в ИТВ.

Целостность данных в основном рассматривается в двух аспектах – целостность сущностей и целостность согласования. [13].

Требование целостности сущностей можно сформулировать так: сущности реального мира должны быть различимы. То есть ни одна запись в таблице не должна повторяться. В противном случае возникнет множество проблем в процессе обработки данных.

Целостность согласования: ссылаться можно только на те данные, которые существуют. Такую целостность еще называют ссылочной целостностью. Нарушение этого правила может привести к трудноразрешимым проблемам в процессе функционирования БД.

Обеспечение целостности данных связано с правильным использованием первичных и внешних ключей. В работах [59-61] достаточно полно рассмотрены вопросы обеспечения целостности данных при проектировании РБД на основе ИТВ.

1.1.4. Нормализация и семантическое моделирование

Нормализация отношений – аппарат ограничений на формирование отношений, который позволяет устранить избыточность БД, обеспечивает непротиворечивость хранимых данных.

“Концепции этой методологии являются не чем иным как соображениями здравого смысла, записанными в формальном виде. Сутью теории является поиск и формирование этих принципов здравого смысла, что, конечно же, является весьма непростой задачей. Однако, если такая задача будет решена, найденные принципы могут быть положены в основу решений автоматизации, т.е. можно будет написать программу, позволяющую выполнять проектирование с помощью компьютера” [13].

Процесс нормализации отношений основывается на концепции нормальных форм. Переменная отношения находится в нормальной форме,

если она удовлетворяет заданному для нее набору условий. В таком случае процедуру нормализации можно охарактеризовать как последовательное приведение данного набора отношений к некоторой более желательной форме. Процедура нормализации включает разбиение или декомпозицию переменной отношения, не удовлетворяющей условиям нормальной формы [13]. Процесс декомпозиции на самом деле является операцией проекции исходной переменной отношения.

Важно отметить, что операции проекции имеют смысл, когда отношение имеет не только схему, но и тело. А при традиционном проектировании тело отношения может иметь лишь гипотетический характер.

В работах [4-7, 27-37, 53] предложены и реализованы решения автоматизации, которые возможны для весьма часто встречающихся ситуаций, когда РБД проектируются на основе существующей ИТВ. В этих работах, в том числе, на достаточно высоком уровне разработаны и методы нормализации таблиц ИТВ. В связи с этим вопросы нормализации в данной работе не рассматриваются.

Семантическое моделирование. Разработчики БД обычно обладают весьма ограниченными сведениями о смысле хранящихся в них данных. Однако в РМД присутствуют семантические (смысловые) аспекты. Их примерами являются домены, первичные и внешние ключи [13]. В связи с этим семантическое моделирование данных представляет теоретический и практический интерес. Этим проблемам посвящены работы как зарубежных [11 − 13], так и российских ученых [54 − 57].

Это моделирование осуществляется на этапе инфологического проектирования РБД. При этом выполняются следующие шаги:

− выявляется множество смысловых понятий;

− выделяются сущности предметной области;

− выполняется классификация сущностей;

− определяются свойства сущностей;

− формируются связи между сущностями.

Широкое распространение получил метод семантического моделирования “сущность-связь” или ER-модель [58]. Связи в модели “сущность-связь” могут иметь тип “один к одному”, “один ко многим”, “многие к одному” и “многие ко многим”. Одной из существенных проблем, которая возникает при формировании ER-диаграмм, является назначение связей.

В работах [59-61] предложены эффективные методы формирования всех типов связей между РТ, которые формируются на основе ИТВ. В связи с этим вопросы формирования связей между РТ в данной работе не рассматриваются.

1.2. Определение понятия информации табличного вида (ИТВ) и мотивы разработки способа преобразования ИТВ в реляционное представление

1.2.1. Понятие информации табличного вида

Во введении приведено неформальное определение ИТВ. В связи с тем, что это понятие является основополагающим в данной работе его необходимо обсудить более детально.

Под ИТВ подразумевается информация, которая воспринимается ее потребителями как таблицы. Данные − это информация, фиксированная в регламентированной форме [16, 52]. Под ИТВ в работе понимается информация, фиксированная только внешним видом табличного представления. Поэтому ИТВ не являются данными в полном смысле этого понятия. С другой стороны, в ИТВ отражена и семантика данных в форме заголовков таблиц [53].

Представление информации в табличном виде настолько удобно, что это определило появление класса систем, ориентированных на работу с

табличной информацией − электронных таблиц. Таких как Super Calc, Lotus, Excel[65].

Форма представления ИТВ может быть самой различной: на бумаге, в виде файла текстового редактора, в виде файла текстового процессора, в виде электронных таблиц и др. Вид представления ИТВ также самый разнообразный. В частности, разделителями столбцов и строк могут быть обычные символы, специальные символы, непечатаемые символы, линии или пробелы. Разделители могут и отсутствовать. В этом случае подразумевается, что пользователь может интуитивно выделить строки и столбцы. [5]. Информация, принадлежащая заголовкам и информационным единицам, может располагаться в одной или нескольких строках. ИТВ может быть сгруппирована в соответствии с различными критериями, и тогда внутри таблиц появляются подзаголовки. Информация, расположенная в ячейках одного столбца, не всегда одного типа. Например, дата завершения проекта может быть указана и в формате даты и в текстовом формате. Строго говоря, информация такого вида часто не является данными − информацией, представленной в регламентированном виде. Это обуславливают не всегда однозначное восприятие и интерпретацию информации ее потребителями, а также зачастую приводит к невозможности ее автоматизированной обработки без принятия комплекса мер по ее преобразованию [37].

На рис. 1.1 приведен фрагмент реальной таблицы каталога выставки собак “Союза кинологических обществ России”, сформированной в текстовом редакторе.

ЙОРКШИРСКИЙ ТЕРЬЕР

КЛАСС ЩЕНКОВ САМЦЫ

001 **БЕНИ САНДР`С ВАЛЕНТАЙМ**, щ/к, р.01.09.04, сталь с золотом, вл. Смирнов В., м. Сантана Готсенд Ланд К

002 **ГАЙ ЮЛИЙ КРИСС**, щ/к, р.20.07.04, сталь с золотом, вл. Кузьмичев И.В., о. W.F`s Blue Krystal Kascade, м. Верджиния Ач Норд

003 **МАЙКЛ АЧ НОРД**, щ/к, р.21.07.04, сталь с золотом, вл. Белова Н.И., о. Ozmilion Emperor, м. BHK Behold A Bit of Blue

004 **МАРК АЧ НОРД**, щ/к, р.21.07.04, сталь с золотом, вл. Токарева Н.Д., о. Ozmilion Emperor, м. BHK Behold A Bit of Blue

005 **СТЕФАН GOLDEN LABEL**, щ/к, р.30.09.04, сталь с золотом, вл. Губанов А.П., о. Ozmilion Anticipation, м. Ридана

КЛАСС ЩЕНКОВ САМКИ

006 **БЕТТИ САНДР`С ВАЛЕНТАЙМ**, щ/к, р.01.09.04, сталь с золотом, вл. , м. Сантана Готсенд Ланд К

007 **БЭЛЛА**, щ/к, р.24.08.04, сталь с золотом, вл. Лаврова И.В., о. Ozmilion Emperor, м. Галатея

008 **ГАБРИЛЛЬ КРИСС**, щ/к, р.20.07.04, сталь с золотом, вл. Кузьмичев И.В., о. W.F`s Blue Krystal Kascade, м. Верджиния Ач Норд

009 **EBA DI**, щ/к, р.26.11.04, сталь с золотом, вл. Волкова Г.В., о. Ozmilion Anticipation, м. Дениза Вивальди

010 **КАНА**, щ/к, р.06.07.04, сталь с золотом, вл. Ермолаева Л.С., о. Ozmilion Anticipation, м. Крошка Барби

011 **ЛИЗИ МАРДИ ГРАСС**, щ/к, р.27.11.04, сталь с золотом, вл Неганова Н.А., о. Ozmilion Emperor, м. Лили Марди Грасс

012 **ЛОЛА МАРДИ ГРАСС**, щ/к, р.27.11.04, сталь с золотом, вл Нагайцева М., о. Ozmilion Emperor, м. Лили Марди Грасс

013 **ЛЯЛЯ МАРДИ ГРАСС**, щ/к, р.27.11.04, сталь с золотом, вл Карасева С.Ю., о. Ozmilion Emperor, м. Лили Марди Грасс

014 **САНТАНА GOLDEN LABEL**, щ/к, р.30.09.04, сталь с золотом, вл. Рыбакова А.А., о. Ozmilion Anticipation, м. Ридана

015 **СТЕФАНИ GOLDEN LABEL**, щ/к, р.30.09.04, сталь с золотом, вл. Петрова Д.А., о. Ozmilion Anticipation, м. Ридана

016 **ШАНЕЛЬ ЛАМБЕРГ**, щ/к, р.22.08.04, сталь с золотом, вл. Кинах Н.М., о. Ozmilion Anticipation, м. Хиллари Ламберг

017 **DELFINA OD KA MI**, PKR.III-52031, р.08.07.04, сталь с золотом, вл. Ивонина Е.В., о. Orby De Penghibur, м. Giorgia Ad Operam

КЛАСС ЮНИОРОВ САМЦЫ

018 **АЙРОН ВИГАС**, щ/к, р.05.01.04, сталь с золотом, вл. Рюмина Е., о. Ozmilion Anticipation,

019 **БЭКХЕМ ROYAL DAN**, щ/к, р.07.06.04, сталь с золотом, вл. Азарова М., о. Ozmilion Anticipation, м. Аэлита Сандра

КЛАСС ЮНИОРОВ САМКИ

020 **БРИТАНИКА ВИКТОРИЯ ROYAL DAN**, RUS04119150, р.07.06.04, сталь с золотом, вл. Расторгуева Е., о. Ozmilion Anticipation, м. Аэлита Сандра

021 **ЕВГЕНИЯ ИЗ ХРУСТАЛЬНОГО ДВОРЦА**, щ/к, р.20.04.04, сталь с золотом, вл. Тусеева Л., о. Song Na Niepogode, м. Axehead King Texla

022 **ЕЛЕТТА НАТАЛИ**, щ/к, р.28.01.04, сталь с золотом, вл. Аршинов А., о. W.F`s Blue Krystal Kascade, м. Ксения Натали

023 **ЕЛИЗАВЕТА НАТАЛИ**, щ/к, р.28.01.04, сталь с золотом, вл. Бистяйкина Н., о. Ozmilion Emperor, м. Елизавета

КЛАСС МОЛОДЫХ САМЦЫ

024 **ВИКОНТ ЛУИ**, щ/к, р.12.04.03, сталь с золотом, вл. Климкина Т.А., о. Луинни, м. Кэтрин Зета Джон

Рис. 1.1. Фрагмент таблицы каталога выставки собак “Союза кинологических обществ России”.

В таблице нет разделителей заголовков, нет разделителей строк. Эти разделители интуитивно воспринимаются человеком, но представляют малоразрешимую проблему для выделения записей с помощью компьютера. В таблице присутствуют подзаголовки, причем двух уровней – породы собак и их классы. Документ подготовлен в Word и включает в себя несколько сотен страниц. В Союзе кинологических обществ России появилась потребность создания БД, которая позволила бы хранить каталоги за прошедшие годы. Заполнение вручную таблиц БД информацией такого рода – задача нетривиальная.

Еще более неудачная ИТВ, с точки зрения ее преобразования в РТ, фрагмент реальной таблицы продаж оборудования, представленная на рис. 1.2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		Продажа модернизации - месяц						Продажа модернизации - YTD						Коммерческие предложения - месяц						К
2		Объем		Единицы		Цели		Объем		Единицы		Цели		План		Факт		Цели		Пла
3	Region/branch					Объем	ед.					Объем	ед.	ед.	Объем	ед.	Объем	ед.	Объем	ед.
4		План	Факт	План	Факт	000 руб.		План	Факт	План	Факт	000 руб.			000 руб.		000 руб.		000 руб.	
5	MSR	12500,0	52,6	66,7	1,0	14750,0	77,2	1375,0	52,6	6,0	1,0	3150,0	15,0	75,8	16666,7	35,0	8379,0	178,0	39166,7	62,5
6	РУ-1	250,0	0,0	1,0	0,0	600,0	2,8	750,0	0,0	3,0	0,0	1800,0	8,5	11,4	2500,0	26,0	7309,0	27,3	6000,0	34,1
7	Б.Кочуров	0,0		0,0		0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0			0,0	0,0	0,0
8	Д.Сапожников	83,3		0,3		158,3	0,8	250,0	0,0	1,0	0,0	475,0	2,3	3,8	833,3			7,2	1583,3	11,4
9	А.Трофимов	83,3		0,3		150,0	0,8	250,0	0,0	1,0	0,0	450,0	2,3	3,8	833,3	7,0	1973,0	6,8	1500,0	11,4
10	О.Базыкин (new)	41,7		0,2		145,8	0,7	125,0	0,0	0,5	0,0	437,5	2,0	1,9	416,7	11,0	3554,0	6,6	1458,3	5,7
11	А.Москвитин (new)	41,7		0,2		145,8	0,7	125,0	0,0	0,5	0,0	437,5	2,0	1,9	416,7	8,0	1782,0	6,6	1458,3	5,7
12	РУ-2	208,3	52,6	1,0	1,0	450,0	2,2	625,0	52,6	3,0	1,0	1350,0	6,5	9,5	2083,3	9,0	1070,0	20,5	4500,0	28,4
13	А.Моисеев	0,0		0,0		0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0			0,0	0,0	0,0
14	С.Санникова	104,2		0,5		154,2	0,8	312,5	0,0	1,5	0,0	462,5	2,3	4,7	1041,7			7,0	1541,7	14,2
15	В.Левин	104,2	52,6	0,5	1,0	154,2	0,8	312,5	52,6	1,5	1,0	462,5	2,3	4,7	1041,7	9,0	1070,0	7,0	1541,7	14,2
16	Вакансии-1	0,0		0,0		141,7	0,7	0,0	0,0	0,0	0,0	425,0	2,0	0,0	0,0			6,4	1416,7	0,0
17	РУ-3	250,0	0,0	1,0	0,0	600,0	2,8	0,0	0,0	0,0	0,0	0,0	0,0	11,4	2500,0	0,0	0,0	27,3	6000,0	0,0
18	А.Ямашкин	0,0		0,0		0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0			0,0	0,0	0,0

Рис 1.2. фрагмент таблицы продаж оборудования

В этой таблице задействованы заголовки и подзаголовки 3-х уровней, 1-й столбец используется в качестве двухуровневого подзаголовка.

На рис 1.3 приведен результат импорта данной таблицы в формат СУБД Access. Как видно из рисунка получены маловразумительные данные, которые в таком состоянии не могут быть обработаны. Следует отметить, что в процессе импорта возникали ошибки, а мощность исходной таблицы несколько тысяч записей. И это все приводит к дополнительным трудностям преобразования.

Код	Поле1	Продажи моде	Поле3	Поле4	Поле5	Поле6	Поле7	Поле8	Поле9	Поле10	Поле11	Продажи моде	П
1		План 2001 (тыс					Цели 2001 (тыс					План 1кв.	
2	Region/branch				Другие					Другие		Объем	ед.
3		всего	коммерч.	муницип.	города	Ед.	всего	коммерч.	муницип.	города	Ед.	000 руб.	
4	MSR	150000,0	20000,0	130000,0	0,0	800,0	177000,0	47000,0	130000,0	0,0	926,0	37500,0	200,0
5	РУ-1	3000,0	3000,0	0,0	0,0	12,0	7200,0	7200,0	0,0	0,0	34,0	750,0	3,0
6	Б.Кочуров	0,0					0,0					0,0	0,0
7	Д.Сапожников	1000,0	1000,0			4,0	1900,0	1900,0			9,0	250,0	1,0
8	А. Трофимов	1000,0	1000,0			4,0	1800,0	1800,0			9,0	250,0	1,0
9	О.Базыкин (ne	500,0	500,0			2,0	1750,0	1750,0			8,0	125,0	0,5
10	А.Москвитин (r	500,0	500,0			2,0	1750,0	1750,0			8,0	125,0	0,5
11	РУ-2	2500,0	2500,0	0,0	0,0	12,0	5400,0	5400,0	0,0	0,0	26,0	625,0	3,0
12	А.Моисеев	0,0					0,0					0,0	0,0
13	С.Санникова	1250,0	1250,0			6,0	1850,0	1850,0			9,0	312,5	1,5
14	В.Левин	1250,0	1250,0			6,0	1850,0	1850,0			9,0	312,5	1,5
15	Вакансии-1	0,0					1700,0	1700,0			8,0	0,0	0,0
16	РУ-3	3000,0	3000,0	0,0	0,0	12,0	7200,0	7200,0	0,0	0,0	34,0	750,0	3,0
17	А.Ямашкин	0,0					0,0					0,0	0,0
18	Ю.Фомин	1000,0	1000,0			4,0	1900,0	1900,0			10,0	250,0	1,0
19	Е.Зензина (nev	750,0	750,0			3,0	1800,0	1800,0			8,0	187,5	0,8
20	М.Ануров (new	750,0	750,0			3,0	1750,0	1750,0			8,0	187,5	0,8
21	Б.Петричко (r	500,0	500,0			2,0	1750,0	1750,0			8,0	125,0	0,5
22	РУ-4	1500,0	1500,0	0,0	0,0	5,0	3600,0	3600,0	0,0	0,0	17,0	375,0	1,3
23	С.Мытарев	0,0					0,0					0,0	0,0

Рис. 1.3. Результат импорта в Access таблицы продаж оборудования

В качестве последнего, но далеко не единственного примера, приведем фрагмент реальной таблицы оборудования, сформированной в формате Excel (Рис.1.4).

10.01.01	ШК	Шварев С.	3	Люкс			2500$	90%	сл.нед.
10.01.01	ШК	Алпатова И.	3	Бизнес	13VTR-M	507,00		90%	сл.нед.
10.01.01	ШК	Алпатова И.	3	Люкс	13VTR-M	549,00		90%	сл.нед.
10.01.01	РУ	Шварев С.	3	Бизнес		94,00		90%	сл.нед.
10.01.01	ШК	Шварев С.	3	Люкс			2500$	90%	сл.нед.
11.01.01	РУ	Шварев С.	5	Люкс	13VTR-M	767,00		60%	06.02.2001
11.01.01	РУ	Козловская Л.	5	Люкс	13VTR-M	1 307,00		60%	06.02.2001
10.01.01	ШК	Алпатова И.	3	Люкс	13VTR-M	549,00		90%	сл.нед.
12.01.01	ШК	Алпатова И.	2	Стандарт	13VTR-M	287,00		90%	сл.нед.
16.01.01	ШК	Алпатова И.	5	Бизнес	13VTR-M	324,00		80%	06.02.2001
16.01.01	ШК	Козловская Л.	2	Люкс	13VTR-M	1 232,00		60%	06.02.2001
22.01.01	ШК	Алпатова И.	3	Люкс		137,00		70%	06.02.2001
22.01.01	ШК	Козловская Л.	2	Люкс		120,00		60%	06.02.2001
15.01.01	РУ-2	Левин В.	3	Люкс	13VTR	550,00			21.05.2001
25.01.01	РУ-2	Левин В.	4	Бизнес	13VTR	430,00			06.03.2001
26.01.01	РУ-2	Левин В.	2	Стандарт	13VTR	864,00			22.03.2001
08.01.01	РУ-3	Зензина Е.	3	Бизнес	13VTR	400,00			26.03.2001
15.01.01	РУ-3	Зензина Е.	6	Бизнес	13VTR	400,00		Сентябрь 01	30.03.2001
19.01.01	РУ-3	Зензина Е.	6	Стандарт	13VTR	350,00		Октябрь 01	30.03.2001
16.01.01	РУ-4	Родченков Н.	8	Бизнес	13VTR-M	270,00		на рассмотр.	16.04.2001

Рис.1.4. Фрагмент таблицы оборудование

Здесь последний и предпоследний столбцы представлены доменами, в которых внесены данные различных типов. Действительно, в предпоследнем столбце представлены данные процентного и символьного типов, в последнем столбце представлены данные типа дата и символьного типов.

После преобразования данной таблицы в Access (рис 1.5) казалось бы, что все произошло удачно. Однако предпоследний столбец сформировался символьного типа, а напрашивается числовой. Последний столбец также сформировался символьного типа, а по логике вещей должен быть тип дата. Ведь сортировка по датам в данном случае невозможна! К тому же в процессе импорта возникло более 200 ошибок преобразования.

Код	Поле1	Поле2	Поле3	Поле4	Поле5	Поле6	Поле7	Поле8	Поле9	Поле10
1	10.01.2001	ШК	Шварев С.	3	Люкс			2500$	90%	сл.нед.
2	10.01.2001	ШК	Алпатова И.	3	Бизнес	13VTR-M	507		90%	сл.нед.
3	10.01.2001	ШК	Алпатова И.	3	Люкс	13VTR-M	549		90%	сл.нед.
4	10.01.2001	РУ	Шварев С.	3	Бизнес		94		90%	сл.нед.
5	10.01.2001	ШК	Шварев С.	3	Люкс			2500$	90%	сл.нед.
6	11.01.2001	РУ	Шварев С.	5	Люкс	13VTR-M	767		60%	06.02.2001
7	11.01.2001	РУ	Козловская Л.	5	Люкс	13VTR-M	1307		60%	06.02.2001
8	10.01.2001	ШК	Алпатова И.	3	Люкс	13VTR-M	549		90%	сл.нед.
9	12.01.2001	ШК	Алпатова И.	2	Стандарт	13VTR-M	287		90%	сл.нед.
10	16.01.2001	ШК	Алпатова И.	5	Бизнес	13VTR-M	324		80%	06.02.2001
11	16.01.2001	ШК	Козловская Л.	2	Люкс	13VTR-M	1232		60%	06.02.2001
12	22.01.2001	ШК	Алпатова И.	3	Люкс		137		70%	06.02.2001
13	22.01.2001	ШК	Козловская Л.	2	Люкс		120		60%	06.02.2001
14	15.01.2001	РУ-2	Левин В.	3	Люкс	13VTR	550			21.05.2001
15	25.01.2001	РУ-2	Левин В.	4	Бизнес	13VTR	430			06.03.2001
16	26.01.2001	РУ-2	Левин В.	2	Стандарт	13VTR	864			22.03.2001
17	08.01.2001	РУ-3	Зензина Е.	3	Бизнес	13VTR	400			26.03.2001
18	15.01.2001	РУ-3	Зензина Е.	6	Бизнес	13VTR	400		Сентябрь 01	30.03.2001
19	19.01.2001	РУ-3	Зензина Е.	6	Стандарт	13VTR	350		Октябрь 01	30.03.2001
20	16.01.2001	РУ-4	Родченков Н.	8	Бизнес	13VTR-M	270		на рассмотр.	16.04.2001

Рис 1.5. Результат импорта в Access таблицы оборудование

Перевод такой таблицы в формат реляционной таблицы базы данных в автоматическом режиме принципиально невозможен.

Таким образом, ИТВ в общем случае обладает следующим свойствами.

1. ИТВ − это информация, которая воспринимается ее потребителями, как таблицы.
2. В ИТВ могут отсутствовать разделители строк и разделители столбцов.
3. Элементы данных таблицы могут размещаться в нескольких строках.
4. Типы элементов данных одного столбца могут не совпадать.
5. Заголовки ИТВ могут включать в себя подзаголовки нескольких уровней.
6. В ИТВ могут быть задействованы внутренние подзаголовки.
7. Имена заголовков могут совпадать.
8. Первый столбец ИТВ может быть задействован как многоуровневый заголовок.
9. В ИТВ возможны пустые строки.
10. В ИТВ могут отсутствовать заголовки столбцов.
11. В ИТВ возможно дублирование строк.

В рамках способа преобразования ИТВ в реляционные таблицы необходима разработка подходов и соответствующих алгоритмов и средств

преобразования таблиц ИТВ к виду РТ, приемлемому для их автоматизированной обработки, которые позволят использовать преимущества наличия ИТВ и свести к минимуму дефекты проектирования РБД.

1.2.2. Мотивы и проблемы разработки способа формирования реляционных таблиц на основе использования ИТВ

Немало ИТВ, как показано в предыдущем параграфе, представлено в текстовом формате или в формате текстовых процессоров или еще чаще в формате электронных таблиц. В связи с новыми задачами нередко потребности пользователей этой информации таковы, что им удобнее работать с ней, используя средства современных СУБД [15].

Большая часть ИТВ сегодня хранится в формате электронных таблиц (ЭТ). Это обусловлено рядом достоинств ЭТ. В связи с этим ЭТ широко распространены и, если необходима работа с информацией табличного вида, в первую очередь обращаются к каким-либо средствам работы с ЭТ [53]. Иногда это оправданно, но зачастую, когда таблицы построены и в них размещены большие объемы информации, возникает необходимость использования возможностей БД. В связи с этим возникает проблема переноса накопленной информации в БД. Преимущества БД по сравнению с ЭТ обсуждаются в ряде работ, в частности в [26, 37].

В частности, БД по сравнению с электронными таблицами обладают следующими преимуществами, приведенными ниже.

- БД позволяет не только вводить данные в таблицы, но и контролировать правильность вводимых данных.
- В БД гораздо проще, чем в ЭТ построить пользовательский интерфейс.
- Если все необходимые для работы данные хранить в ЭТ, то необходимо вести каталоги таблиц, в которых легко запутаться.

- В БД возможно создание связей между таблицами.
- В БД обеспечивается контроль дублирования данных.
- БД в отличие от ЭТ обычно ориентировано на работу с несколькими пользователями одновременно.
- БД имеет развитую систему защиты от несанкционированного доступа.
- В БД предусмотрены развитые средства администрирования.
- В БД реализуются специальные средства защиты и восстановления данных.

В ЭТ имеются специфические особенности представления информации, которые недопустимы в БД. Так, например, в заголовках ЭТ могут встречаться символы недопустимые с точки зрения БД, заголовки и подзаголовки могут различаться только цветом или видом шрифта. Например, на рис. 1.6 приведен реальный фрагмент таблицы выполнения работ. В ней помимо сложных заголовков в именах заголовков используются символы ”.”, а это недопустимо во многих БД.

	A	B	C	D	E	F	G	H	I	J	K	L	M
4		Заказчик			Ген.подрядчик			Инвестор			Архитектор		
5	**Заказчик**	Ф.И.О.	Телефон, факс	Дата посл. Переговор.	Ф.И.О.	Телефон, факс	Дата посл. Переговор.	Ф.И.О.	Телефон, факс	Дата посл. Переговор	Ф.И.О.	Телефон, факс	Дата посл. Переговор.
6	Хохтиф				Хохтиф			Конти					
7	Хохтиф				Хохтиф			Конти					
8	Хохтиф				Хохтиф			Конти					
9	Хохтиф												
10	Хохтиф												
11	Хохтиф												
12	Прогрес				Прогрес			Газпром					
13	Прогрес				Прогрес			Газпром					
14	а/п Домодедово				CYVAS								
15	а/п Домодедово				CYVAS								
16	KGM										KGM		
17	KGM										KGM		
18	KGM										KGM		
19	KGM										KGM		
20	ФЕКА												
21	ЕНКА Минфин												

Рис. 1.6. Фрагмент таблицы выполнения работ

Как следует из сказанного выше, при импорте в БД ИТВ и в частности при импорте ЭТ, могут возникнуть ошибки различных типов. Исправление ошибок процесс трудоемкий и не всегда приводит к успеху.

Кроме того, после преобразования ИТВ в РТ возникает проблема назначения первичных и внешних ключей.

В связи вышесказанным можно сделать выводы:

- С точки зрения пользователей ИТВ мотивами преобразования ИТВ в РТ является потребность использования преимуществ БД.
- С точки зрения разработчика способа преобразования ИТВ в РТ мотивом его разработки являются перечисленные выше преимущества РТ перед ИТВ.

Резюмируя сказанное, можно сделать вывод о том, что проблема разработки методики преобразования ИТВ в РТ актуальна и представляют теоретический и практический интерес.

1.3. Анализ применимости современных теоретических и практических разработок

Анализ применимости современных теоретических разработок

Как отмечают специалисты в области проектирования и использования БД на протяжении всего существования реляционных БД они постоянно предлагали наилучшую смесь простоты, устойчивости, гибкости, производительности, масштабируемости и совместимости в сфере управления данными [44].

Это связано с наличием стройной методологии разработки реляционных БД [11, 17, 29, 43, 46, 47]. Руководствуясь ею, разработчик может успешно выполнить все этапы проектирования, включая постановку и анализ требований, концептуальное проектирование, датологическое и физическое проектирование.

Методология позволяет решить проблемы нормализации таблиц БД, назначить ключевые поля, сформировать связи между таблицами.

Но даже основоположники реляционного подхода утверждают, что формализовать процесс проектирования РБД пока не удается [13]. Это

связано с тем, что все выкладки делаются на основе ручного анализа предполагаемых схем отношений. Но в этих схемах возможны ошибки, которые проявятся, может быть, только в процессе эксплуатации спроектированной БД. Но ничего лучшего на сегодня нет. Реляционная методология проектирования БД является одной из лучших на сегодня методологий. И не использовать ее компоненты, по меньшей мере, не разумно.

Ситуация существенно меняется, когда при проектировании БД используются не гипотетические схемы отношений, а реальные, заполненные таблицы. Вот здесь и можно формализовать процесс проверки соответствия данных реляционным таблицам, процесс назначения ключевых полей, процесс нормализации таблиц, процесс формирования связей таблиц. Ведь все это можно сделать на основе автоматизированного анализа реальных данных (если, конечно, данные такого рода есть). В этом случае наиболее полно и эффективно можно использовать традиционную методологию. Ведь при решении всех вопросов разработчик отталкивается от реальных данных, и все рекомендации методологии проектирования РБД могут быть в полной мере формализованы.

Поэтому методика, разрабатываемый в диссертации, базируется на теории проектирования РБД.

Анализ применимости современных практических разработок

Во всех СУБД предусмотрены средства импорта данных. При этом данные могут быть представлены в различных форматах. В том числе и в форматах .txt и .xls. Но как показывает практика при импорте таблиц, которые хотя бы частично не удовлетворяют требованиям к РТ, возникает множество ошибок. И исправление этих ошибок очень трудоемко, а иногда и не представляется возможным.

Так, например, в системе Oracle предусмотрено средство для перемещения данных - SQL*Loader [66, 67]. При перемещении данных из другого приложения формируются файлы отвергнутых и некорректных

записей. Некорректные записи − это записи, которые по каким-либо причинам не могут быть включены в БД. Это говорит о том, что проблема преобразования информации табличного вида в файлы БД актуальна, а с другой стороны, эта проблема при использовании данной утилиты не всегда имеет корректное решение.

В Access есть развитые средства импорта данных. Но при импорте, казалось бы, самых “безобидных” таблиц, как правило, формируется файл ошибок, который зачастую включает в себя сотни записей.

Microsoft SQL Server [69, 70] также обеспечивают импорт данных. Но предварительно эти данные должны быть представлены в форме реляционных таблиц.

Таким образом, все современные СУБД можно использовать для импорта данных. Причем импортируемые данные могут быть представлены в различных форматах.

Однако использование средств импорта ограничено. Их оправдано использовать, когда таблицы имеют небольшую мощность и степень. Тогда необходимые исправления впоследствии можно внести вручную. Кроме того средства импорта оправдано использовать, когда импортируемые таблицы максимально приближены к реляционному представлению.

В качестве итога можно сделать вывод. При незначительных размерах таблиц (десятки полей и сотни записей) преобразования ИТВ в РТ можно выполнять “вручную” – импортировать данные, а затем ввести нераспознанные отложенные записи и выполнить корректировку импортированных записей. При больших размерах таблиц ИТВ (сотни полей и тысячи записей) целесообразно использовать автоматизированные средства. Это подтверждает опыт разработок и экспертные исследования.

1.4 Постановка задачи разработки способа формирования реляционных таблиц на основе использования ИТВ

Состав способов, алгоритмов и средств, разрабатываемых в рамках методики

Сравнительный анализ неформальных описаний РТ и ИТВ, выполненный выше, позволили определить состав способов, основных алгоритмов и средств, ориентированных на преобразование ИТВ в РТ.

В рамках способа преобразования нереляционных таблиц ИТВ в реляционные таблицы необходимо разработать следующие алгоритмы:

- исключения заголовков внутри таблиц;
- исключения подзаголовков внутри таблицы;
- исключения сложных подзаголовков;
- исключения пустых подзаголовков;
- избавления от вертикальных подзаголовков;
- приведения значений атрибутов к одному типу;
- исключения дублирования записей;
- исключения пустых записей;

При этом в отличие от алгоритмов, предложенных в работах [5, 37, 53]:

− рассматриваются возможные сочетания подзаголовков (внешние, внутренние, вертикальные);

− рассматриваются пустые заголовки;

− расширяется понятие вертикальных подзаголовков;

− анализируется проблема исключения пустых записей;

− детально анализируется проблема приведения значений доменов к одному типу.

Кроме того, все алгоритмы, предложенные в работах [5, 37, 53], уточнены, дополнены и представлены более детально и наглядно.

В рамках способа назначения ключевых полей в преобразованных таблицах ИТВ необходимо разработать следующие алгоритмы:

- выявления домена с уникальными значениями его элементов;

- выявления сочетания доменов с уникальными сочетаниями соответствующих им элементов;
- поиска минимальных первичных ключей, включающих в себя один атрибут;
- поиска минимальных первичных ключей, включающих в себя несколько атрибутов;
- выявления атрибутов, которые входят в первичный ключ и содержат уникальные значения;
- выявления внешних ключей.

При этом в отличие от алгоритмов, предложенных в работах [5, 37, 53]:

− выявление первичных ключей включаются в этап преобразования ИТВ в РТ и тем самым обеспечивается одно из требований к РТ;

− выявление сочетания доменов с уникальными сочетаниями соответствующих им элементов выполняется более тщательно и детально;

− поиск минимальных первичных ключей, включающих в себя несколько атрибутов, выполняется в соответствии со всеми требованиями к минимальности ключей;

− выявление внешних ключей включаются в этап преобразования ИТВ в РТ и тем самым на ранних этапах проектирования РБД решается важная задача.

Кроме того, все алгоритмы, предложенные в работах [5, 37, 53], разработаны заново, дополнены и представлены более детально и наглядно.

С учетом вышесказанного построена схема иерархии разрабатываемых алгоритмов − рис. 1.8.

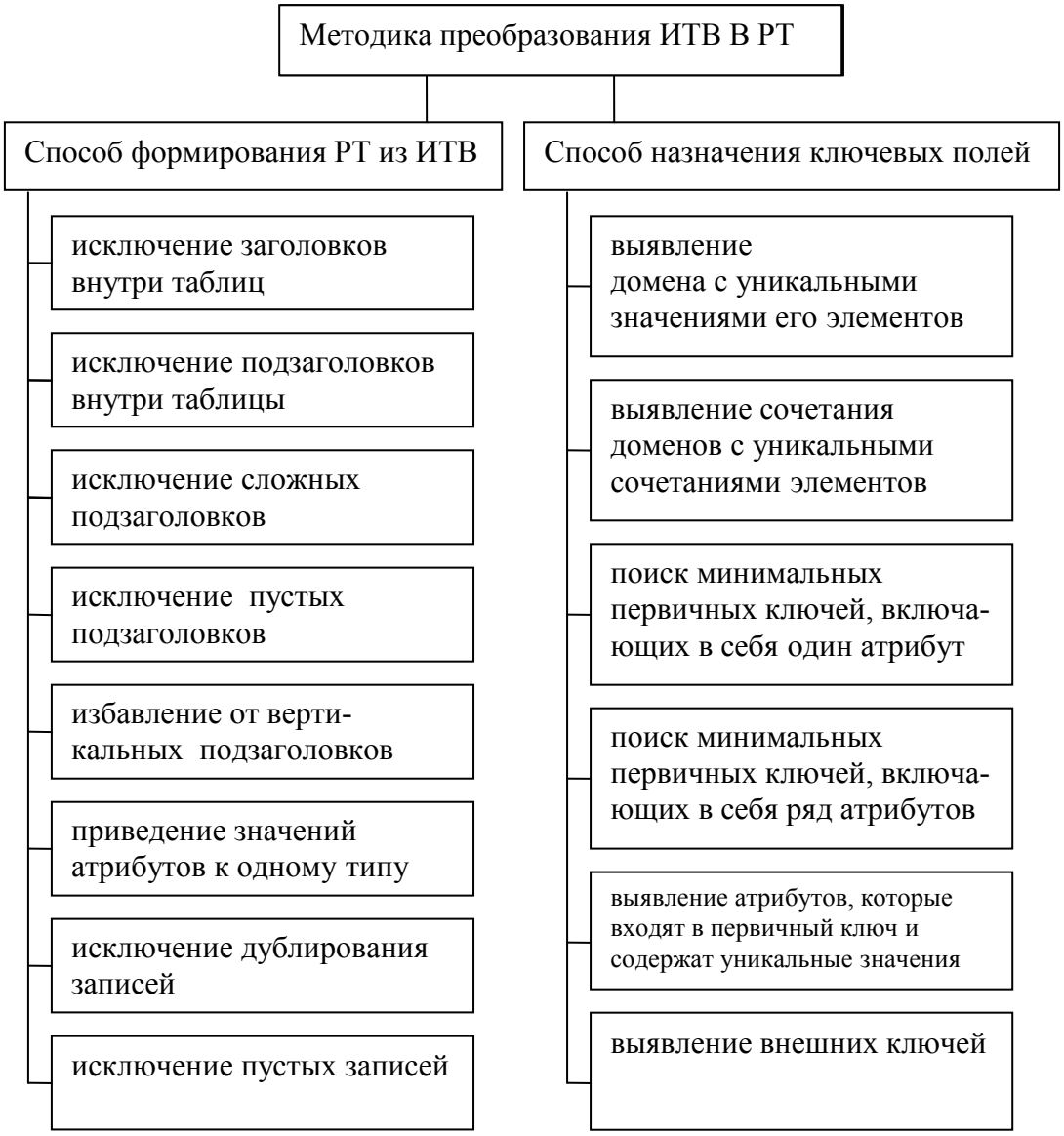


Рис. 1.8. Схема иерархии разрабатываемых алгоритмов

Важно отметить, что разрабатываемые алгоритмы – человеко-машинные. То есть предполагается активное участие разработчика РБД в ходе преобразования ИТВ в РТ. Разработчику предоставляется возможность принимать решения по выбору последовательности действий и выбору вариантов проектных решений, предлагаемых подсистемой. Эта возможность в конечном итоге сведет к минимуму ошибки проектирования и позволит в полном объеме использовать опыт разработчика.

Проблемам автоматизированного проектирования посвящено ряд работ, в частности [70 − 73]. Следует учитывать общие закономерности

автоматизированного проектирования, которые изложены в соответствующей литературе.

Выводы по главе 1

1. Традиционная теория проектирования РБД это лучшее, что сегодня имеется для компактных, непротиворечивых БД.
2. При наличии ИТВ, возможна формализация преобразования ИТВ в РТ на основе анализа имеющейся информации.
3. Способ преобразования ИТВ в РТ органично сочетается с традиционной методологией проектирования РБД.
4. ИТВ представляет собой информацию, которая интерпретируется заинтересованными в ней людьми двумерными таблицами.
5. ИТВ не удовлетворяют требованиям к реляционным таблицам.
6. Потребители ИТВ заинтересованы в использовании преимуществ РБД при обработке информации.
7. Разработчики РБД заинтересованы в наличии автоматизированных средств преобразования ИТВ в РТ.
8. Для преобразования ИТВ в РТ необходимо решить следующие задачи:
   - приведение ИТВ к реляционному виду;
   - назначение первичных и внешних ключей.
9. Современная методология проектирования РБД может быть использована для преобразования ИТВ в РТ как основа для формализации алгоритмов преобразования.
10. Комплексная методика преобразования ИТВ в РТ в настоящее время в полном объеме не разработана.
11. Современные практические разработки в области импорта, реляционных таблиц могут быть использованы только для таблиц,

представленных в реляционном виде, или как вспомогательные средства для решения задач малой размерности.

2. СПОСОБ ПРЕОБРАЗОВАНИЯ ЗАПОЛНЕННЫХ НЕРЕЛЯЦИОННЫХ ТАБЛИЦ В РЕЛЯЦИОННЫЕ ТАБЛИЦЫ

Во второй главе разработаны модели объектов исследования (РТ и ИТВ), способ преобразования таблиц ИТВ к реляционному виду. В рамках способа рассмотрены алгоритмы приведения атрибутов заполненных таблиц к единому типу, исключения дублирования записей в ИТВ, избавления от сложных атрибутов и исключения простых и сложных подзаголовков.

2.1 Модели объектов исследования

2.1.1 Модель реляционных таблиц

Практически во всех работах посвященных РБД сформулированы требования к РТ. Однако эти требования, как правило, представляют собой перечисление свойств, которыми должны обладать таблицы. Требования не формализованы, да они и не могут быть формализованы, т.к. РБД проектируются на основе схем отношений. Например, схема отношения Студент может иметь вид: Студент = (Фамилия, Имя, Отчество, Дата рождения, Дата поступления, Адрес,…, Средний балл). Требования же предъявляются к данным, а данных-то во многих случаях еще и нет. Они появятся потом в процессе эксплуатации БД, и может оказаться, что сформулированные на начальных этапах проектирования требования к РТ не удовлетворяются. В частности, отталкиваясь от схемы отношения Студент, вполне можно предположить, что сочетание Фамилия, Имя, Отчество вполне подходит для первичного ключа. Однако, как показывает опыт в рамках даже одной организации, нередко встречаются полные тезки. А основным требованием к первичному ключу является его уникальность.

Нередко исходная информация присутствует в виде ИТВ. В этом случае ситуация существенно меняется в лучшую сторону. Ведь на основе реальных данных вполне возможно формализовать требования к ним.

В работе [5] описана формальная модель РТ. Основные ее черты следующие. Реляционная таблица *(RT)* представляется множеством

$RT=\{Z, D\}$,

где Z – множество заголовков, D - множество данных.

$z = \{z_1, \ldots, z_i, \ldots, z_n\},\ i = 1, n;\ n >= 1,$ (1)

где n - степень множества заголовков.

Должно быть обеспечено условие $z_i \neq z_m$, $i =1, n$; $m = 1, n$; $i \neq m$, (2)

где n – степень множества заголовков, т.е. недопустимо совпадение заголовков.

$D = \{SD\}$, где SD – множество строк данных. (3)

$SD = \{SD_1, \ldots, SD_i, \ldots, SD_n\}, i = 1, n;\ n >> 1,$ (4)

где n - мощность множества строк данных.

$SD_i = \{ED_{i1}, \ldots, ED_{ij}, \ldots, ED_{ik}\},\ j = 1, k;\ k >= 1,$

где k - степень множества i-ой строки данных, ED_{ij} – элемент данных.

Недопустима ситуация, когда внутри таблицы данных могут встретиться заголовки, т.е. должно выполнятся условие:

$SD_i \neq z_j, i = 1, n;\ n >> 1;\ j = 1, k;\ k >= 1,$ (5)

где n - мощность множества строк данных, k - степень множества заголовков.

Для реляционных таблиц выполняется правило:

$(\forall ED)\ (ED \in SD)\ (\exists z(z \in Z)\ (z \leftrightarrow ED))$ (6)

Т.е. каждому элементу данных соответствует только один заголовок.

$(\forall ED)\ (ED \in SD\ )\ (\exists (TED)\ \ (ED \leftrightarrow TED))$, (7)

где $TED = string\ V\ integer\ V\ datetime\ V\ real\ V\ logical.$ Т.е. каждому элементу данных соответствует определенный тип данных.

В реляционных таблицах обязательно выполнение следующего требования:

$TED_{11} =, \ldots, = TED_{i1} =, \ldots, = TED_{n1}$

... ...

$TED_{1j} =, \ldots, = TED_{ij} =, \ldots, = TED_{nj}$

… … … … … … … … … … …

$TED_{1k} =, \ldots, = TED_{ik} =, \ldots, = TED_{nk}, i = 1, n; n >> 1; j = 1, k;\ k >= 1,$ (8)

где n - мощность множества строк данных, k - степень множества i-ой строки данных; ED_{ij} – элемент данных. Другими словами, значения типов данных одного столбца должны совпадать.

Недопустима ситуация, когда $SD_i = SD_j, i = 1, n; j = 1, n; i \neq j,$ (9)

где n – мощность множества данных. Т.е. невозможно полное совпадение строк данных.

Однако в данном случае отражены не все свойства РМД, которые в числе прочих предполагается взять за основу в данной работе.

Несмотря на достоинства этой модели, она не в полном объеме отражает особенности РТ в рамках темы диссертации и соответственно не может в полной мере служить исходной формализацией для способа, разрабатываемого в диссертации.

В реляционной модели данных [5] не предусмотрено наличие первичных ключей.

Пусть $SD_i = \{ED_{i1}, \ldots, ED_{ij}, \ldots, ED_{ik}\},\ j = 1, k;\ k >= 1,$

где SD_i - i-я строка данных множества, k - степень множества, ED_{ij} – элемент данных.

Тогда должен быть найден такой заголовок Z_i ($Z_i \leftrightarrow ED_{ij}$), такой чтобы выполнялось выражение:

$ED_{i1 \ldots} \neq {}_{\ldots} ED_{ij \ldots} \neq {}_{\ldots} ED_{in}$, где n – мощность множества *(11)*

Таким образом, в таблице должен присутствовать такой столбец, все значения которого бы отличались. Такой столбец можно использовать в качестве первичного ключа.

Кроме того, в качестве первичного ключа могут использоваться сочетания атрибутов таблицы.

В реляционной модели данных не должно быть пустых заголовков:

$Z_i \neq \varnothing$ *(12)*

В реляционной модели данных не должно быть пустых строк:

$SD_i \neq \varnothing$ *(13)*

В реляционной модели данных содержимое никакого столбца не могут использоваться как подзаголовки. *(14)*

В реляционной модели данных не должно быть сочетаний различного типа подзаголовков.

2.1.2 Модель информации табличного вида

Если в предыдущем параграфе рассматривалось целевая модель данных РТ, то здесь разговор пойдет об исходной модели, которую необходимо преобразовать в форму РТ – модель ИТВ.

Модель ИТВ в известной литературе, как правило, не рассматривается. В работе [53] представлена неполная формализованная модель ИТВ, которая с модификациями может быть принята в качестве базовой для решения задач диссертации. Рассмотрим эту модель.

Информация табличного вида (ИТВ) представляется множеством DT = {Z, D}, где Z – множество заголовков ИТВ, D - множество данных, соответствующим заголовкам . (15)

$Z = \{Z_1, \ldots, Z_2, \ldots, Z_n\},\ i = 1, n;\ n >= 1,$ *(16)*

где n – степень множества заголовков.

Допустима ситуация, когда $Z_i = Z_m,\ i = 1, n\ ;\ m = 1, n;\ i \neq m\ ,$ *(17)*

где n – степень множества заголовков, т.е. возможно полное совпадение заголовков

В данных табличного вида возможны подзаголовки 1-го уровня, что формально выглядит следующим образом.

$Z_i = \{PZ_{i1}, \ldots, PZ_{ij}, \ldots, PZ_{ik}\},\ j = 1, k;\ k >= 2,$ (18)

где k - степень множества подзаголовков i-го заголовка.

$Z_p = \{PZ_{p1}, \ldots, PZ_{pt}, \ldots, PZ_{pm}\},\ t = 1, m;\ m >= 2,$ (19)

где m – степень множества подзаголовков p-го заголовка.

Допустима ситуация, когда $PZ_{ij} = PZ_{pt}$. (20)

В данных табличного вида возможны подзаголовки 2-го уровня, что формально выглядит следующим образом.

$PZ_{ij} = \{PPZ_{ij1}, \ldots, PPZ_{ijm}, \ldots, PPZ_{ijf}\}$, m = 1, f; f >= 2, (21)

где f – степень множества подзаголовков 2-го уровня ij-го подзаголовка 1-го уровня.

$PZ_{pt} = \{PPZ_{pti}, \ldots, PPZ_{ptr}, \ldots, PPZ_{ptg}\}$, r = 1, q; q >= 2, (22)

где q – степень множества подзаголовков 2-го уровня pt-го подзаголовка 1-го уровня

Допустима ситуация, когда $PPZ_{ijm} = PPZ_{ptm}$. (23)

Теоретически в данных табличного вида может быть больше уровней подзаголовков.

На рис. 2.1 приведен реальный пример таблицы ИТВ с подзаголовками.

	№ п/п	Дата	Город	Продавец	Заказчик	Сегмент	Количество	Тип оборудования							Тип конт-	Цена	
Регион		выдачи предложения			(организация)		лифтов	Купе кабины	Лебедка	С-ма управлен.	Двери шахты	Вызывная и сигнализ. аппар	Устр-ва безоп-ти	Другое	ракта (M, M+L, L)	в тыс. руб.	или $
0	1	02.01.01	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
MSR	E2S68-MSR	10.01.01	ШК	Шварев С.	Частное лицо	2	1							фотореверс	M+L		1 750
MSR	E2S69-MSR	10.01.01	ШК	Аллатова	Централ.те	3	1	Бизнес	13VTR-M	KLM-VF	Ств. ДШ	Бизнес	ВКЛ		M+L	507,00	
MSR	E2S70-MSR	10.01.01	ШК	Аллатова	Централ.те	3	1	Люкс	13VTR-M	KLM-VF	Ств. ДШ	Люкс	ВКЛ		M+L	549,00	
MSR	E2S71-MSR	10.01.01	ШК	Аллатова	Централ.те	3	1	Бизнес				Бизнес	НЕ ВКЛ		M+L	94,00	
MSR	E2S72-MSR	10.01.01	ШК	Аллатова	Централ.те	3	1	Люкс				Люкс	НЕ ВКЛ		M+L	149,00	
MSR	E2S73-MSR	11.01.01	ШК	Козловская	ОАО	5	2	Люкс	13VTR-M	KLM-1	Ств. ДШ	Люкс	ВКЛ		M+L	767,00	
MSR	E2S74-MSR	11.01.01	ШК	Козловская	ОАО	5	2	Люкс	13VTR-M	KLM-VF	Ств. ДШ	Люкс	ВКЛ		M+L	1 307,00	
MSR	E2S75-MSR	11.01.01	ШК	Шварев С.	Госкомрезе	2	2	Бизнес			Ств. ДШ	Люкс	НЕ ВКЛ		M+L	388,00	
MSR	E2S76-MSR	12.01.01	ШК	Аллатова	Централ.те	2	1	Стандарт	13VTR-M	KLM-1	Ств. ДШ	Бизнес	ВКЛ		M+L	287,00	
MSR	E2S77-MSR	16.01.01	ШК	Аллатова	ГНП	5	1	Бизнес	13VTR-M	KLM-1	Ств. ДШ	Бизнес	ВКЛ		M+L	324,00	
MSR	E2S78-MSR	16.01.01	ШК	Козловская	Министер.к	2	3	Люкс	13VTR-M	MCS-220	Ств. ДШ	Люкс	ВКЛ		M+L	1 232,00	
MSR	E2S79-MSR	22.01.01	ШК	Аллатова	"Импексбан	3	1	Люкс				Люкс	НЕ ВКЛ		M+L	137,00	
MSR	E2S80-MSR	22.01.01	ШК	Козловская	Министер.ф	2	1	Люкс				Люкс	НЕ ВКЛ		M+L	120,00	
MSR	E2S81-MSR	15.01.01	РУ-2	Левин В.	"Рособорон	3	3	Люкс	13VTR	KLM-B		Люкс	ВКЛ		M+L	550,00	
MSR	E2S82-MSR	25.01.01	РУ-2	Левин В.	Г-ца	4	2	Бизнес	13VTR	KLM-1	Ств. ДШ	Бизнес	ВКЛ		M+L	430,00	
MSR	E2S83-MSR	26.01.01	РУ-2	Левин В.	МГУ ПС	2	2	Стандарт	13VTR	KLM-B	Ств. ДШ	Стандарт	ВКЛ		M+L	864,00	
MSR	E2S84-MSR	08.01.01	РУ-3	Зензина Е.	ЗАО	3	1	Бизнес	13VTR	KLM	Пластик	Бизнес		нет	M+L	400,00	

Рис. 2.1. Пример таблицы ИТВ с подзаголовками

Здесь заголовок ”Тип оборудования” включает в себя 7 подзаголовков, заголовок “Цена” включает в себя 2-а подзаголовка.

D = {SD, Z}, где SD – множество строк данных. (24)

Такого рода представление D допускает наличие нескольких заголовков и подзаголовков 1-го и 2-го уровней, расположенных в области данных.

В том числе допускается наличие заголовков и подзаголовков, расположенных до, после и между строк данных. На рис. 2.2 приведен реальный пример таблицы ИТВ с заголовками внутри области данных. Таковыми являются строки 1 и 9 и др. Кроме того, на рис. 2.2 представлен внутренний подзаголовок более высокого уровня - “Собрать свой компьютер”.

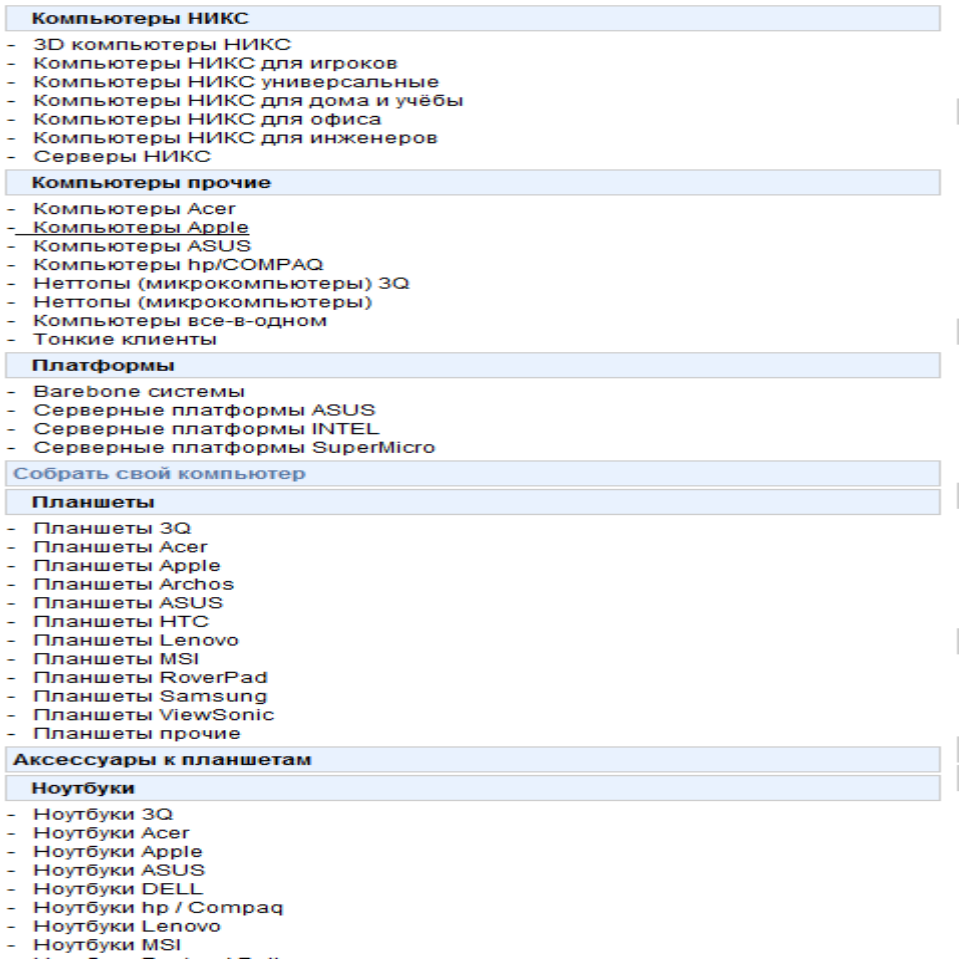


Рис. 2.2. Пример таблицы ИТВ с заголовками внутри области данных

$$SD = \{SD_1, \ldots, SD_i, \ldots, SD_n\},\ i = 1, n;\ n >> 1, \quad (25)$$

где n - мощность множества строк данных.

$$SD_i = \{ED_{i1}, ...,ED_{ij}, \ldots, ED_{ik}\},\ j = 1, k;\ k >= 1, \quad (26)$$

где k – степень множества элементов данных i-ой строки данных;

ED_{ij} – элемент данных.

Для информации табличного вида должно выполняться следующее правило: (∀ED) (ED ∈ SD) (∃ z (z ∈ Z) (z↔ED)) V (∃ (PZ)) (z ∈PZ) (PZ ↔ ED) V (∃ (PPZ)) (PPZ ∈ PZ) (PPZ ↔ ED) (27)

Т.е. каждому элементу данных соответствует заголовок или подзаголовок 1-го или 2-го уровней и наоборот.

(∀ED) (ED ∈ SD) (∃TED (TED ∈ T(ED))), (28)

где TED = string V integer V datetime, где T(ED) – тип ED.

Но каждому элементу данных соответствует какой-либо тип данных.

В общем случае:

$TED_{11}\neq, \ldots, \neq TED_{i1}\neq, \ldots, \neq TED_{n1}$

... ...

$TED_{1j}\neq, \ldots, \neq TED_{ij}\neq, \ldots, \neq TED_{nj}$

... ...

$TED_{1k}\neq, \ldots, \neq TED_{ik}\neq, \ldots, \neq TED_{nk}$, $i = 1, n; n >> 1; j = 1, k; k >= 1$, (29)

где n - мощность множества строк данных, k - степень множества элементов i-ой строки данных. Другими словами, значения типов данных одного столбца могут не совпадать.

Допустима ситуация, когда $SD_i = SD_j$, i = 1, n ; j = 1, n; i ≠ j , (30)

где n – мощность множества данных, т.е. возможно полное совпадение строк данных.

В модели ИТВ могут быть пустые заголовки:

$Z_i = \varnothing$ (31)

В модели ИТВ могут быть пустые строки:

$SD_i = \varnothing$ (32)

В ИТВ содержимые столбца могут использоваться как подзаголовки таблиц. (33)

В ИТВ могут быть сочетания различного типа подзаголовков (34)

Несмотря на некоторое сходство модели таблиц ИТВ и модели РТ, в них имеются существенные различия.

Сравним выражения для РТ и ИТВ.

1. Выражения (17-24) для ИТВ противоречат выражению (2) для РТ.
2. Выражения (27-28) для ИТВ противоречат выражению (5) для РТ.
3. Выражение (31-33) для ИТВ противоречат выражению (12-14) для РТ.
4. Выражение (29) для ИТВ противоречат выражению (8) для РТ.
5. Выражение (30) для ИТВ противоречат выражению (9) для РТ.

Кроме того, в исходной модели РТ не предусмотрено наличие первичного ключа. Соответственно отсутствуют требования к первичному ключу.

Основной задачей работы и является разработка способа и средств, которые бы позволили преобразовать ИТВ в РТ, ликвидировав указанные различия в их моделях.

2.2 Проблема приведения значений атрибутов заполненных таблиц к одному типу

2.2.1 Типы полей в реляционных таблицах

В РТ основные типы полей: числовой, текстовый, дата-время. В БД, если в каком-либо поле таблицы РТ вводятся новые значения, тип значения проверяется автоматически. В случае, если введенное значение не соответствует назначенному типу, выдается сообщение об ошибке. При заполнении таблиц ИТВ проверки типов вводимых значений не производятся. Поэтому в нереляционных таблицах в одном и том же столбце могут храниться данные различных типов. Эти недопустимо в РБД.

Поэтому необходимы средства преобразования каждого столбца элементов таблиц ИТВ к одному типу.

В качестве примера с различными типами значений элементов в одноименных столбцах рассмотрим таблицу, представленную на рис. 2.3.

ТаблицаПор : таблица

	Имя	Заработок	Возраст	День свадьбы	Должность
▶	Федоров	1700$	34 года	09-03-1997	Инженер
	Алексей	нет работы	17 лет	неженатый	Студент
	Андрей	1100$	45 лет	12-11-1995	Продавец
	Елена	1500$	27 лет	23-04-2009	Преподаватель
	Ольга	нет работы	16 лет	незамужняя	Студентка
	Инна	2200$	45 лет	02-10-1995	Врач
*					

Рис. 2.3. Таблица с различными типами значений атрибутов в одноименных столбцах

В этой таблице в столбце ”Заработок” должны располагаться значения денежного типа, а в столбце “День свадьбы” – значения типа ”Дата/Время”. Но в этой таблице тип всех столбцов текстовый, т. е. имеет место смешение типов.

Для таблицы с незначительным числом записей и атрибутов несложно исправить значения и свойства атрибутов. Однако для больших объемов данных это неприемлемо. В связи с этим предлагается автоматизированный способ назначения типов атрибутов заполненных таблиц.

Суть алгоритма состоит в следующем. В каждом столбце проверяются все значения, которые не соответствуют назначенному типу, который имеют большинство значений столбца. Если значение столбца в соответствии с контекстом может быть отнесено к назначенному типу, то это значение преобразуется в формат, принятый для данного типа. Запоминается позиция преобразованного значения и его прежнее значение. Прежнее

значение и его номер записываются в соответствующую таблицу. Если значение столбца в соответствии со своим контекстом не может быть отнесено к назначенному типу, его номер заносится в таблицу непреобразованных значений атрибута.

2.2.2. Преобразование значений атрибутов заполненных таблиц к одному типу

При преобразовании ИТВ в РТ значения атрибутов заполненных таблиц ИТВ в каждом столбце необходимо привести к одному типу.

Рассмотрим пример на Access. В примере задана простейшая структура двух таблиц – для ТаблицыТекст и ТаблицыДеньги. Каждая таблица имеет одно поле. В ТаблицеТекст поля типы – ” Текстовый ”, в ТаблицеДеньги поля типы – ”Денежный”. В обеих таблицах имена полей одинаковые – ”Деньги”. ТаблицаДеньги не заполнена, а ТаблицаТекст заполнена данными на рис. 2.4.

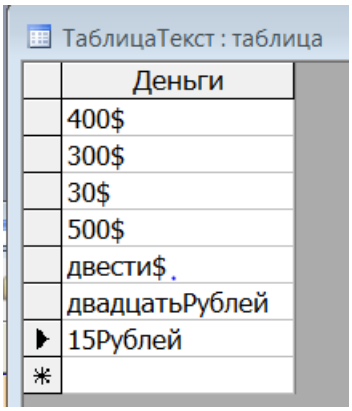


Рис.2.4. Таблица Текст с данными

Можно добавить данные из ТаблицыТекст в ТаблицуДеньги. Для этого сформирован следующий запрос.

INSERT INTO ТаблицаДеньги

SELECT ТаблицаТекст.*

FROM ТаблицаТекст;

Здесь конструкция «INSERT INTO» ТаблицаДеньги указывает на данные, которые добавляются ко всем полям таблицы “ ТаблицаДеньги ”. Конструкция “SELECT ТаблицаТекст.* FROM ТаблицаТекст;” указывает на то, что данные для добавления выбираются из всех полей “*” таблицы ”ТаблицаТекст ”.

В результате выполнения этого запроса будет выведено сообщение, приведенное на рис.2.5.

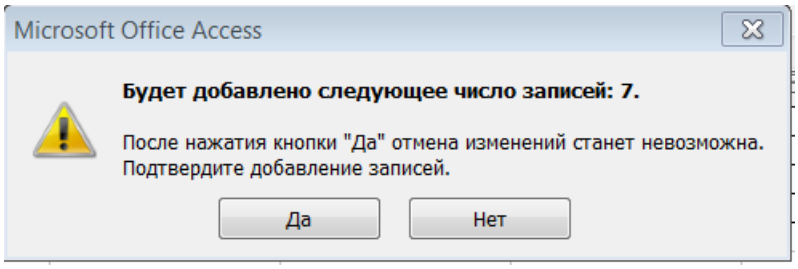


Рис.2.5. Первое сообщение, сформированное после выполнения запроса на добавление.

Затем сформируется сообщение на рис.2.6.

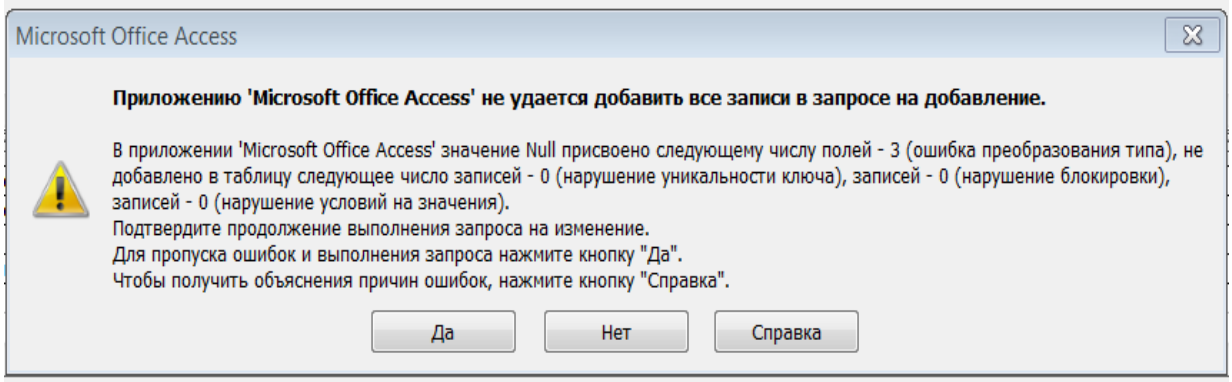


Рис.2.6. Второе сообщение, сформированное после выполнения запроса на добавление

В результате выполнения этого запроса ТаблицаДеньги примет следующий вид, представленный на рис. 2.7.

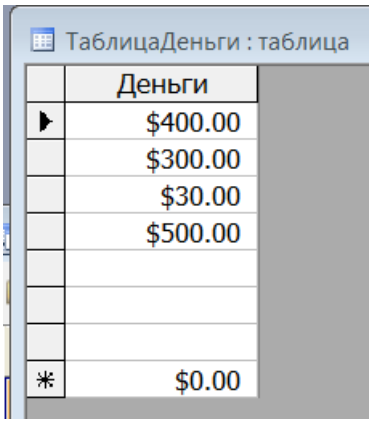


Рис. 2.7. Результат выполнения запроса на добавление

После сравнения рис. 2.4 и рис. 2.7 нетрудно сделать вывод о том, что, несмотря на различные формы представления денег, большая часть значений успешно преобразовалась. Не преобразовались только последние три значения, которые явно на деньги не похожи, но в реальных таблицах такого рода ”деньги” зачастую имеют место.

В связи с этим необходимы средства, использование которых позволило бы преобразовывать элементы столбцов ИТВ к одному типу. Реализация этих средств разработана в приложении 1.

2.3 Исключение дублирования записей

В РТ недопустимо дублирование записей, а в ИТВ это возможно. В связи с этим необходимы средства исключения дублирования записей в ИТВ.

2.3.1 Типы дублирования записей

Перед разработкой средств исключения дублирования записей в ИТВ рассмотрим типы дублирования. Рассмотрим, как происходит дублирование записей в таблице, представленной на рис.2.8.

ТаблицаДур : таблица

№ Паспорт	Имя	факультет	№ Группа	Предмет	Оценка
117449	Алексей	ИУ-6	122	БД	5
224576	Андрей	ЯК-7	233	Алгебра	4
115779	Анна	ИУ-1	126	КИС	5
117449	Алексей	ИУ-6	122	БД	5
114577	Марина	ИУ-3	223	РИС	4
0			0		0

Рис.2.8 . Таблица с дублированиями записей

Здесь повторяющиеся записи выделены. В РТ такая ситуация недопустима. Поэтому повторяющиеся записи необходимо оставить в единственном экземпляре.

Чтобы выявить повторяющиеся записи, можно сделать запрос на Microsoft Access, который имеет следующий вид.

SELECT First(ТаблицаДуп.[№ Паспорт]) AS [№ Паспорт поле], First(ТаблицаДуп.Имя) AS [Имя поле], First(ТаблицаДуп.факультет) AS [факультет поле], First(ТаблицаДуп.[№ Группа]) AS [№ Группа поле], First(ТаблицаДуп.Предмет) AS [Предмет поле], First(ТаблицаДуп.Оценка) AS [Оценка поле], Count(ТаблицаДуп.[№ Паспорт]) AS Повторы

FROM ТаблицаДуп

GROUP BY ТаблицаДуп.[№ Паспорт], ТаблицаДуп.Имя, ТаблицаДуп.факультет, ТаблицаДуп.[№ Группа], ТаблицаДуп.Предмет, ТаблицаДур.Оценка

HAVING (((Count(ТаблицаДуп.[№ Паспорт]))>1) AND ((Count(ТаблицаДуп.Оценка))>1));

Этот запрос занимает немало строк, но он несложен по сути : из таблицы “ ТаблицаДур” выбираются значения номера паспорта (First(ТаблицаДур.[№

Паспорт])), и этим значениям присваивается имя (AS [№ Паспорт поле]) . Выводимые данные группируются по этим значениям (конструкция GROUP BY). С помощью конструкции HAVING выводятся только те строки, в которых имеются повторения (((Count(ТаблицаДуп.[№ Паспорт]))>1) AND ((Count(ТаблицаДуп.Оценка))>1)).

Результат выполнения запроса на выборку повторяющихся записей представлен на рис.2.9. В отдельном столбце ”Повторы” выводятся количества повторений соответствующих записей в таблице.

Поиск повторений для ТаблицаДур : запрос на выборку

№ Паспорт пол	Имя поле	факультет пол	№ Группа поле	Предмет поле	Оценка поле	Повторы
117449	Алексей	ИУ-6	122	БД	5	2

Рис.2.9. Результат выполнения запроса на выборку повторяющихся записей

В Microsoft Access запрос такого рода строится просто – с помощью специального мастера ”Повторяющиеся записи”.

Иногда в базах данных происходит дублирование записей другого типа, как в таблице, представленной на рис.2.10.

ТаблицаДур1 : таблица

№ Паспорт	Имя	Факультет	№ Группа	Предмет	Оценка
117449	Алексей	ИУ-6	122	БД	5
224576	Андрей	ЯК-7	233	Алгебра	4
115779	Анна	ИУ-1	126	КИС	5
117450	Алексей	ИУ-6	222	БД	5
114577	Марина	ИУ-3	223	РИС	4
0			0		0

Рис.2.10. Таблица с дублированием записей

На этом рисунке нетрудно увидеть, что не все записи дублируются. В таблице «ТаблицаДуп1» столбцы «Имя, Факультет, Предмет, Оценка» имеют дублированные записи, а столбец «№ паспорта» не дублируются. Если в столбце «№ паспорта» номер отличается, то такая запись не считается дублированной. Для этой таблицы запрос на SQL имеет следующий вид.

SELECT First(ТаблицаДуп1.[№ Паспорт]) AS [№ Паспорт поле], First(ТаблицаДуп1.Имя) AS [Имя поле], First(ТаблицаДуп1.Факультет) AS [Факультет поле], First(ТаблицаДуп1.[№ Группа]) AS [№ Группа поле], First(ТаблицаДуп1.Предмет) AS [Предмет поле], First(ТаблицаДуп1.Оценка) AS [Оценка поле], Count(ТаблицаДуп1.[№ Паспорт]) AS Повторы

FROM ТаблицаДуп1

GROUP BY ТаблицаДуп1.[№ Паспорт], ТаблицаДуп1.Имя, ТаблицаДуп1.Факультет, ТаблицаДуп1.[№ Группа], ТаблицаДуп1.Предмет, ТаблицаДуп1.Оценка

HAVING (((Count(ТаблицаДуп1.[№ Паспорт]))>1) AND ((Count(ТаблицаДуп1.Оценка))>1));

В результате выполнения запроса нет дублирования записей, что представлено на рис 2.11.

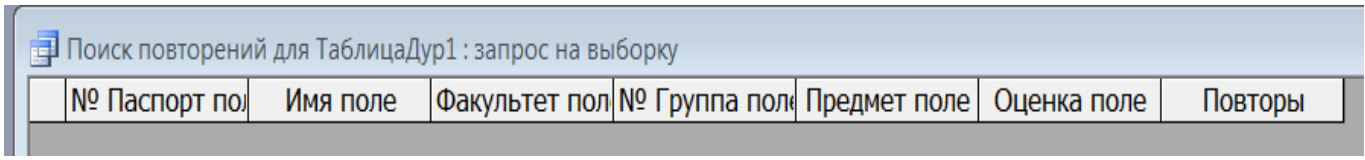


Рис.2.11. Результат выполнения запроса

2.3.2. Удаление дублирования записей

Если допускается переименование таблицы с повторяющимися записями, то практически во всех СУБД можно задать запрос, который формирует на основе исходной таблицы новую таблицу, но без повторяющихся записей. Запрос для таблицы будет представлен в следующем виде.

```
SELECT DISTINCT ТаблицаДуп.*INTO Без Дуп
FROM ТаблицаДуп;
```

В этом запросе все поля ТаблицаДуп (ТаблицаДуп.*) добавляются к Таблице «Без Дур» (INTO Без Дуп) из ТаблицаДуп (FROM ТаблицаДуп). При этом посредством конструкции ”DISTINCT” в Таблицу «Без Дуп» добавляются только уникальные значения.

После выполнения запроса Таблица «Без Дуп» примет следующий вид (рис. 2.12).


Без Дур : таблица

№ Паспорт	Имя	факультет	№ Группа	Предмет	Оценка
114577	Марина	ИУ-3	223	РИС	4
115779	Анна	ИУ-1	126	КИС	5
117449	Алексей	ИУ-6	122	БД	5
224576	Андрей	ЯК-7	233	Алгебра	4
0			0		0

Рис.2.12. Таблица без повторяющихся записей

В результате получается таблица без повторяющихся записей. Если не допускается переименование таблицы с повторяющимися записями, то рассмотренный запрос на создание новой таблицы тоже может иметь место, но при этом следует обязательно выполнить дополнительные действия. Они

заключаются в удалении исходной таблицы и переименовании полученной таблицы.

2.4. Исключение сложных атрибутов и подзаголовков

В нереляционных таблицах могут встречаться подзаголовки трех типов: внешние подзаголовки, внутренние подзаголовки и подзаголовки-столбцы. В реляционных таблицах подзаголовки недопустимы. Рассмотрим примеры подзаголовков различных типов.

2.4.1. Задача исключения подзаголовков

Внешние подзаголовки часто встречаются в таблицах-справочниках. В качестве примера в таблице 2.1 приведем фрагмент каталога электрооборудования автомобилей (таблица 2.1).

Таблица 2.1

Указатели				Системы зажигания			
Габаритные указатели		Поворотные указатели		Контактные		Бесконтактные	
№	Марка	№	Марка	№	Марка	№	Марка

Как видно из структуры таблицы, она имеет заголовки трех уровней: основной заголовок, подзаголовок первого уровня и подзаголовок второго уровня. В реляционных таблицах допустимы заголовки только одного уровня. Для ИТВ в общем случае справедливы следующие утверждения:

T={З1,З2,…,ЗК,….,ЗN}, ЗК={П1ЗК,П2ЗК,…,ПLЗК,….,ПFЗК}

ПLЗК={П1ПLЗК, П2ПLЗК,…., ПQПLЗК,…., ПРПLЗК},

Где T – схема таблицы;

ЗК – схема К-того заголовка таблицы Т;

ПLЗК – L-ый подзаголовок первого уровня ЗК- го заголовка таблицы Т;

ПQПLЗК – Q-ый подзаголовок второго уровня.

Следует обратить внимание на то, что число уровней подзаголовков может быть и б*о*льшим. Но, как показывает анализ доступных документов различных предметных областей, обычно задействуют два уровня подзаголовков. Это видно из примера, приведенного во фрагменте таблицы крепежных деталей (таблица 2.2).

Таблица 2.2

Болты		Шайбы		Винты				Штифты		Гайки		Шплинты
6-и гранные	Цилиндрические с внутренним 6-й гранником	Плоские	Гравер	с чечевичной головкой	с полусферической головкой	с потайной головкой	с цилиндрической головкой	Простые	Конусообразные	Основные	Коронки	

Для этой таблицы могут быть задействованы заголовки более высокого уровня. Их схема следующая:

Нормаль = {гайки, болты, винты, шайбы}.

Фиксирующие детали = {штифты, шплинты, клинья}.

В обозначениях столбцов типа ПQПLЗК заложены индексы, с помощью которых могут быть построены циклы, содержащиеся в алгоритмах исключения внешних заголовков. Так, например, для основного цикла могут быть задействованы индексы K и N , для 1-го внутреннего цикла индексы L и K , для 2-ого внутреннего индексы Q и P.

Чтобы исключить сложные заголовки и привести таблицу к реляционному виду, необходимо организовать сканирование подзаголовков

самого нижнего уровня и для каждого подзаголовка осуществить сбор всей относящейся к нему информации из всех заголовков более высокого уровня. Затем собранную информацию необходимо использовать в качестве неделимого (атомарного) заголовка.

Например, 1-й полученный таким образом заголовок таблицы 2 будет выглядеть следующим образом: N шестигранные болты или болты шестигранные N. Второй заголовок может быть таким: болты шестигранные тип. Далеко не всегда заголовки, полученные таким образом, могут быть воспринимаемы потенциальным пользователем базы данных. Более того, их длина может превышать максимально допустимую длину атрибутов, используемых инструментальных средств. В связи с этим процедура формирования атомарных столбцов должна быть не автоматической, а автоматизированной. То есть пользователь для исключения внешних подзаголовков должен иметь возможность вмешаться в процесс формирования заголовков столбцов с целью присвоения атомарным столбцам приемлемых имен. Один из возможных алгоритмов исключения внешних подзаголовков приведен в работе [53].

Следует обратить внимание на то, что исходная информация может быть представлена различными способами: на бумаге, в виде текстовых файлов, в формате электронных таблиц и др. Удобнее всего для разработчика средств преобразования было бы использование в качестве исходного единый формат данных. Например, в качестве основного – использование формата Excel, а все другие представления данных преобразовывать в данный формат. Но, к сожалению, не всегда это возможно. Например, табличные данные, представленные на бумаге, далеко не всегда удается отсканировать в формат электронных таблиц.

Поэтому необходимо предусмотреть реализацию алгоритма не только для информации табличного вида, представленной в формате .xls, но и в формате .txt.

2.4.2. Исключение внутренних подзаголовков

Подзаголовки такого рода часто встречаются в прайс-листах, в отчетах по покупке и продаже товаров, то есть в тех случаях, когда структуры нескольких таблиц совпадают, но их данные относятся к различным группам. Группировка может осуществляется, например, по датам, категориям товаров, регионам. В качестве примера рассмотрим список проданных за день товаров магазина «Соки - воды» (таблица 2.3).

Таблица 2.3

Название	Обьем	Цена	Количество
Газированная вода			
Тархун	1л	30р	7
Байкал	2л	60р	5
Колокольчик	1.5л	40р	10
Соки			
Вишневый	1л	45р	3
Ананасовый	1л	45р	15
Яблочный	1л	45р	8
Морсы			
Клюквенный	1.5л	60р	7
Малиновый	1.5л	60р	7
Рябиновый	1л	45р	7
…			

Как видно из примера, таблица легко воспринимается визуально. Однако в таком виде она неприменима в составе баз данных. Такая таблица с небольшим числом строк без особых усилий может быть преобразована в реляционную таблицу вручную. Но в реальных таблицах может быть тысячи или более строк. В этом случае их преобразование невозможно. Необходимы специальные автоматизированные средства.

В случае если таблица 2.3 будет импортирована в какую-либо систему управления базами данных, она примет вид таблицы 2.4.

Таблица 2.4

Название	Обьем	Цена	Количество
Газированная вода			
Тархун	1л	30р	7
Байкал	2л	60р	5
Колокольчик	1.5л	40р	10
Соки			
Вишневый	1л	45р	3
Ананасовый	1л	45р	15
Яблочный	1л	45р	8
Морсы			
Клюквенный	1.5л	60р	7
Малиновый	1.5л	60р	7
Рябиновый	1л	45р	7

Как видно из таблицы 2.4, смысл импортированной таблицы утрачен. В частности, из таблицы следует, что напиток Газированная вода не имеет объема, цены и не продавался. Хотя на самом деле все обстоит по-другому. Необходимо избавиться от противоречий такого рода. Для этого необходимо выполнить следующие действия:

– выявить внутренние подзаголовки;

– присвоить номер очередному напитку;

– сформировать новую таблицу напитков с соответствующими номерами;

– исключить записи с категориями напитков из исходной таблицы;

– сформировать реляционную таблицу или таблицы таким образом, чтобы сохранить смысл исходных таблиц.

Новый заполненный столбец с номерами напитков приведен в таблице 2.5.

Таблица 2.5

№	Название	Обьем	Цена	Количество
1	Газированная вода			
1	Тархун	1л	30р	7
1	Байкал	2л	60р	5
1	Колокольчик	1.5л	40р	10
2	Соки			
2	Вишневый	1л	45р	3
2	Ананасовый	1л	45р	15
2	Яблочный	1л	45р	8
3	Морсы			
3	Клюквенный	1.5л	60р	7
3	Малиновый	1.5л	60р	7
3	Рябиновый	1л	45р	7

В сформированной таблице исключатся записи с напитками из исходной таблицы (табл. 2.6 и табл. 2.7) .

Таблица 2.6

№	Напиток
1	Газированная Вода
2	Соки
3	Морсы

Таблица 2.7

№	Название	Обьем	Цена	Количество
1	Тархун	1л	30р	7
1	Байкал	2л	60р	5
1	Колокольчик	1.5л	40р	10

2	Вишневый	1л	45р	3
2	Ананасовый	1л	45р	15
2	Яблочный	1л	45р	8
3	Клюквенный	1.5л	60р	7
3	Малиновый	1.5л	60р	7
3	Рябиновый	1л	45р	7

Для таблиц данного вида вполне можно строить реляционные запросы, связанные между собой связью типа 1: ∞.

В связи с этим предлагается машинный алгоритм преобразования. Для описания алгоритм используется общий вид отношения, представленный в табл. 2.8.

Таблица 2.8

A_1	……	A_i	……..	A_k
a_{11}	……	NULL	…….	NULL
a_{21}	……	a_{2i}	…….	a_{2k}
…….	……	…….	…….	……..
a_{j1}	……	NULL	…….	NULL
…….	……	…….	…….	……..
a_{f1}	……	NULL	…….	NULL
a_{m1}	……	a_{mi}	…….	a_{mk}

Таблица такого рода – это таблица, в некоторых строках которой значение имеет только один атрибут, который является внутренним подзаголовком таблицы.

Неформальный алгоритм исключения внутренних подзаголовков содержит следующие действия.

П1: Каждая запись проверяется на наличие в ней только одного значения атрибута. Записи такого рода подсчитываются. Если таких записей

несколько, то подзаголовки в отношении R присутствуют и переходят к следующему пункту (П2).

П2: К отношению R присоединится дополнительный атрибут KR с типом «числовой» (формируется R0).

COUNTER :=0;

П3: Сканируются все записи отношения R0 . Если записи имеют только одно заполненное значение атрибута, то счетчик подзаголовков COUNTER увеличивается на 1.

Значению атрибута KR присваивается значение COUNTER.

П4: Создается новое отношение R2, включающее в себя 3-а атрибута FR и атрибут с подзаголовком.

Сканируются все записи отношений R0 . Записи, которые имеют только одно (кроме ключевого) заполненное значение, атрибут перемещается в отношение R2.

На последней стадии алгоритма формируются отношение R2 (с подзаголовками исходной таблицы) и отношение R1 без подзаголовков. Связи между таблицами объединяются ключевыми атрибутами KR, которые присутствуют в обоих отношениях.

Формализованный алгоритм исключения внутренних подзаголовков выглядит следующим образом.

```
COUNTER = 0
FOR s = 1 то m
   COUNTER1 = 0
    FOR f =1 то к
        IF a_sk= NULL THEN COUNTER1 = COUNTER1 + 1
    NEXT f
    IF COUNTER1 = k-1 THEN COUNTER= COUNTER+1
NEXT s
IF COUNTER< 2 THEN EXIT
```

Формирование двух отношений

$R0 = R(A_{1,\ldots,}A_{i,\ldots,}A_k)+R(KR)$

COUNTER = 0

FOR s = 1 то m

COUNTER1 = 0

FOR f =1 то к

IF a_{sk}= NULL THEN COUNTER1 = COUNTER1 + 1

NEXT f

IF COUNTER1 = k-1 THEN

COUNTER = COUNTER + 1

$C(R2_{COUNTER,1})$=COUNTER

$C(R2_{COUNTER,2})= a_{sk}$

DELETE * FROM R0 WHERE ($A_{1=}$ a_{sk})

ELSE

$C(R0_{s,1})$=COUNTER

END IF

NEXT s

Здесь m – мощность R.

k – степень R.

Выражение $R0 = R(A_{1,\ldots,}A_{i,\ldots,}A_k)+R(KR)$ означает добавление к R атрибута с именем KR.

Выражение $C(R2_{COUNTER,1})$ означает значение элемента R2 в строке COUNTER и 1-ом столбце.

Выражение $C(R0_{s,1})$ означает значение элемента R0 в строке s и 1-ом столбце.

В таблице приведен пример с внутренними подзаголовками сложных атрибутов. Рассмотрим таблицу с использованием стандартных средств Microsoft Access, представленную на рис. 2.13.

ТаблицаВнурт : таблица

Название	Обьем	Цена	Количество
Газированная вода			
Тархун	1 л	$30.00	7
Байкал	2 л	$60.00	5
Колокольчик	1.5 л	$40.00	10
Соки			
Вишневый	1 л	$45.00	3
Ананасовый	1 л	$45.00	15
Яблочный	1 л	$45.00	8
Морсы			
Клюквенный	1.5 л	$60.00	7
Малиновый	1.5 л	$60.00	7
Рябиновый	1 л	$45.00	7
		$0.00	0

Рис.2.13. Таблица с внутренними подзаголовками

Нетрудно заметить, что внутренние заголовки «Газированная Вода, Соки, Морсы» стоят в столбце с заголовком «Название».

В базах данных такая таблица невозможна в связи с наличием других таблиц и поэтому надо исключить такие внутренние заголовки. Для приведения таблицы к приемлемому виду можно в режиме « Конструктора» определить новый столбец «Напиток», а потом в режиме «Просмотра» этот столбец вручную заполнить и можно удалить внутренние подзаголовки. Результат такого преобразования представлен на рис. 2.14.

ТаблицаВнурт : таблица

Название	Обьем	Цена	Количество	Напиток
Тархун	1 л	$30.00	7	Газированная вода
Байкал	2 л	$60.00	5	Газированная вода
Колокольчик	1.5 л	$40.00	10	Газированная вода
Вишневый	1 л	$45.00	3	Соки
Ананасовый	1 л	$45.00	15	Соки
Яблочный	1 л	$45.00	8	Соки
Клюквенный	1.5 л	$60.00	7	Морсы
Малиновый	1.5 л	$60.00	7	Морсы
Рябиновый	1 л	$45.00	7	Морсы
		$0.00	0	

Рис. 2.14. Преобразованная таблица

Теперь таблица «ТаблицаВнурт» примет приемлемый вид. Реальные таблицы нередко включает в себя десятки тысяч записей. В этом случае такого рода преобразование таблиц трудоемко и может привести к ошибкам.

2.4.3. Избавление от сложных атрибутов и подзаголовков

Информация табличного вида нередко представлена таким образом, что в области заголовков имеют место заголовки, которые включают в себя несколько позиций. Например, какой-либо заголовок может быть представлен следующим образом: континент, часть света, страна. Таблица такого рода не является реляционной и преобразование ее к реляционному виду задача нетривиальная. Более того, проблема преобразования может существенно усложниться в связи с тем, что такие заголовки влекут за собой необходимость использования сложных заголовков. Например, таблица 2.9.

Таблица 2.9

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Европа			
Восток	Запад		
Россия		33	26
	Испания	8	14

И в первом и во втором случаях нарушена первая нормальная форма – атрибуты реляционных таблиц должны быть неделимы. Кроме того, такого рода подзаголовки могут встречаться внутри таблицы. На рисунке приведен пример такого рода таблицы (таблица 2.10).

Таблица 2.10

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Европа			
Восток	Запад		
Россия		33	26
	Испания	8	14
Азия			
Восток	Запад		

Мьянма		14	4
	Индия	26	8

Такую таблицу невозможно обрабатывать с помощью языка запросов.

В связи с этим эту таблицу оправданно представить в виде 2-х связанных реляционных таблиц: «Части света. Континенты» и «Страны». Но в этой таблице существуют сложные атрибуты, поэтому надо избавляться от сложных атрибутов, избавляться от подзаголовков, которые попали в значения атрибутов. В таблице 2.11 показаны эти изменения.

Таблица 2.11

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Восточная Европа	Западная Европа		
Россия		33	26
	Испания	8	14
Восточная Азия	Западная Азия		
Мьянма		14	4
	Индия	26	8

Предлагается следующая последовательность действий. Формируется новый столбец с номерами частей света, континентов. Сканируется преобразованная таблица, очередной части света, континенту присваивается номер и этот номер распространяется на страны. Но эта таблица не очень удачна для 2-х связанных реляционных таблиц: «Части света. Континенты» и «Страны». Поэтому предлагается следующий вариант – таблица 2.12.

Таблица 2.12

Континент, часть света, страна	Количество крупных городов	Количество крупных рек
Восточная Европа		

Россия	33	26
Западная Европа		
Испания	8	14
Восточная Азия		
Мьянма	14	4
Западная Азия		
Индия	26	8

Результат формирования нового столбца с номерами части света, континентов и заполнением столбца приведен в таблице 2.13.

Таблица 2.13

№	Континент, часть света, страна	Количество крупных городов	Количество крупных рек
1	Восточная Европа		
1	Россия	33	26
2	Западная Европа		
2	Испания	8	14
3	Восточная Азия		
3	Мьянма	14	4
4	Западная Азия		
4	Индия	26	8

Результат формирования новой таблицы «Части света. Континенты» и исключения записей с «частью света, континентов» из исходной таблицы приведены соответственно в таблице 2.14. В таблице 2.15 - Страны.

Таблица 2.14

№	Континент, часть света

1	Восточная Европа
2	Западная Европа
3	Восточная Азия
4	Западная Азия

Таблица 2.15

№	страна	Количество крупных городов	Количество крупных рек
1	Россия	33	26
2	Испания	8	14
3	Мьянма	14	4
4	Индия	26	8

Для таблиц данного вида таблиц можно строить реляционные запросы. Затем из записей с «частью света, континентов» формируется новая таблица, соответствующие записи из исходной преобразованной таблицы удаляются. После этих преобразований будут сформированы 2-е таблицы, связанные между собой связью типа 1: ∞.

Для небольшого рассмотренного примера описанные манипуляции вполне можно выполнить вручную. Но для реальных таблиц мощностью десятки тысячи записей это затруднительно. Поэтому необходима разработка алгоритма автоматизированного избавления от сложных атрибутов при преобразовании заполненных нереляционных таблиц к реляционному виду.

Предварительно представим таблицу (отношение) рассматриваемого типа в общем виде (таблица 2.16).

Таблица 2.16

A_1	A_2	……..	A_i	……..	A_k
a_{11}	NULL	……..	NULL	……..	NULL

a_{21}	a_{22}	…….	NULL	…….	NULL
a_{31}	NULL	……..	a_{3i}	……..	a_{3k}
NULL	a_{42}	……..	a_{4i}	……	a_{4k}
a_{j1}	NULL	……..	NULL	……	NULL
……..	…….	……..	…….	…….	……
a_{f1}	NULL	……..	NULL	…….	NULL
a_{m1}	a_{m2}	.....	a_{mi}	……	a_{mk}

Особенность таблицы такого рода состоит в том, что в некоторых ее строках значение имеет один или два атрибута. Принимается, что такой атрибут является внутренним сложным подзаголовком таблицы.

Предлагается укрупненный алгоритм исключения сложных подзаголовков:

П1: Выполняется сканирование всех записей отношения R. Каждая запись проверяется на наличие в ней только одного значения атрибута. Записи такого рода подсчитываются. Если таких записей несколько, то подзаголовки в отношения R присутствуют и выполняется переход к следующему пункту (П2). В противном случае алгоритм завершает работу.

П2: Избавление от сложных атрибутов:

r=1;

FOR r=1 то m

  FOR i=1 то 2

    WHILE a_{1i}<>a_{2i}

     a_{1i}:=Concat (a_{1i},’ ‘,a_{2i})

  END;

END;

П3: Для 2-х связанных реляционных таблиц:

k=0;

i=1;

j=1;

r=1;

WHILE a_{rj}<>END FILE

  WHILE a_{ij} <> Подзаголовок

    IF k=0 THEN r=1

    k=1;

    $a1_{ij}$:=a_{ij};

    i:=i+1;

  END WHILE;

  k:=0;

  IF j=1 THEN j:=2;

  ELSE j:=1;

END WHILE;

П4: К отношению R приписывается дополнительный атрибут KR с типом «числовой».

COUNTER:=0;

П5: Выполняется сканирование всех записей отношения R’. Если в записи имеется только одно заполненное значение атрибута, то счетчик подзаголовков COUNTER увеличивается на 1.

Значению атрибута KR присваивается значение COUNTER.

П6: Создается новое отношение R2, включающее в себя 2-а атрибута NR и атрибут с подзаголовком.

Выполняется сканирование всех записей отношений R’. Записи, которые имеют только одно (кроме ключевого) заполненное значение атрибута перемещаются в отношение R2.

В результате выполнения алгоритма сформируются отношение R2 (с подзаголовками исходной таблицы) и отношение R1 без подзаголовков. Связи между таблицами обеспечиваются посредством ключевых атрибутов KR, которые присутствуют в обоих отношениях.

Формализованный алгоритм исключения внутренних подзаголовков и избавление от сложных атрибутов при преобразовании нереляционных таблиц к реляционному виду выглядит следующим образом.

```
k=0;
j=1;
r=1;
FOR r=1 то m
 i=1;
 FOR i=1 то 2
   WHILE a1i<>a2i ;
   a1i:=Concat (a1i,' ',a2i);
END
WHILE arj<>END FILE
  WHILE aij <> Подзаголовок
     IF k=0 THEN r=1; k=1;
     a1ij:=aij;
     i:=i+1;
   END WHILE;
   k:=0;
   IF j=1 THEN j:=2;
END WHILE;
COUNTER=0;
FOR r=1 то m
   COUNTER1=0;
   FOR f=1 то p
     IF arp=NULL  THEN COUNTER1=COUNTER1+1;
   NEXT f
   IF COUNTER1=p-1 THEN COUNTER=COUNTER+1;
NEXT r
IF COUNTER < 2 THEN EXIT
```

REM Формирование двух отношений

$R'=R(A_1,\ldots\ldots,A_i,\ldots\ldots,A_p)+R(KR)$

COUNTER=0

FOR r=1 to m

  COUNTER1=0;

  FOR f=1 to p

    IF a_{rp}=NULL THEN COUNTER1=COUNTER1+1;

  NEXT f

 IF COUNTER1 =p-1 THEN

   COUNTER=COUNTER+1;

   $Z(R2_{COUNTER,1})$=COUNTER;

   $Z(R2_{COUNTER,2})=a_{rp}$;

   DELETE * FROM R' WHERE ($A_1=a_{rp}$);

 ELSE

   $Z(R'_{r,1})$=COUNTER;

 END IF

NEXT r;

Здесь:

m – мощность отношения R;

P – степень отношения R;

END FILE - конец таблицы;

Подзаголовок – заголовок 2-го уровня;

Выражение R'=R+R(KR) означает добавление к R атрибута с именем KR;

Выражение $Z(R2_{COUNTER,1})$ означает значение элемента R2 в строке COUNTER и 1-м столбце;

Выражение Z(R'r,1) означает значение элемента R' в строке r и 1-м столбце.

Часто оправдано использовать существующие инструментальные средства для избавления от сложных атрибутов и подзаголовков в

заполненной таблице. Рассмотрим пример на рис. 2.15, где исходная таблица сформирована в Microsoft Excel.

	A	B	C	D
1	Континент, часть света, страна		Количество крупных городов	Количество крупных рек
2	Восточная Европа	Западная Европа		
3	Россия		33	26
4		Испания	8	14
5	Восточная Азия	Западная Азия		
6	Мьянма		14	4
7		Индия	26	8
8				
9				
10				

Рис.2.15. Исходная таблица с избавлением от сложных атрибутов и подзаголовков

В таблице имеется сложный атрибут – «Континент, часть света, страна». Выполним импорт этой таблицы в СУБД Access. Для импорта этой таблицы используется меню «Файл/ Внешние данные/Импорт». В процессе выполнения шагов мастера импорта указывается лист рабочей книги Microsoft Excel, назначается строка заголовки, имя создаваемой таблицы. Окна мастера на его очередных шагах имеют виды рисунки 2.16 - 2.18.

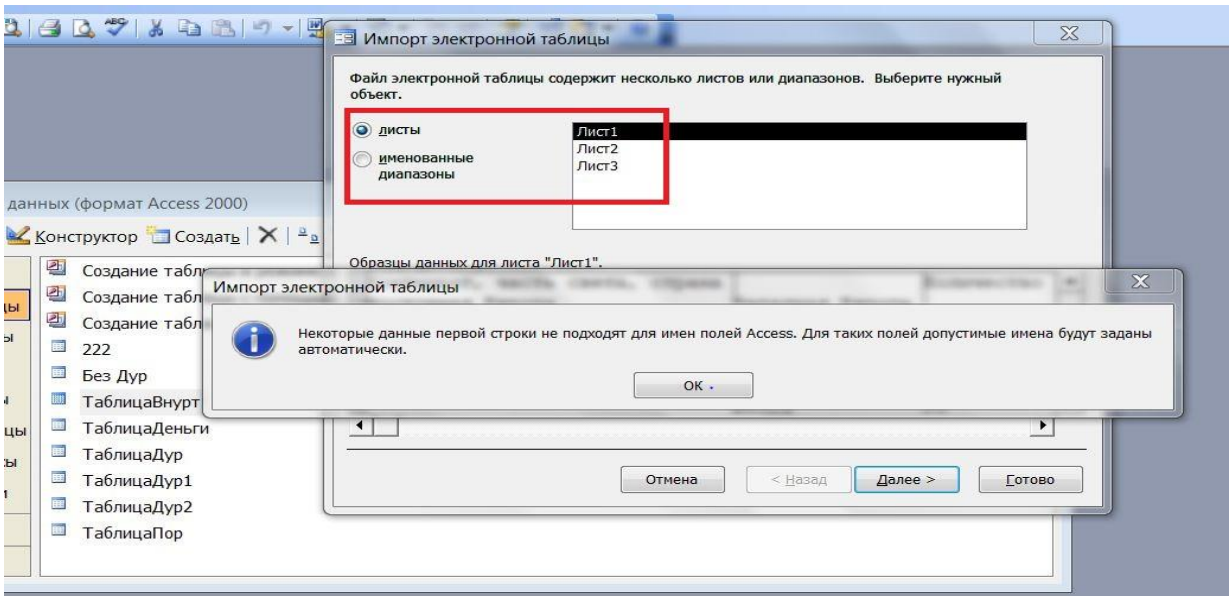


Рис. 2.16. Окно мастера импорта указания листа

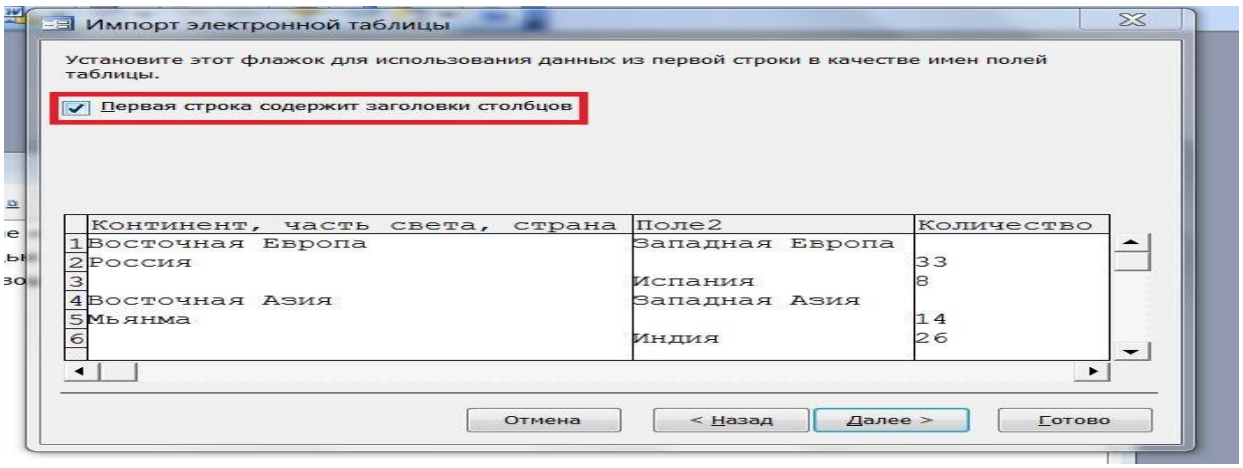


Рис. 2.17. Окно назначения строки заголовков

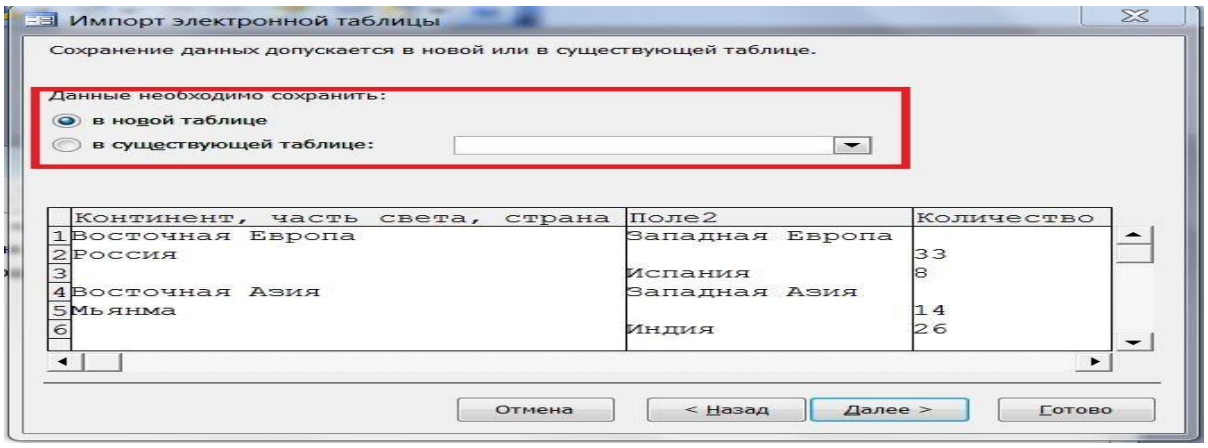


Рис. 2.18. Окно создания имени таблицы

В результате выполнения всех шагов мастера исходная таблица в формате Microsoft Access примет такой вид (рис.2.19).

Континент, часть света, страна	Поле2	Количество крупных городов	Количество крупных рек
Восточная Европа	Западная Евро		
Россия		33	26
	Испания	8	14
Восточная Азия	Западная Азия		
Мьянма		14	4
	Индия	26	8

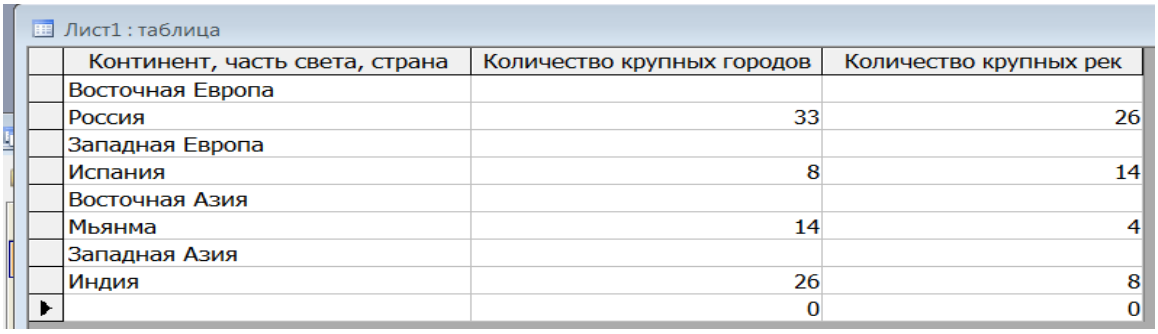
Рис. 2.19. Исходная таблица в формате Microsoft Access

Как видно из рис. 2.19, в таблице нет заголовка второй строки, так как в формате Microsoft Access невозможно импортировать сложные атрибуты и

подзаголовки. Такая таблица неприемлема для использования. Для избавления от сложных атрибутов и подзаголовков необходимо, в соответствии с алгоритмом, сформировать простые заголовки и избавиться от подзаголовков, которые попали в значения атрибутов.

Но эта таблица не очень удачна для связанных реляционных таблиц: «Части света. Континенты» и «Страны». Поэтому надо редактировать таблицу в режиме «Конструктора».

После выполнения необходимых действий в режиме «Конструктора» и в режиме «Просмотра» таблица примет такой вид (рис. 2.20):



Лист1 : таблица

Континент, часть света, страна	Количество крупных городов	Количество крупных рек
Восточная Европа		
Россия	33	26
Западная Европа		
Испания	8	14
Восточная Азия		
Мьянма	14	4
Западная Азия		
Индия	26	8
	0	0

Рис. 2.20. Преобразованная таблица в формате Microsoft Access

Потом формируется новый столбец с номерами частей света, континентов с помощью «Конструктора» (рис. 2.21**):**

Лист1 : таблица

Континент, часть света, страна	Количество крупных городов	Количество крупных рек	№
Восточная Европа			1
Россия	33	26	1
Западная Европа			2
Испания	8	14	2
Восточная Азия			3
Мьянма	14	4	3
Западная Азия			4
Индия	26	8	4
	0	0	0

Рис. 2.21. Таблица, полученная после добавления нового столбца

Результат формирования новой таблицы «Части света. Континенты» и исключения записей с «частью света, континентов» из исходной таблицы приведены соответственно на рис. 2.22.

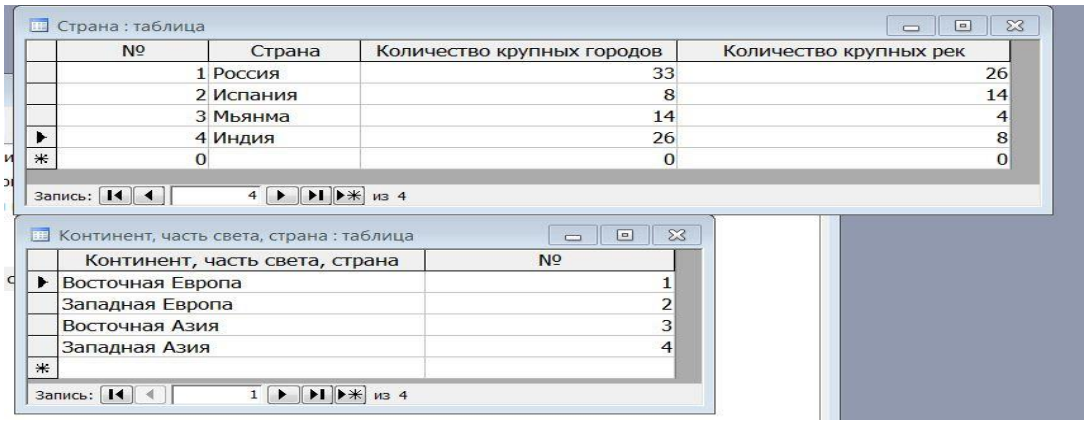


Рис. 2.22. Раздельные таблицы «Континент, часть света» и «Страна».

Для таблиц данного вида таблиц можно строить реляционные запросы, чтобы связь между собой связью типа 1: 1. В формате Microsoft Access можно создать такой запрос с помощью меню Запросы/ Конструктор/ Создать. Потом разработчик получит схему данных, представлен на рис. 2.23.

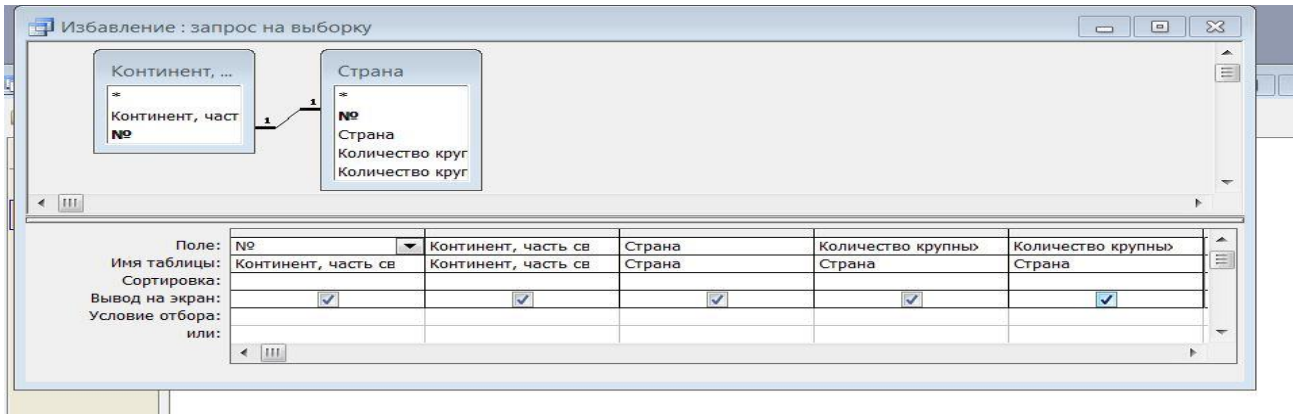


Рис. 2.23. Схема данных, полученная после создания запроса

Как видно из схемы данных, таблица «Континент, часть света, страна» и «Страна» связи между собой связью типа 1: 1 использования ключевого поля «№». Этот запрос имеет следующий вид на Microsoft Access.

SELECT [Континент, часть света, страна].№, [Континент, часть света, страна].[Континент, часть света, страна], Страна.Страна, Страна.[Количество крупных городов], Страна.[Количество крупных рек]

FROM [Континент, часть света, страна] INNER JOIN Страна ON [Континент, часть света, страна].№ = Страна.№;

Этот запрос выбирает все строки таблиц с помощью конструкции (SELECT). Если номер таблицы ([Континент, часть света, страна].№) равно номер таблицы (Страна.№), то эти таблицы связи между собой внутреннею связью с помощью конструкции (INNER JOIN).

Результат выполнения этого запроса представлен на рис. 2.24.


Избавление : запрос на выборку

№	Континент, часть света, страна	Страна	Количество крупных городов	Количество крупных рек
1	Восточная Европа	Россия	33	26
2	Западная Европа	Испания	8	14
3	Восточная Азия	Мьянма	14	4
4	Западная Азия	Индия	26	8

Рис. 2.24. Результат выполнения запроса

Как видно из рис.2.24 в результате выполнения запроса получена таблица без сложных атрибутов и подзаголовков

Суть способа преобразования заполненных нереляционных таблиц в реляционные таблицы состоит в преобразовании разработанной модели ИТВ в модель РТ в соответствии с предложенным комплектом алгоритмов и методикой их использования.

Другими словами, способ состоит в контекстном анализе значений таблиц ИТВ, выявлении их несоответствий сформулированным требованиям к РТ и исключению этих несоответствий на основе применения предложенной формализации.

Выводы по главе 2.

1. В ИТВ значения одного атрибута часто имеют различный тип, что не соответствует требованиям к реляционным таблицам.

2. Предложенный алгоритм приведения значений атрибутов заполненных таблиц к одному типу позволяет в полном объеме обеспечить соответствующее требование к реляционным таблицам.

3. В ИТВ встречаются внешние и внутренние подзаголовки, что недопустимо в РТ.

4. Предложенный алгоритм исключения подзаголовков позволяет в полном объеме обеспечить соответствующее требование к реляционным таблицам.

5. При небольших размерностях таблиц ИТВ возможен визуальный анализ данных и использование существующих инструментальных СУБД для преобразования ИТВ к РТ.

6. В ИТВ значения одного атрибута часто имеют различный тип, что не соответствует требованиям к реляционным таблицам.

7. Предложенный алгоритм приведения значений атрибутов заполненных таблиц к одному типу позволяет в полном объеме обеспечить соответствующее требование к реляционным таблицам.

8. В ИТВ встречаются внешние и внутренние подзаголовки, что недопустимо в РТ.

9. Предложенный алгоритм исключения подзаголовков позволяет в полном объеме обеспечить соответствующее требование к реляционным таблицам.

10. При небольших размерностях таблиц ИТВ возможен визуальный анализ данных и использование существующих инструментальных СУБД для преобразования ИТВ к РТ.

3. СПОСОБ НАЗНАЧЕНИЯ КЛЮЧЕВЫХ ПОЛЕЙ В ЗАПОЛНЕННЫХ ТАБЛИЦАХ

В третьей главе разработан способ назначения ключевых полей в заполненных реляционных таблицах, который по сравнению с традиционными способами позволяет снизить трудоемкость и достигнуть наилучшего решения при выполнении данной проектной процедуры.

3.1 Проблема назначения ключевых полей в заполненных таблицах

В работе [62] предложен способ назначения первичных ключей в информации табличного вида, который вполне приемлем для использования. Однако в нем учтены не все особенности ключевых полей. В частности:

– рассматривается возможность включения в первичный ключ только 3-х атрибутов;

– не полностью учитывается требование минимальности первичного ключа;

– не до конца прояснены вопросы формирования первичных ключей из нескольких атрибутов;

– затруднительно понимание предложенной формализации;

– мало освещены вопросы назначения внешних ключей

– назначение первичных ключей не рассматривается как неотъемлемая задача преобразования ИТВ в РТ.

Как сформулировано в 1-й главе основными требованиями к первичным ключам являются уникальность и минимальность. Формализуем эти требования, а затем используем их в качестве целевых функций при разработке соответствующих алгоритмов.

Уникальность. Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i=\overline{1,k}$, где $к$ – степень отношения; A_i – атрибут отношения.

$A_i = \{e_{i_1},...,e_{i_j},...,e_{i_n}\}$ $j = \overline{1,n}$, где n – мощность отношения, e_{i_j} – j - й элемент атрибута A_i.

$\mathrm{A_m} = \{e_{m_1},...,e_{m_j},...,e_{mn}\}$ $j = \overline{1,n}$, где n – мощность отношения, e_{m_j} – j - й элемент атрибута A_m

Необходимо найти такие атрибуты $A_i,\ldots,A_m$ чтобы обеспечилась истинность выражения:

$$\mathrm{concat}(e_{i_1},\ldots,e_{m_1}) \neq,...,\neq \mathrm{concat}(e_{i_j},\ldots,e_{m_j}) \neq,...,\neq \mathrm{concat}(e_{i_n},\ldots,e_{mn}) \quad (1)$$

Из выражения (1) следует, что необходимо найти такое сочетание атрибутов, чтобы конкатенация их значений была уникальна. При этом:

– проверяемый кортеж атрибутов может включать несколько атрибутов;

– число возможных сочетаний атрибутов может быть очень большим – это зависит от степени отношения (общего числа атрибутов в отношении);

– ключевой атрибут может быть только один;

– может не найтись таких атрибутов, которые обеспечивают истинность выражения (1), в этом случае назначают суррогатный ключ.

В процессе назначения первичных ключей в рамках традиционной технологии РБД исходят из визуального анализа предполагаемых схем отношений, опыта разработок, особенностей предметной области. Но такой подход не всегда приводит к успеху. Действительно, в схеме отношения могут быть погрешности, степень отношения может измеряться сотнями единиц, после заполнения таблицы могут проявиться ее непредусмотренные особенности. Но альтернативного решения пока нет.

При наличии ИТВ разработчик понимает семантику таблицы, знает степень и мощность отношения, а главное – имеет возможность анализа реальных данных. В этом случае процесс формирования первичного ключа вполне формализуем. И возможна разработка алгоритмов и соответствующего метода, которые обеспечат наилучшее решение.

Минимальность. Минимальность ключевого поля рассматривается в двух аспектах.

В первом случае во главу угла становится объем памяти, который необходим для хранения значений атрибутов, входящих в первичный ключ. Поэтому самая очевидная целевая функция – минимальное число атрибутов, входящих в первичный ключ:

$\min|A1,\ldots,Ai,\ldots,Am,\ldots,Ak|$, где $i=\overline{1,k}$, где κ – число атрибутов, входящих в первичный ключ; A_i – атрибут отношения, входящий в первичный ключ.

Строго говоря, более правильная целевая функция следующая:

$\min(\text{Length}(A_i)+\ldots+\text{Length}(A_к))$

Действительно, минимальность ключей определяется не только количеством атрибутов, входящих в них, но и суммарной длиной этих атрибутов. А длина атрибутов в основном определяется их типом и свойствами. В общем случае для выяснения средней длины значения атрибута знания его типа не всегда достаточно. Например, данные символьного типа могут быть представлены значениями меньшими допустимой длины для данного типа и даже меньшими установленного ограничения в свойствах атрибута.

При наличии только схемы отношения зачастую непросто выбрать атрибуты, которые составят первичные ключ. При недостаточном знании предметной области можно ошибиться, т.к. после заполнения таблицы реальная длина данных может быть больше или меньше предполагаемой.

Используя ИТВ, проектировщик РБД имеет возможность принимать решение на основе реальных данных. Более того, процесс поиска минимальных ключей можно формализовать.

Во втором случае под минимальностью первичного ключа подразумевается отсутствие в составе ключа атрибута, значения которого

уникальны [16]. Пусть первичный ключ К представлен множеством атрибутов:

$K = (A_1,...,A_i,...,A_j,...,A_k)$, $i = \overline{1,k}$, где k – число атрибутов, входящих в первичный ключ;

A_i – i - й атрибут отношения, входящий в первичный ключ.

$S_1 = (a_{1_1},...,a_{1_i},...,a_{1_j},...,a_{1_k})$, где S_1 – часть 1-й записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

$S_n = (a_{n_1},...,a_{n_i},...,a_{n_j},...,a_{n_k})$, где S_n – часть n-й записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

$S_m = (a_{m_1},...,a_{m_i},...,a_{m_j},...,a_{m_k})$, где S_m – часть m-й, последней записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

m – мощность отношения;

a_{n_j} – значение j -го атрибута n -й записи.

Тогда, для ключевого поля, которое включает в себя несколько атрибутов должно выполняться условие:

$$\neg((a_{1_1} \neq,...,\neq a_{n_1} \neq ... \neq a_{m_1}) \vee (a_{1i} \neq,...,\neq a_{n_i} \neq ... \neq a_{m_i}) \vee$$
$$(a_{1j} \neq,...,\neq a_{n_j} \neq ... \neq a_{m_j}) \vee (a_{1_k} \neq,...,\neq a_{nk} \neq ... \neq a_{mk}))$$

Интересно отметить, что это условие, по сути, противоположно условию уникальности ключа, если он включает в себя единственный атрибут.

На основе целевых функций, сформулированных выше, далее предлагаются неформальные, а затем формальные алгоритмы назначения первичных ключей в заполненных таблицах.

3.2 Алгоритмы назначения первичных ключей в заполненных таблицах

3.2.1 Неформальные алгоритмы назначения первичных ключей в заполненных таблицах

Следует отметить, что в качестве ключевых полей, а также в качестве атрибутов, входящих в первичный ключ, не рассматриваются атрибуты, которые имеют тип логический, MEMO, LOB, BLOB, поле объекта OLE. В связи с этим такие атрибуты необходимо исключить из рассмотрения.

П1. Поиск единственного атрибута, все значения которого уникальны.

$MKA = \varnothing$, где MKA – множество ключевых атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где $к$ – степень отношения; A_i – атрибут отношения.

$A_i = (e_{i_1} \ldots . e_{i_m})$, где e_{i_1} – 1-й элемент домена с атрибутом A_i, e_{i_m} – m-й элемент домена с атрибутом A_i.

Выполняется поиск такого атрибута A_i такого, что $e_{i_1} \neq \ldots . \neq e_{i_m}$.

В связи с этим перебираются все атрибуты отношения.

Если такой атрибут находится, то он запоминается: $MKA = MKA + A_i$

Если после перебора всех атрибутов $MKA = \varnothing$, то это означает, что назначить первичный ключ, включающий в себя один атрибут невозможно.

В этой связи разработчик должен принять одно из решений: назначить суррогатный ключ или сформировать первичный ключ на базе нескольких атрибутов. Если принято 2-е решение, то переход к П.2

Если $MKA \neq \varnothing$, то атрибуты с уникальными значениями нашлись и необходимо выбрать из них атрибут, удовлетворяющий требованиям минимальности. Так как на роль ключевого атрибута претендует только один из найденных атрибутов, то необходима проверка только первой части требования минимальности длины атрибута. В связи с этим необходимо найти в множестве MKA атрибут с минимальной длиной т.е. $\min(\text{Length}(A_i), \ldots, \text{Length}(A_к))$, где $(A_i, \ldots, A_к) \in MKA$ Таким образом, найден первичный ключ. STOP.

Несмотря на то, что найден атрибут минимальной длины, разработчику должен быть предоставлен весь список возможных ключевых полей, чтобы он имел возможность выбора ключевого атрибута, пусть и не минимального. Это может оказаться необходимо в связи с особенностями предметной области.

П2. Поиск атрибутов, конкатенация значений, которых минимальна.

Необходимо проанализировать все возможные сочетания атрибутов. Каждое сочетание проверить на уникальность конкатенации их значений.

Все сочетания с уникальными значениями необходимо сохранить, а затем для каждого сочетания измерить их суммарную длину. Для обеспечения возможности принятия волевого решения необходимо расположить эти сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности, и придется выбирать альтернативный ключ.

Имеет смысл сначала выполнить проверку соответствия требований к первичному ключу 2-х атрибутов, т.е. необходимо проанализировать C_N^2 сочетаний, где N – степень отношения.

$MKA^2 = \varnothing$, где MKA^2 – множество пар атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где κ – степень отношения; A_i – атрибут отношения.

$MPA = \varnothing$

Ищутся все возможные сочетания пар атрибутов и запоминаются в массив MPA:

```
Count=0
For i = 1 to k-1
    For j = i+1 to k
        Count = Count + 1
        S = Concat (Ai , Aj)
```

MPA (Count) = MPA + S

Next i

Next j

Таким образом, в массиве MPA сформируются все возможные пары атрибутов, а в счетчике *Count* хранится их количество.

Проверяются все пары на уникальность.

MUP = ∅ / Массив пар атрибутов, представляющих собой атрибуты, все соответствующие пары значений которых уникальны */*

Count1 = 0

For i = 1 to Count

S = MPA(Count)

/ По сути S представляет собой пару атрибутов* (A_i, A_j) $A_i = (e_{i_1} \ldots . e_{i_m})$, *где* e_{i_1} – *1-й элемент домена с атрибутом* A_i, e_{i_m} – *m-й элемент домена с атрибутом* A_i.

$A_j = (e_{j_1} \ldots . e_{j_m})$, *где* e_{j_1} – *1-й элемент домена с атрибутом* A_j, e_{j_m} – *m-й элемент домена с атрибутом* A_j, m – *мощность отношения. */*

For n = 1 to m

Для каждой пары атрибутов (A_i, A_j) *выполняется проверка условия* $Concat(e_{i_1}, e_{j_1}) \neq \ldots . \neq Concat((e_{j_1}, e_{j_m})$

Next n

Если текущая пара атрибутов имеет все соответствующие пары значений уникальными, то эта пара добавляется к массиву пар с уникальными значениями:

Count1 = Count1 + 1

MUP(Count1) = S

Next i

Если претенденты на ключевой атрибут найдены, т.е. $MUP \neq \varnothing$, то для проверки второго требования минимальности выполняется переход к П3. Для обеспечения возможности принятия волевого решения необходимо

расположить все найденные сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности и придется выбирать альтернативный ключ.

Если не найдено таких двух атрибутов, которые удовлетворяют требованиям к первичному ключу, то разработчик может назначить суррогатный ключ или попытаться найти такие 3-и атрибута, которые удовлетворяют требованиям к первичному ключу, т.е. проанализировать C_N^3 сочетаний.

П3. Поиск первичного ключа на основе 3-х атрибутов

$MKA^3 = \varnothing$, где MKA^3 – множество троек атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где $к$ – степень отношения; A_i – атрибут отношения.

$MPA = \varnothing$

Ищутся все возможные сочетания троек атрибутов и запоминаются в массив MPA:

```
Count=0
For i = 1 to k-2
   For j = i+1 to k
      For r = j+1 to k
         Count = Count + 1
         S = Concat (Ai , Aj, , Ar)
         MPA (Count) = MPA + S
      Next r
   Next i
Next j
```

Таким образом, в массиве MPA сформируются все возможные тройки атрибутов, а счетчике *Count* хранится их количество.

Проверяются все тройки на уникальность.

MUP = ∅ / Массив троек атрибутов, представляющих собой атрибуты, у которых соответствующие тройки значений которых уникальны */*

Count1 = 0

For i = 1 to Count

S = MPA(Count)

/ По сути S представляет собой тройку атрибутов (A_i, A_j, , A_r)*

$A_i = (e_{i_1} ....e_{i_m})$, где e_{i_1} – 1-й элемент домена с атрибутом A_i, e_{i_m} – m-й элемент домена с атрибутом A_i.

$A_j = (e_{j_1} ....e_{j_m})$, где e_{j_1} – 1-й элемент домена с атрибутом A_j, e_{r_m} – m-й элемент домена с атрибутом A_j, m – мощность отношения.

*A_r = ($e_{r_1} ...e_{r_m}$), где e_{r_1} – 1-й элемент домена с атрибутом A_r, e_{j_m} – m-й элемент домена с атрибутом A_r, m – мощность отношения. */*

For n = 1 to m

Для каждой тройки атрибутов (A_i, A_j, A_r) выполняется проверка условия Concat($e_{i_1}, e_{r_1}, e_{j_1}$) ≠ .... ≠ Concat(($e_{j_1}$,.$e_{r_m}$, e_{j_m}))

Next n

Если текущая тройка атрибутов имеет все соответствующие тройки значений уникальными, то эта тройка добавляется к массиву троек с уникальными значениями:

Count1 = Count1 + 1

MUP(Count1) = S

Next i

Если претенденты на ключевой атрибут найдены, т.е., *MUP* ≠ ∅,то для проверки второго требования минимальности выполняется переход к П3.

Для обеспечения возможности принятия волевого решения необходимо расположить все найденные сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности и придется выбирать альтернативный ключ.

Аналогичный подход распространяется и на 4 и на 5 атрибутов, но как показывает практика, в ключевом поле очень редко задействуют более 4-х атрибутов.

Может сложиться впечатление, что этот процесс займет немало машинного времени, которое напрямую зависит от степени отношения. Ведь число возможных комбинаций может оказаться очень большим. Но это не совсем так.

Из комбинаторики известно, что число возможных сочетаний C_n^k = n!/(n-k)!/k!, где n – число элементов множества, а к – количество проверяемых значений число возможных сочетаний C_n^k = n!/(n-k)!/k!, где n – число элементов множества, а к – количество проверяемых значений [63, 64]. Например, для множества из 4-х элементов (a, b, c, d) возможны следующие сочетания пар элементов (a, b), (a, c), (a, d), (b, c), (b, d) , (c, d).

Подсчитаны числа возможных сочетаний для различных n и k, которые представляют наибольший интерес. Результаты сведены в таблицу 3.1.

Таблица 3.1

C_{10}^2	C_{10}^3	C_{10}^4	C_{20}^2	C_{20}^3	C_{20}^4	C_{30}^2	C_{30}^3	C_{30}^4
45	120	210	190	1140	4845	435	4060	27405

Степени отношений (10, 20, 30) наиболее близки к распространенным степеням в реальных БД, а число атрибутов (2, 3, 4) обычно достаточно для первичного ключа. Подсчитанное число сочетаний вполне может быть обработано на современных компьютерах без существенной потери машинного времени.

П3. Поиск первичных ключей, у которых нет атрибутов, входящих в первичный ключ, значения которого уникальны.

После выполнения П2 настоящего алгоритма будет получен массив пар или массив троек атрибутов, представляющих собой атрибуты, у которых соответствующие тройки значений уникальны MUP $\neq\varnothing$. Необходимо проверить все элементы массивов на предмет наличия атрибутов, которые входят в элементы массива и которые имеют уникальные значения. Если такие элементы (атрибуты) найдутся, то они не могут в соответствии со вторым правилом минимальности первичного ключа претендовать на составную часть первичного ключа.

Проверка массива пар атрибутов.

В этом случае $MUP = (P_1, \ldots, P_i, \ldots . P_n)$, где P_i – i-я пара атрибутов, n – число пар атрибутов с уникальными конкатенациями значений.

$P_i = (A_{i1}, A_{i2})$

$A_{i1} = (e_{i1_1}, \ldots, e_{i1_m})$, *где* e_{i1_1} – 1-й элемент домена с атрибутом A_{i1}, m – мощность отношения.

$A_{i2} = (e_{i2_1}, \ldots, e_{i2_m})$, *где* e_{i2_1} – 1-й элемент домена с атрибутом A_{i2}, m – мощность отношения.

При этом должно выполняться условие:

$\neg ((e_{i1_1} \neq, \ldots, \neq e_{i1_m}) \vee (e_{i2_1} \neq, \ldots, \neq e_{i2_m}))$

Подобные условия должны выполняться для всех элементов массива $MUP = (P_1, \ldots, P_i, \ldots . P_n)$, то есть для всех пар атрибутов, претендующих на ключевые.

Проверка массива троек атрибутов.

В этом случае $MUP = (P_1, \ldots, P_i, \ldots . P_n)$, где P_i – i-я тройка атрибутов, n – число пар атрибутов с уникальными конкатенациями значений.

$P_i = (A_{i1}, A_{i2}, A_{i3})$

$A_{i1} = (e_{i1_1}, \ldots, e_{i1_m})$, *где* e_{i1_1} – 1-й элемент домена с атрибутом A_{i1}, m – мощность отношения.

$A_{i2} = (e_{i2_1},...,e_{i2_m})$, *где* e_{i2_1} – 1-й элемент домена с атрибутом A_{i2}, m – мощность отношения.

$A_{i3} = (e_{i3_1},...,e_{i3_m})$, *где* e_{i3_1} – 1-й элемент домена с атрибутом A_{i3}, m – мощность отношения.

При этом должно выполняться условие:

$$\neg ((e_{i1_1} \neq,...,\neq e_{i1_m}) \vee (e_{i2_1} \neq,...,\neq e_{i2_m}) \vee e_{i3_1} \neq,...,\neq e_{i3_m}))$$

Подобные условия должны выполняться для всех элементов массива $MUP = (P_1, ...,P_i,....P_n)$, то есть для всех троек атрибутов, претендующих на ключевые.

Для четверок атрибутов, претендующих на роль первичного ключа, выполняются аналогичные пункты алгоритма. STOP.

В конечном итоге будет получен список пар или троек атрибутов, которые претендуют на первичный ключ. Из этого списка разработчик должен выбрать единственный кортеж, исходя из своих прикладных соображений. Список может состоять и из одного элемента. Если список окажется пустым, то назначается суррогатный ключ. Однако такая ситуация скорее всего будет отсечена на шаге поиска первичного ключа, включающего в себя один атрибут.

3.2.2. Формальные алгоритмы назначения первичных ключей в заполненных таблицах

П1. Поиск единственного атрибута, все значения которого уникальны.

```
MKA = ∅ _ _ Массив ключевых атрибутов
Count = 0 _ _ Счетчик ключевых атрибутов
For i = to k _ _ Перебор всех атрибутов отношения
    F = 1 _ _ Флажок - все элементы текущего атрибута уникальны
    For j = 1 to m-1 _ _ Перебор всех элементов атрибута
        For r = j + 1 to m _ _ Перебор всех элементов атрибута
            If e_ij = e_ir Then _ _ Сравнение каждой пары элементов
            Begin _ _ Если они равны
```

```
                F = 0 _ _ Флажок сбрасывается
                Exit r _ _ Завершается цикл по r
                Exit j _ _ Завершается цикл по j
            End
        Next r
    Next j
    If F = 1 Then _ _ Если флажок не сброшен
    Begin
        Count = Count + 1 _ _ В массив добавляется очередной
        MKA(Count) = A_i _ _ претендент на ключевой атрибут
    End
Next i
IF MKA = ∅ Then
Begin
    Print ("Претендентов на ключевой атрибут нет")
    Print ("Сформируйте суррогатный ключ или")
    Print ("перейдите к поиску ключа из нескольких атрибутов")
Else
Begin
    Print ("Претенденты на ключевые атрибуты:  ", MKA)
End
/* Здесь к- число атрибутов отношения; m – мощность отношения. * /
/* Поиск атрибута с минимальной длиной */

/* Упорядочивание по возрастанию длины элементов массива МКА */
/* Используется известный алгоритм сортировки */
B = 0
For i = 1 to Count-1
    For j = i + 1 Count
        If Length( MKA_j) < If Length( MKA_i) Then
```

Begin

$P = MKA_j$

$MKA_i = MKA_j$

$MKA_j = P$

End

Next j

Next i

Print ("Претенденты на ключевые атрибуты в порядке возрастания длины : ")

Print (MKA)

STOP

П2. Поиск атрибутов, конкатенация значений, которых минимальна.

/ Ищутся все возможные сочетания пар атрибутов */*

MPA = ∅ *_ _ массив пар атрибутов*

MUP = ∅ / Массив пар атрибутов, представляющих собой атрибуты, все соответствующие пары значений которых уникальны */*

Count = 0 _ _ счетчик пар атрибутов

For i = 1 to k-1 _ _ цикл по числу атрибутов

For j = i+1 to k

Count = Count + 1

$S = Concat (A_i , A_j)$

MPA (Count) = MPA + S

Next i

Next j

Count1 = 0 _ _ счетчик пар атрибутов с уникальными значениями

For i = to Count _ _Перебор всех возможных пар атрибутов

F = 1 _ _ Флажок – все элементы текущей пары

_ _ атрибутов уникальные

For j = 1 to m-1 _ _ Перебор всех пар элементов атрибутов

```
        For r = j + 1 to m _ _ Перебор всех пар элементов атрибутов
            If e''_j(MPA_i) = e''_r(MPA_i) Then
            /* Здесь e''_j(MPA_i) j-я пара элементов, соответствующая i-
            му элементу массива пар атрибутов MPA */
            Begin _ _ Если они равны
                F = 0 _ _ Флажок сбрасывается
                Exit r _ _ Завершается цикл по r
                Exit j _ _ Завершается цикл по j
            End
        Next r
    Next j
    If F = 1 Then _ _ Если флажок не сброшен
    Begin
        Count1 = Count1 + 1 _ _ В массив добавляется очередной
        MUP(Count1) = MPA_i _ _ претендент на пару атрибутов,
                                _ _ входящий в ключевой атрибут
    End
Next i
IF MUP = ∅ Then
Begin
    Print ("Претендентов на пару ключевых атрибутов нет")
    Print ("Сформируйте суррогатный ключ или")
    Print ("перейдите к поиску ключа из трех атрибутов")
Else
Begin
    Print ("Претенденты на ключевые атрибуты: ", MUP)
End
/* m – мощность отношения. * /
```

/ Поиск пары атрибута с минимальной длиной */*

/ Упорядочивание по возрастанию длины элементов массива MUP */*

B = 0

For i = 1 to Count1-1

For j = i + 1 Count1

If Length(MUP_j) < If Length(MUP_i) Then

Begin

P = MUP_i

MUP_i = MUP_j

MUP_j= P

End

Next j

Next i

Print (“Претенденты на пары ключевых атрибутов в порядке возрастания длины: ”)

Print (MUP)

STOP

П3. Поиск первичного ключа на основе 3-х атрибутов

/* Ищутся все возможные сочетания троек атрибутов */

MUT = ∅ / Массив троек атрибутов, представляющих собой атрибуты, все соответствующие тройки значений которых уникальны */*

Count=0_ _ счетчик троек атрибутов

For i = 1 to k-2_ _ цикл по числу атрибутов - 2

For j = i+1 to k

For r = j+1 to k

Count = Count + 1

S = Concat (A_i , $A_{j,}$, A_r)

MUT (Count) = MUT + S

Next r

Next i

Next j

/ к – общее число атрибутов */*

Count1 = 0 _ _ счетчик троек атрибутов с уникальными значениями

For i = to Count _ _Перебор всех возможных троек атрибутов

F = 1 _ _ Флажок - все элементы текущей тройки

_ _ атрибутов уникальные

For j = 1 to m-1 _ _ Перебор всех троек элементов атрибутов

For r = j + 1 to m _ _ Перебор всех троек элементов атрибутов

If $e_j'''(MUT_i) = e_r'''(MUT_i)$ *Then*

/ Здесь* $e_j'''(MUT_i)$ *j-я тройка элементов, соответствующая i-му элементу массива троек атрибутов MUT */*

Begin _ _ Если они равны

F = 0 _ _ Флажок сбрасывается

Exit r _ _ Завершается цикл по r

Exit j _ _ Завершается цикл по j

End

Next r

Next j

If F = 1 Then _ _ Если флажок не сброшен

Begin

Count1 = Count1 + 1 _ _ В массив добавляется очередной

MUT(Count1) = MPA_i *_ _ претендент на тройку атрибутов,*

_ _ входящий в ключевой атрибут

End

Next i

IF $MUT = \varnothing$ *Then*

Begin

Print (“Претендентов на пару ключевых атрибутов нет”)

Print (“Сформируйте суррогатный ключ или”)

Print (“перейдите к поиску ключа из трех атрибутов”)

Else

Begin

Print (“Претенденты на ключевые атрибуты: ”, MKA)

End

/ m – мощность отношения. * /*

/ Поиск тройки атрибутов с минимальной длиной */*

/ Упорядочивание по возрастанию длины элементов массива MUT */*

B = 0

For i = 1 to Count1-1

For j = i + 1 Count1

If Length(MUT_j) < If Length(MUT_i) Then

Begin

P = MUT_i

MUT_i = MUT_j

MUT_j= P

End

Next j

Next i

Print (“Претенденты на тройки ключевых атрибутов в порядке возрастания длины : ”)

Print (MUT)

STOP

Как нетрудно заметить, алгоритм поиска первичного ключа на основе 2-х атрибутов и алгоритм поиска первичного ключа на основе 3-х атрибутов имеют схожую структуру. В связи с этим алгоритмы, разработанные для большего числа атрибутов, в работе не приводятся.

3.3. Алгоритмы назначения внешних ключей в заполненных таблицах

3.3.1. Неформальные алгоритмы назначения внешних ключей в заполненных таблицах

В соответствии с [11] определение внешнего ключа звучит следующим образом. Пусть имеется отношение R2, значения атрибутов которого равны значениям атрибутов первичного ключа отношения R2. Тогда такие атрибуты являются внешним ключом по отношению к R1.

В соответствии с определением для нахождения внешних ключей необходимо проанализировать каждую таблицу, все возможные сочетания пар таблиц БД. Для каждой пары таблиц выполнить поиск неключевых атрибутов одной таблицы, значения которых были бы равны значениям ключевых атрибутов другой таблицы. Если таковые найдутся, то они и будут претендовать на внешний ключ.

Следует обратить внимание на то, что эти ключи ориентированы на формирование связи между таблицами типа “многие к одному”. Они и представляют наибольший интерес для БД. Это связано с тем, что связь типа “один к одному” по сути представляет собой связь между частями одной таблицы, а связь типа “многие ко многим” базируется на двух связях типа “многие к одному” между рассматриваемыми и какой-либо третьей таблицей.

Неформальный алгоритм.

П1. Осуществляется поиск всех возможных сочетаний пар таблиц анализируемой БД.

Пусть имеется табличное пространство T:

$T=(T_1, \ldots, T_i, \ldots, T_m, \ldots, T_k)$, $i=\overline{1,k}$, где $к$ – число прикладных таблиц БД.

$MPT = \varnothing$

Ищутся все возможные сочетания пар таблиц и запоминаются в массив *MPT*

Count=0

For i = 1 to k-1

For j = i+1 to k

Count = Count + 1

S = Concat (T_i , T_J)

MPT (Count) = S

Next i

Next j

Таким образом, в массиве MPT сформируются все возможные пары атрибутов, а счетчике *Count* хранится их количество.

П2. Для каждой пары таблиц из массива *MPT* выполняется поиск внешнего ключа.

Организуется цикл на Count итераций, в котором перебираются все пары таблиц.

Для каждой пары таблиц выполняются проверки. Причем рассматриваются два случая: когда в качестве таблицы с первичным ключом рассматривается 1-я таблица; когда в качестве таблицы с первичным ключом рассматривается 2-я таблица.

П2.1. Для ключа состоящего из одного атрибута.

$A_i = (e_{i_1} ....e_{i_m})$, где A_i – атрибут, соответствующий первичному ключу 1-й таблицы, e_{i_1} – 1-й элемент домена с атрибутом A_i, – e_{i_m} m-й элемент домена с атрибутом A_i, m – мощность 1- й таблицы.

$A_j^2 = (e_{j_1}^2 ...e_{j_f}^2)$, где A_j^2 – атрибут, соответствующий неключевому атрибуту 2-й таблицы, $e_{j_1}^2$ – 1-й элемент домена с атрибутом A_j^2, $e_{j_f}^2$ – f-й элемент домена с атрибутом A_j^2, f – мощность 2-й таблицы. Важно помнить, что число столбцов у 2-й таблицы, как правило, больше одного.

Следует отметить, что m и f в общем случае неравны. Ведь таблицы, как правило, имеют различные мощности.

Организуется цикл на g итераций, где g число столбцов 2-й таблицы

Сравниваются значения первичного ключа 1-й таблицы и значения текущего столбца 2-й таблицы:

Count1 = 0

For t = 1 to m

For p = 1 to f

If $e_{i_t} = e^2_{j_p}$ *THEN Count1 = Count1 + 1*

End p

End t

If Count1 ≥ Min(m, f) Then PRINT (“столбец ”, g, “2-й таблицы претендует на внешний ключ”

Завершается цикл по g

Если н для одного столбца таблицы 2 не сформируется сообщение, то во 2-й таблице внешних ключей нет.

Здесь m – мощность 1-й таблицы,

f – мощность 2-й таблицы.

В качестве пояснения можно сказать, что истинность конструкции Count ≥ Min(m, f) в подавляющем случае подтверждает, что найден внешний ключ. Действительно, для каждого значения одной таблицы нашлось равное ему значение другой таблицы.

П2.2. Для ключа, состоящего из двух атрибутов.

Принципиальное отличие от предыдущего алгоритма состоит в том, что во второй таблице рассматриваются все возможные сочетания пар атрибутов 2-й таблицы.

В связи с этим необходимо предварительно сформировать массив всех возможных пар атрибутов 2-й таблицы.

MPA = ∅

Count1 = 0

For i = 1 to f-1

For j = 1 to f

S = Concat(A_i, A_j)

Count1 = Count1 + 1

MPA(Count1) = s

Next j

Next i

Здесь MPA – массив пар атрибутов 2-й таблицы,

f – степень 2-й таблицы,

A_i, A_j – два атрибута 2- й таблицы.

Организуется цикл на Count1 итераций, где Count1 число сочетаний пар атрибутов 2- й таблицы, которые содержатся в массиве MPA***, z – параметр цикла***

Сравниваются все пары значений, соответствующих первичному ключу 1-й таблицы и все пары значений для каждого сочетания атрибутов 2-й таблицы:

Count1 = 0

For t = 1 to m

For p = 1 to f

If Concat(e_{i_t}, e_{k_t}) = e_p (MPA_z) THEN Count1 = Count1 + 1

End p

End t

If Count ≥ Max(m, f) Then PRINT (“столбцы”, MPA_z, “2-й таблицы претендует на внешний ключ”

Завершается цикл по Count*1*

Если ни для одного сочетания таблицы 2 не сформируется сообщение, то во 2-й таблице внешних ключей нет.

Здесь

$A_i, A_k = ((e_{i_1}, e_{k_1}) \ldots (e_{i_m}, e_{k_m}))$, где A_i – 1-й атрибут, соответствующий первичному ключу 1-й таблицы, A_k – к-й атрибут, соответствующий первичному ключу 1-й таблицы, e_{i_1} – 1-й элемент домена с атрибутом A_i, e_{k_1} – 1-й элемент домена с атрибутом A_k.

m – мощность 1-й таблицы,

f – мощность 2-й таблицы,

e_{i_t}, e_{k_t} – текущие t-е 2-а элемента первичного ключа 1-й таблицы,

e_p (MPAz) - текущие p-е 2-а элемента 2-й таблицы, при этом номера сочетаний определяются z-м элементом массива MPA (массива, в котором содержаться все пары атрибутов 2-й таблицы).

П2.3. и П2.4. Для ключа состоящего из трех атрибутов и для ключа состоящего из четырех атрибутов выполняются в основном на базе конструкций **П2.1. и П2.2.** Но из соображений их значительного объема не приводятся.

Завершается цикл на Count итераций, в котором перебираются все пары таблиц

3.3.2 Формальные алгоритмы назначения внешних ключей в заполненных таблицах

П1. Осуществляется поиск всех возможных сочетаний пар таблиц анализируемой БД.

/* $T=(T_1, \ldots, T_i, \ldots, T_m, \ldots, T_k)$, $i = \overline{1,k}$, где *к* – число прикладных таблиц БД */

MPT = ∅ _ _ Массив всех пар таблиц БД

/* Ищутся все возможные сочетания пар таблиц */

Count=0

For i = 1 to k-1 _ _ к – число таблиц БД

For j = i+1 to k

Count = Count + 1_ _ счетчик числа пар таблиц

S = Concat (T_i , T_J)

MPT (Count) = S

Next i

Next j

П2. Для каждой пары таблиц из массива *MPT* выполняется поиск внешнего ключа.

/* Перебираются все возможные пары таблиц, сформированные ранее, перебираются все столбцы 1-й таблицы текущей пары, перебираются все столбцы 2-й таблицы текущей пары, каждый элемент первого текущего столбца сравнивается с каждым элементом второго текущего столбца, если элементы равны, то они подсчитываются. Если совпавших элементов оказалось больше, чем наибольшая мощность пары таблиц, то столбцы претендуют на ключевые */

F = 0 _ _ флажок наличия связанных ключей

Count1 = 0 _ _ счетчик совпадений

For i = 1 to Count _ _ – число пар таблиц БД

For l = 1 to C1_ _ – степень 1-й таблицы из i-й пары

For m= 1 to C2 _ _ – степень 2-й таблицы из i-й пары

For j= 1 to M1_ _ – мощность 1-й таблицы из i-й пары

For k= 1 to M2_ _ – мощность 2-й таблицы из i-й пары

If $e_{l_j} = e_{m_k}$ *Then Coun1 = Coun1 + 1*

Next k

Next j

If Coun1 >= Max(M1, M2) Then

Begin

Print (“Для пары таблиц ”, i, “столбцы”, l, “ и ”, m)

Print (“ претендуют на первичный и внешние ключи”)

Count1 = 0

```
                F = 1
            End
        Next m
    Next l
Next i
If F = 0 Then  Print (“ внешних ключей не обнаружено не для одной из таблиц”)
```

Следует обратить внимание на то, что атрибуты, предложенные алгоритмом в качестве связанных атрибутов, могут не удовлетворять разработчика, поэтому необходимо предоставить ему возможность окончательного решения.

Суть способа назначения ключевых полей в ИТВ состоит в преобразовании модели ИТВ в модель РТ в соответствии с требованиями к ключевым полям, со схемой взаимодействия разработчика и подсистемы и предложенным комплектом алгоритмов.

Другими словами, способ состоит в контекстном анализе значений таблиц ИТВ, выявлении несоответствий требованиям к ключевым полям и исключению этих несоответствий на основе применения предложенной формализации.

Выводы по главе 3

1. Традиционный подход к назначению ключевых полей в РТ – это лучшее, что сегодня имеется для компактных, непротиворечивых РБД.
2. При наличии ИТВ, возможна формализация назначения ключевых полей на основе анализа имеющейся информации.
3. Предложенный способ назначения первичных ключей на основе анализа ИТВ органично сочетается с традиционной методологией проектирования РБД.

4. Предложенный способ назначения внешних ключей на основе анализа ИТВ органично сочетается с традиционной методологией проектирования РБД.

5. ИТВ не удовлетворяют требованиям к ключевым полям РТ.

6. Разработчики РБД заинтересованы в наличии автоматизированных средств назначения первичных и внешних ключей на основе формализованного анализа ИТВ.

7. Для преобразования ИТВ в РТ необходимо решить следующие задачи:

– назначение первичных ключей на базе одного атрибута;

– назначение первичных ключей на базе нескольких атрибутов;

– минимизация первичных ключей;

– назначение внешних ключей.

8. Современные способы назначения ключевых полей могут быть использованы как основа для формализации алгоритмов формирования ключевых полей в ИТВ.

4. МЕТОДИКА ФОРМИРОВАНИЯ РЕЛЯЦИОННЫХ ТАБЛИЦ НА ОСНОВЕ ИНФОРМАЦИИ ТАБЛИЧНОГО ВИДА

В четвертой главе решена задача формализации методики формирования реляционных таблиц на основе информации табличного вида.

Разработана операторная модель методики. Разработана сетевая модель методики. Выполнено исследование методики на основе использования сетевой модели. На основе анализа сетевой модели методики проектирования скорректирована операторная модель.

4.1. Постановка задачи формализации методики

Целью работы является разработка методики формирования реляционных таблиц на основе информации табличного вида. Другими словами, необходимо разработать описание совокупности приемов, способов для достижения поставленной цели [92]. Понятие способ в энциклопедиях и словарях определяется зачастую через понятие метод и наоборот. В любом случае в развернутых определениях этих понятий обычно используют такие термины как модели, преобразования, совокупность приемов, задачи, система действий, достижение цели. Все это отражено в предыдущих главах. Но из соображений корректности мы используем и впредь будем использовать понятие способ.

Состав подзадач, решаемых в рамках основной задачи, и соответственно состав необходимых способов определен в 1-й главе. Способ решения задач формирования реляционных таблиц на основе информации табличного вида разработан во 2-й главе. Способ назначения ключевых полей разработан в 3-й главе. Эти способы тесно связаны между собой и ориентированы на достижение одной цели. Каждый из разработанных способов включает в себя совокупность алгоритмов, которые взаимосвязаны

и взаимозависимы и выполняются в логической последовательности. А взаимосвязанные способы и алгоритмы, выполняемые в логической последовательности, составляют методику.

Методика в общем и конкретном случаях − это не только совокупность способов выполнения действий. Большинству из способов соответствуют частные приемы использования их компонент.

Так, например, способ преобразования нереляционных таблиц ИТВ в реляционные таблицы включает в себя приемы исключения заголовков внутри таблиц, исключения сложных подзаголовков, исключения пустых подзаголовков, избавления от вертикальных подзаголовков, приведения значений атрибутов к одному типу, исключения дублирования записей, исключения пустых записей.

Способ назначения ключевых полей включает в себя приемы выявления домена с уникальными значениями его элементов, выявления сочетания доменов с уникальными сочетаниями соответствующих им элементов, поиска минимальных первичных ключей, включающих в себя один атрибут, поиска минимальных первичных ключей, включающих в себя несколько атрибутов, выявления атрибутов, которые входят в первичный ключ и содержат уникальные значения, выявления внешних ключей.

В связи с этим необходимо выполнить следующие мероприятия по разработке и использованию модели методики формирования РТ на основе использования ИТВ:

- определить компоненты методики (способы, приемы, алгоритмы, специализированные средства);
- сформировать взаимосвязи между этими компонентами;
- выбрать математический аппарат, позволяющий описать методику и исследовать ее характеристики;
- выполнить построение адекватной модели методики на основе использования выбранного математического аппарата;

- выявить концептуальные ошибки в методике;
- выполнить исследование динамических свойств методики на основе использования ее модели;
- использовать разработанную модель в качестве основы для реализации алгоритмов и специализированных средств.

Методика преобразования ИТВ в РТ – это процесс, в котором активное участие принимает разработчик. Вопросам человеко-машинных процессов решения проектных задач посвящен ряд работ [6, 74, 75], которые активно использованы при разработке модели методики.

Реализована следующая процедура по построению модели методологии проектирования РБД на основе существующей ИТВ [6].

На первом этапе по аналогии с описанием процесса взаимодействия решающих систем [75], используя отличительные особенности ИТВ и РТ, в операторной форме описываются отдельные шаги преобразования ИТВ в РТ, формируются связи между ними, определяются правила и порядок их использования. Такое описание разработано с целью выявления основных компонент разрабатываемой человеко-машинной системы, выявления основных связей между ними. В дальнейшем оно будет использовано для разработки формальных моделей интерактивных процессов проектирования РБД на основе ИТВ.

На втором этапе операторная модель используется в качестве исходной формализации для разработки модели методики, построенной на основе аппарата сетей Петри, и формируется соответствующая сеть.

На третьем этапе с помощью аппарата сетей Петри выявляются и исключаются дефекты модели, а, следовательно, исключаются дефекты объекта моделирования. В конечном итоге строится сетевая модель методики, свободная от концептуальных ошибок, а значит адекватная объекту моделирования.

4.2. Операторная модель методики преобразования ИТВ в РТ

В описанных мероприятиях за основу взят подход, который предложен в работах руководителя диссертации Брешенкова А.В. Он изложен в работах [6, 7, 32, 37, 53, 71, 76]

На начальном уровне абстрагирования от большинства компонент человеко-машинной системы схема процесса преобразования ИТВ в РТ может быть проиллюстрирована рисунком 4.1. В связи с тем, что БД всегда проектируются с активным участием разработчика и с применением соответствующих автоматизированных средств, то при выполнении мероприятий по преобразованию ИТВ в РБД разработчик необходим [37].

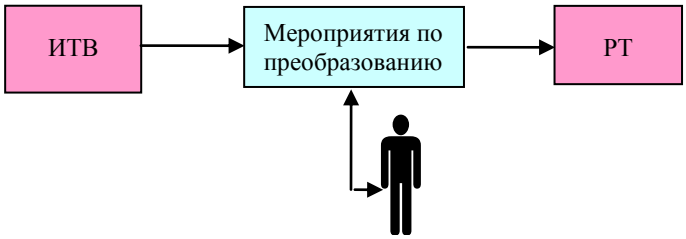


Рис. 4.1. Исходная схема процесса преобразования ИТВ в РБД с участием разработчика

Такая схема предполагает необходимость человеко-машинных методов и средств, т.е. средств автоматизированного преобразования ИТВ в РТ.

Участие разработчика в мероприятиях по преобразованию ИТВ в РТ, с одной стороны, позволяет задействовать его творческий потенциал, а, с другой стороны, ставит результаты преобразования в зависимость от человеческого фактора [76]. В связи с этим необходимо разработать средства, которые позволят:

- активно участвовать разработчику в процессе преобразования;
- с наибольшей эффективностью использовать известные инструментальные средства проектирования РБД.

Введем оператор преобразования ИТВ в РБД – *ОП*. Результатом применения этого оператора *ОП* является модель РБД, отвечающая

требованиям непротиворечивости, минимальности, целостности. К числу операндов оператора *ОП* в первую очередь относится модель ИТВ. Схема преобразования модели ИТВ в модель РБД или операторная модель преобразования примет вид рис. 4.2.

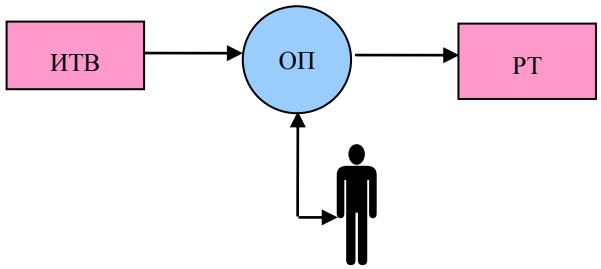


Рис. 4.2. Операторная модель процесса преобразования ИТВ в РБД

В операторной форме преобразование ИТВ в РТ выглядит следующим образом: *РТ= ОП (ИТВ)*.

В качестве результата применения *ОП* формируется не только модель РТ, но и совокупность данных, позволяющих разработчику принимать решения в соответствии со своей моделью поведения.

Таким образом, для принятия решения разработчиком о применении средств оператора *ОП* необходима система оценок *(СО)* качества преобразования ИТВ, а для реализации императив (указаний) необходим набор действий разработчика, который реализует оператор *ОПР*.

С учетом сказанного рис. 4.2 примет вид рис. 4.3

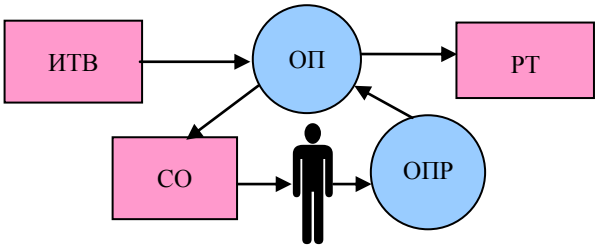


Рис. 4.3. Вторая операторная форма модели преобразования ИТВ в РБД

Коль скоро разработчик на основе *СО* принимает и выполняет решение *ОПР* по поводу использования *ОП*, то логично предположить, что и сам оператор итерационно выполняется на основе анализа системы оценок.

Учитывая это и, исключив из схемы изображение “человека” (теперь он представлен *СО* и *ОПР*), получим операторную модель преобразования ИТВ вида (рис. 4.4).

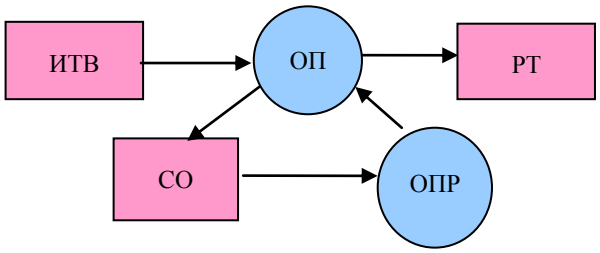


Рис. 4.4. Третья операторная форма модели преобразования ИТВ в РБД

Уже на данном шаге представления операторной модели преобразования можно сформулировать основные задачи исследования и разработки. Необходимо выполнить следующие мероприятия [71]:

- оценить полноту модели ИТВ относительно достаточности данных для выполнения *ОП* и *ОПР*;
- оценить полноту модели РТ относительно достаточности данных для выполнения *ОП* и *ОПР*;
- разработать методы и средства выполнения *ОП* и *ОПР*;
- разработать *СО*.

При этом названные мероприятия необходимо выполнять в комплексе, так как *СО, ОП, ОПР* напрямую зависят от ИТВ и РТ и наоборот.

Выполним разработку *СО, ОП и ОПР*, исходя из схемы иерархии разрабатываемых алгоритмов, предложенной в главе 1 (рис. 1.8).

Модели РТ и ИТВ разработаны во 2-й главе. Как видно из этих моделей, у них одна и та же предметная область. Это может помочь при выполнении *ОП* в том случае, если одни и те же объекты (таблицы, поля) в рамках модели ИТВ называются одинаково, тем более это поможет при выполнении *ОПР*.

В обеих моделях представлено множество таблиц. Это в конечном итоге и определяет возможность реализации *ОП* и *ОПР*.

В плане отличия моделей отмечено:

1. ИТВ в общем случае представлены нереляционными таблицами.
2. В ИТВ в общем случае отсутствуют ключевые поля.
3. ИТВ не связаны между собой (отсутствуют внешние ключи).

Таким образом для выполнения *ОП* и *ОПР* необходима *СО*, которая позволит оценить ИТВ, а также результат преобразования РБД.

Поэтому модель примет вид рис. 4.5.

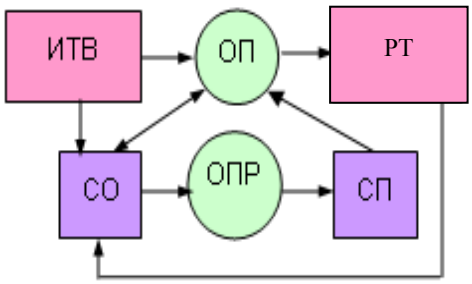


Рис. 4.5. Четвертая операторная форма модели преобразования ИТВ в РБД

Как видно из рис. 4.5 в модель добавился блок *СП* – система преобразований, назначенных разработчиком. Блок введен в связи с реальным положением дел и в соответствии с правилами операторной формы. Теперь можно записать: *РТ = ОП (ИТВ, СО, СП); СП = ОПР (СО)*.

Операторная модель такого рода, использующая укрупненные модели ИТВ и РБД позволит решать задачи преобразования ИТВ в РБД в редких частных случаях. Действительно, если *СО* позволит сделать вывод о том, что пункты 1 – 3 не выполняются, тогда функции *ОП* или *ОПР* сводятся к импорту таблиц.

Функции *СО* может взять на себя разработчик БД, если ИТВ представлена незначительным количеством таблиц небольшого размера и мощности. Тогда *ОПР* выродится в формирование разработчиком указаний инструментальной СУБД по импорту данных из ИТВ в РТ, а *ОП* сведется к выполнению этих указаний средствами инструментальной СУБД. Так как *ОПР* и *ОП* представляют собой систему операторов, то оправданно выделить операторы импортирования, соответственно *ОПРи* и *ОПи*.

Тогда в рассматриваемом частном случае справедливы следующие выражения: *РБД = ОПи (ИТВ, СО, СП); СП = ОПРи (СО).*

Средства импорта данных широко описаны в литературе по инструментальным СУБД [67 - 69]. В связи с этим для рассматриваемого частного случая реализация *ОПи* и *ОПРи* не вызывает затруднения, так как она уже в основном выполнена.

К сожалению, этот частный случай исключительно редок, но все равно, средства импорта данных, как правило, используются во всех случаях, и поэтому практически всегда задействованы соответствующие средства операторов *ОПи* и *ОПРи*.

Что касается *СО*, то она задействуется всегда и ее разработка – предмет специальных исследований.

Для последовательного развертывания динамической модели, если подходить к этому вопросу формально, необходимо рассмотреть все возможные сочетания пунктов несоответствия модели ИТВ и модели РБД. Обозначим несоответствия ИТВ и РБД следующим образом:

НР – ИТВ - нереляционные таблицы;

НК – в ИТВ отсутствуют ключевые поля;

НС – таблицы ИТВ не связаны между собой (отсутствуют внешние ключи).

Таблица возможных сочетаний в этой системе примет вид таблицы 4.1.

Таблица 4.1

№	НР	НК	НС
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Таким образом, если подходить к задаче построения и расширения динамической модели преобразования формально, то необходимо отследить

8 вариантов возможных сочетаний несоответствия моделей ИТВ и РТ. Более того, как известно, существуют 2 требования к ключевым полям и несколько требований к реляционным таблицам, которые в таблице не отражены. Рассмотрим последовательно сочетания и примем решение об их отражении в динамической модели.

Сочетание N1.

Это сочетание признаков несовпадения моделей ИТВ и РТ по сути свидетельствует о том, что эти модели совпадают. В связи с этим проблема преобразования ИТВ в РБД по сути, сводится к импорту данных ИТВ в инструментальную СУБД. Однако вызывает большое сомнение, что в ИТВ каким-либо образом описаны связи между таблицами. Эта ситуация маловероятна и поэтому не отражена в модели ИТВ. Но если такого рода ситуация случится, то оператор *ОП* сводится к выполнению стандартных средств импорта в инструментальную СУБД таблиц ИТВ *ОПи*, а оператор *ОПР*, сводится к управлению разработчиком этими средствами *ОПРи*. Кроме того, *ОП* и *ОПР* реализуют формальные операции по формированию связей между таблицами в инструментальной СУБД, которые будут дублировать связи в ИТВ.

Сочетание N2.

Это более вероятное сочетание – таблицы в ИТВ не связаны (не определены внешние ключи). Данный вариант похож на предыдущий вариант, но для выполнения преобразования ИТВ в РБД необходимо использование операторов формирования связей *ОПс* и *ОПРс*.

Операторная модель преобразования ИТВ в РБД примет вид рис. 4.6.

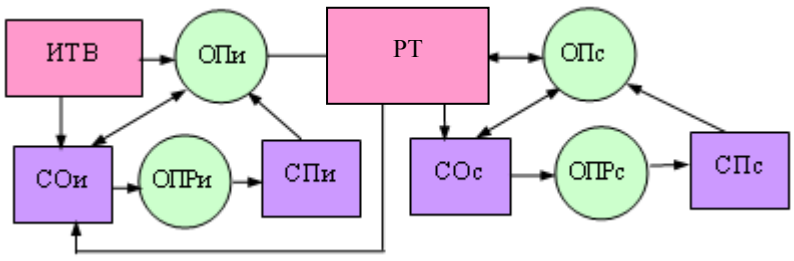


Рис. 4.6. Пятая операторная форма модели преобразования ИТВ в РБД

В связи с тем, что в рассматриваемом случае таблицы ИТВ реляционные, то посредством выполнения операторов *ОПи* и *ОПРи* выполняется их экспорт в РБД, а посредством операторов *ОПс* и *ОПРс* формируются связи между таблицами РБД или выявляются внешние ключи. Здесь *СОс* – система оценки всех возможных пар таблиц РБД на предмет наличия связей. Так как известно 4-е типа связей между реляционными таблицами, то оператор *ОПс* по сути представляет 4-е оператора $ОПс_1$, $ОПс_2$, $ОПс_3$, $ОПс_4$, которые выполняются последовательно друг за другом. Этим операторам ставятся в соответствие $СОс_i$, $ОПРс_i$ и $СПс_i$. Из соображений компактности представления модели эти операторы на ней не отражены.

Для данного случая справедливы следующие операторные выражения:

РБД = ОПи (ИТВ, СОи, СПи, ОПс (РБД, СОс, СПс));

СПи = ОПРи (СОи); СПс = ОПРс (СОс).

Сочетание N3.

Данное сочетание признаков несовпадения моделей свидетельствует о том, что ИТВ представлена реляционными, связанными таблицами, но ключевые поля в этих таблицах отсутствуют. В этом случае задача преобразования сводится к импорту ИТВ в РБД и назначению в таблицах РБД ключевых полей.

Операторная модель преобразования ИТВ в РБД в этом случае примет вид рис. 4.7.

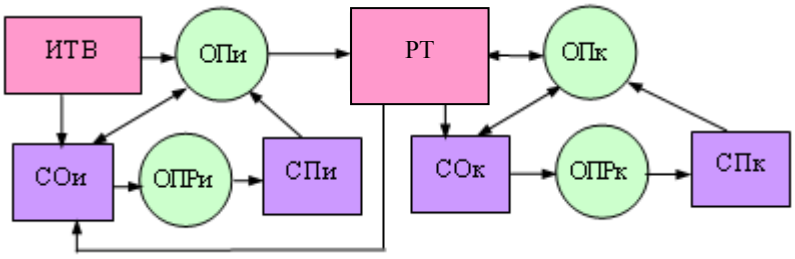


Рис. 4.7. Шестая операторная форма модели преобразования ИТВ в РТ

Внешне этот рисунок совпадает с предыдущим. Однако смысл двух операторов и двух систем оценок отличается.

Здесь:

- *ОПк* – оператор назначения ключевых полей в таблицах РБД;
- *СОк* – система оценок таблиц РБД на предмет выявления и назначения ключевых полей;
- *ОПРк* – оператор преобразования таблиц РБД разработчиком с целью назначения этим таблицам ключевых полей. Преобразования осуществляются на основе использования *Сок*;
- *СПк* – система преобразований РБД, инициируемая разработчиком РБД, которая ориентирована на назначение ключевых полей в соответствии с предъявляемыми к ним требованиями.

В этом случае справедливы следующие операторные выражения.

РБД = ОПи (ИТВ, СОи, СПи, ОПк (РБД, СОк, СПк));

СПк = ОПРк (СОк); СПи = ОПРи (СОи).

Подставив последние два выражения в предыдущее выражение, получим: *РБД = ОПи (ИТВ, СОи, ОПРи(СОи), ОПк (РБД, СОк, ОПРк (СОк)))*

Полученное выражение формально описывает методику преобразования ИТВ в РБД на основе использования введенных операторов и систем оценки.

Сочетание N4.

В данном случае выполняется преобразование таблиц ИТВ, которые являются реляционными, но между таблицами нет связей, и в таблицах отсутствуют ключевые поля. В этом случае таблицы ИТВ можно импортировать в РТ, затем назначить им ключевые поля и только после этого сформировать связи.

Связи между таблицами РБД назначаются посредством ключевых полей. Поэтому выполнение операторов в строго указанной последовательности важно.

Операторная модель преобразования ИТВ в РБД в этом случае будет представлять совокупность двух предыдущих моделей и выглядит следующим образом (рис. 4.8).

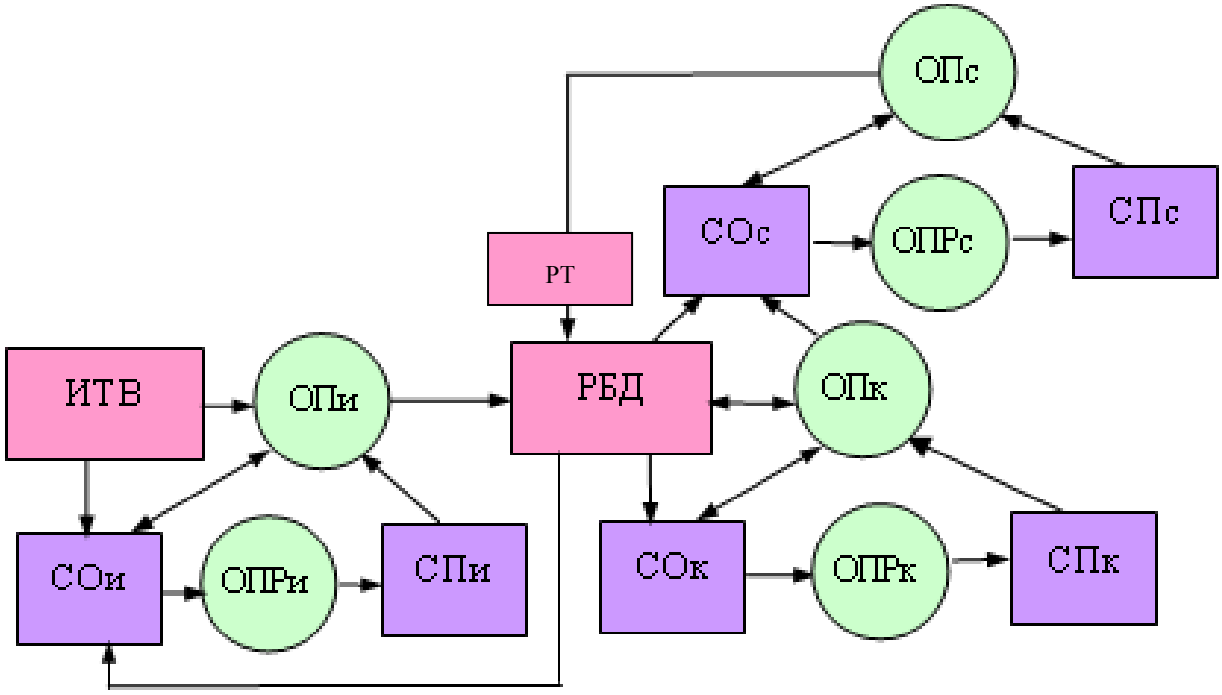


Рис. 4.8. Седьмая операторная форма модели преобразования ИТВ в РБД

Как следует из технологии формирования связей между таблицами РБД, связи формируются из анализа таблиц РБД и их ключевых полей. Поэтому *СОс* (система оценок РБД и ключевых полей на наличие связей между таблицами) в динамической модели преобразования ИТВ в РТ связана с *ОПк* (оператором назначения ключевых полей). В данном случае операторные выражения примут вид:

РБД = ОПи (ИТВ, СОи, СПи, ОПк, ОПс); СПи = ОПРи (СОи);

СПк = ОПРк (СОк);РБД = ОПс (РБД, СОс, СПс); СПс = ОПРс (СОс).

Выполнив соответствующие подстановки, получим:

РБД = ОПи (ИТВ, СОи, ОПРи (СОи)), ОПк (РБД, СОк, ОПРк(СОк), ОПс (РБД, СОс, ОПРс (СОс))

Сочетания N5 – N8.

Во всех оставшихся сочетаниях признаков несовпадения моделей необходимо предусмотреть отслеживание случаев, когда таблицы в ИТВ представлены нереляционными таблицами.

В связи с тем, что в подавляющем большинстве случаев нереляционные таблицы не могут быть импортированы в инструментальные СУБД, этап преобразования таблиц ИТВ должен выполняться до этапа их импорта в РБД. Кроме того, следует принять во внимание, что после преобразования нереляционных таблиц в реляционный вид, вероятнее всего может потребоваться выполнение этапов назначения ключевых полей и формирования связей. Это связано с тем, что после преобразования нереляционных таблиц они обычно меняют свою структуру.

С учетом вышесказанного операторная модель примет вид рис. 4.9.

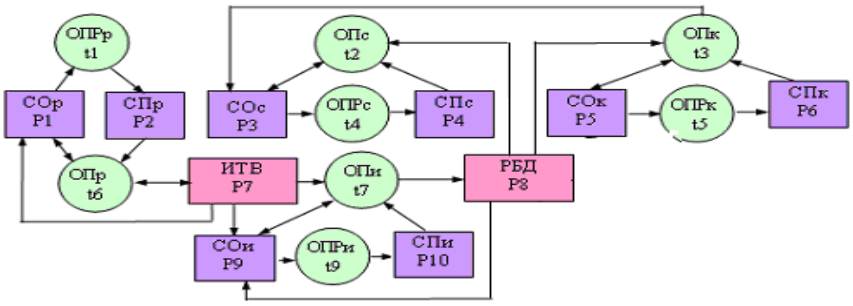


Рис. 4.9. Восьмая операторная форма модели преобразования ИТВ в РБД

К предыдущей операторной форме представления операторной модели добавились следующие компоненты: *ОПРр, ОПр, СОр, СПр.*

ИТВ = (ИТВ, СОр, СПр); СПр = ОПРр (СОр).

Подставив эти компоненты в предыдущую операторную запись, получим:

РБД = ОПи (ОПр (ИТВ, СОр, ОПР(СОр)), СОи, ОПРи(СОи), ОПк (РБДр, СОк, ОПРк (СОк)), ОПс (РБДр, СОс, ОПРс (СОс))

Таким образом, построена операторная модель преобразования ИТВ в РТ, которая учитывает все возможные сочетания признаков несовпадения моделей ИТВ и РТ.

Эта модель отражает методику преобразования ИТВ в РТ. С другой стороны, она позволяет сделать заключение о необходимом составе операторов преобразования, реализуемых программными средствами *(ОП)*, операторов преобразования, реализуемых разработчиком *(ОПР)*, систем оценки текущего состояния дел в процессе преобразования *(СО)*, систем преобразований, назначаемых разработчиком *(СП)*. Из рис. 4.9 можно сделать вывод о минимальном составе операторов, которые необходимо реализовать в форме средств, алгоритмов и методов для обеспечения методики преобразования ИТВ в РТ. К ним относятся операторы, перечисленные ниже.

ОП = {ОПр, ОПи, ОПк, ОПс}

ОПР = {ОПРр, ОПРи, ОПРк, ОПРс}

СО = {СОр, СОи, СОк, СОс}

СП = {СПр, СПи, СПк, СПс}

Как следует из сказанного выше об этапах назначения ключевых полей и формировании связей между таблицами, необходимо реализовать еще ряд следующих операторов.

$ОПс = \{ОПс_1, ОПс_2, ОПс_3, ОПс_4\}$

$ОПРс = \{ОПРс_1, ОПРс_2, ОПРс_3, ОПРс_4\}$

$СОс = \{СОс_1, СОс_2, СОс_3, СОс_4\}$

$СПс = \{СПс_1, СПс_2, СПс_3, СПс_4\}$

Кроме того, к реляционным таблицам предъявляется ряд требований, которые будут детально проанализированы при разработке соответствующих средств преобразования ИТВ. В связи с этим *ОПр, ОПРр, СОр, СПр*, по-видимому, будут представлены несколькими элементами.

Что касается средств назначения ключевых полей *ОПк, ОПРк, СОк, СПк,* то они, вероятно, не будут детализированы, так как к ключевым полям предъявляются взаимозависимые требования – уникальность и минимальность.

До разработки операторной модели преобразования ИТВ в РТ методика преобразования представлялась чисто интуитивно. Реализация алгоритмов и средств преобразования ИТВ в РТ в этом случае чревата ошибками, как минимум по трем причинам. Во-первых, последовательность использования разрабатываемых способов и алгоритмов должна быть регламентированной, а не произвольной. Во-вторых, взаимосвязь между компонентами подсистемы преобразования при интуитивном представлении методики и ее вербальном описании неочевидна. В третьих, неформально затруднительно в полном объеме определить состав необходимых компонент системы преобразования.

Все перечисленные операторы и системы оценок представляют собой средства, которые необходимо разработать. Их разработке посвящено приложение 1.

Построенная операторная модель преобразования ИТВ в РТ обладает еще одним ценным качеством. Она может быть использована для исследования характеристик подсистемы преобразования до ее реализации, для исключения принципиальных ошибок в подсистеме на начальных этапах ее реализации. Для достижения этих целей оправдано использование математического аппарата, который ориентирован на решение названных задач.

4.3 Исследование методики на предмет выявления и исключения концептуальных ошибок

Мотивация перехода от операторной модели к модели, построенной на основе аппарата сетей Петри

Операторная модель, построенная на основе анализа отличий моделей ИТВ и РТ, позволила выделить основные компоненты человеко-машинной подсистемы преобразования, определить порядок их использования, сформировать связи между ними. Таким образом, она может быть использована в качестве начальной формализации для разработки средств

преобразования ИТВ в РТ. Однако нет полной уверенности в том, что эта модель адекватна предмету исследования и свободна от концептуальных ошибок. Во избежание ошибок на ранних этапах разработки подсистемы необходимо исследовать следующие ее характеристики:

- выполнение баланса входных и выходных информационных потоков для всех систем принятия решения (устойчивость);
- достижимость всех состояний человеко-машинной системы (живость);
- отсутствие тупиковых ситуаций в процессе преобразования (живость);
- отсутствие ситуаций, когда подсистема приходит в нестационарное состояние и число информационных потоков превышает критическую отметку (безопасность);
- интегральные временные характеристики подсистемы.

Несмотря на достоинства операторной модели, она не позволяет выполнить исследование перечисленных характеристик подсистемы.

Методика преобразования ИТВ в РТ – это человеко-машинный процесс, поэтому в качестве ее модели оправданно использовать частный случай алгоритмических моделей – имитационные [74 - 75]. С точки зрения представления объектов на начальных этапах проектирования методики в ее рамках используются такие понятия, как ИТВ, РТ, разработчик, способы, системы оценок. Причем, эти объекты, как правило, не конкретизируются, а лишь фиксируется факт их наличия, их взаимосвязи, место в системе преобразования, последовательность и правила их использования. Такое представление наиболее близко к системному уровню, а на данном уровне преимущественно применяют модели систем массового обслуживания и сети Петри [32, 53,74-75]. Кроме того, аппарат сетей Петри позволяет исследовать моделируемые процессы на устойчивость, живость, безопасность и позволяет тем самым расширить круг исследований по сравнению с

операторной моделью. В свою очередь операторная модель служит прототипом для разработки сетевой модели.

На этом основании и с учетом специфики особенностей объекта моделирования в качестве основного математического аппарата для моделирования и предварительного исследования методики по преобразованию ИТВ в РБД выбран аппарат сетей Петри.

Сеть Петри состоит из трех элементов: множество мест *S*, множество переходов *T* и отношение инцидентности *F*. [32, 53,74-75].

Простой сетью Петри называется набор *N = (S, T, F)*, где

1. $S=\{s_1,\ldots,s_n\}$ - множество мест;

2. $T=\{t_1,\ldots,t_n\}$ - множество переходов таких, что $S \cap T = \varnothing$.

3. $F \subseteq \mu S \times T \times \mu S$ - отношение инцидентности такое, что

(а) $\forall \langle Q_1', t_1, Q_1'' \rangle, \langle Q_2', t_2, Q_2'' \rangle \in F : \langle Q_1', t_1, Q_1'' \rangle \neq \langle Q_2', t_2, Q_2'' \rangle \Rightarrow t_1 \neq t_2$;

(б) $\{t \mid \langle Q', t, Q'' \rangle \in F\} = T$.

Места нередко называют позициями или положениями.

Условия в пункте 3 говорят, что для каждого перехода $t \in T$ существует единственный элемент $\langle Q', t, Q'' \rangle \in F$, задающий для него входное мультимножество мест Q' и выходное мультимножество Q''. Дадим определение входному и выходному мультимножеству.

Определение: входное и выходное мультимножества мест и переходов

Пусть задана сеть *N=(S,T,F).*

Если для некоторого перехода *t* имееся $\langle Q', t, Q'' \rangle \in F$, то обозначают

$${}^\bullet t = Q' = \{(s,n) \mid (t,n) \in s^\bullet\}, t^\bullet = Q'' = \{(s,n) \mid (t,n) \in {}^\bullet s\};$$

$${}^\bullet s = \{(t,n) \mid (s,n) \in t^\bullet\}, s^\bullet = \{(t,n) \mid (s,n) \in {}^\bullet t\}.$$

Говорят, что $\left\| {}^\bullet t \right\|$ - входные, а $\left\| t^\bullet \right\|$ - выходные места перехода *t*. Таким образом, согласно определению, справедливо $\forall t \in T : \left\langle {}^\bullet t, t, t^\bullet \right\rangle \in F$.

Место s инцидентно переходу t, если $s \in {}^{\bullet} t$ или $s \in t^{\bullet}$.

Функции ${}^{\bullet}()$ и $()^{\bullet}$ могут быть расширены на мультимножества переходов. Пусть $\Theta \in \mu T$ есть мультимножество переходов такое, что $\Theta = n_1 t_1 + n_2 t_2 + ... + n_k t_k$. Тогда положим

$${}^{\bullet}\Theta = n_1 {}^{\bullet} t_1 + n_2 {}^{\bullet} t_2 + ... + n_k {}^{\bullet} t_k$$

$$\Theta^{\bullet} = n_1 t_1^{\bullet} + n_2 t_2^{\bullet} + ... + n_k t_k^{\bullet}.$$

Сети Петри имеют удобную графическую форму представления в виде графа, в котором места изображаются кружками, а переходы прямоугольниками. Причем место s соединяется с переходом t, если $(s,n) \in {}^{\bullet} t$ и t соединяется с s, если $(s,n) \in t^{\bullet}$ для некоторого натурального числа $n \in N$. Здесь число n называется кратностью дуги, которое графически изображается рядом с дугой. Дуги, имеющие единичную кратность, будут обозначаться без приписывания единицы.

Операторная модель, предложенная в работе, может быть представлена набором $M = (C, O, L)$, где

1. $C = CO \cup СП \cup MO$

$CO = \{CO_1, ..., CO_q, ..., CO_k\}$ – множество систем оценок проектных решений;

$СП = \{СП_1, ..., СП_f, ..., СП_z\}$ – множество систем принятия решения;

$MO = \{MO_1, ..., MO_t, ..., MO_p\}$ – множество моделей.

$CO \cap СП \cap MO = \emptyset$

Для исходной операторной модели множества представлены следующим образом:

$$CO = \{CO_p, CO_c, CO_k, CO_u\};$$

$$СП = \{СП_p, СП_c, СП_k, СП_u\};$$

$MO = \{ИТВ, РТ\}.$

При уточнении операторной модели состав множеств может быть модифицирован.

2. $O = \{ОПР_р, ОП_с, ОП_к, ОПР_с, ОПР_к, ОП_р, ОП_и, ОПР_и\}$

При уточнении операторной модели состав множеств может быть модифицирован.

3. $L \subseteq \mu C \times O \times \mu C$ – отношение инцидентности такое, что

(а) $\forall \langle G_1^{'}, o_1, G_1^{''} \rangle, \langle G_2^{'}, o_2, G_2^{''} \rangle \in L : \langle G_1^{'}, o_1, G_2^{''} \rangle \Rightarrow o_1 \neq o_2$;

(б) $\{o \mid \langle G^{'}, o, G^{''} \rangle \in L\} = O$.

Условие в пункте 3 говорит о том, что для каждого оператора $o \in O$ существует единственный элемент $\mid \langle G^{'}, o, G^{''} \rangle \in L$, задающий для него входное мультимножество систем оценки и принятия решений $G^{'}$ и выходное мультимножество $G^{''}$.

Входное и выходное мультимножества систем оценки и принятия решений, операторов операторной модели определяются по аналогии с входным и выходным мультимножествами мест переходов сети Петри.

Формирование сетевой модели

Между набором *N = (S, T, F)* сети Петри и набором *M = (C, O, L)* операторной модели установим соответствие таким образом, что между элементами наборов обеспечивалось взаимно-однозначное соответствие: $C \leftrightarrow S$; $O \leftrightarrow T$; $L \leftrightarrow F$. *[32]*

Как следствие установленного соответствия:

$|C| = |S|$; $|O| = |T|$; $|L| = |F|$.

В результате получена сетевая модель методики преобразования ИТВ в РТ, которая использована для исследования и уточнения операторной модели. Начальный уровень представления интерактивного взаимодействия в процессе преобразования ИТВ в РБД в форме сети Петри разработан на основе операторной модели (рис. 4.10).

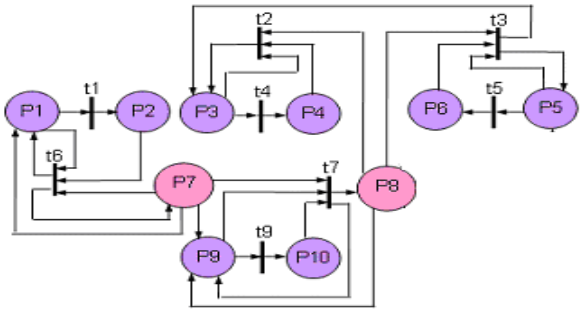


Рис. 4.10. Сеть Петри, соответствующая операторной модели преобразования ИТВ в РТ

Модели ИТВ, модели РТ *СО, СП* поставлены в соответствие положения сети *{P}*. Операторам *ОП, ОПР* поставлены в соответствие переходы сети *{t}*. Имена положений и переходов сети Петри соответствуют номерам компонент динамической модели.

Сеть Петри представляется тройкой $\boldsymbol{P} = <P, T, H>$, где

P – множество положений; T – множество переходов; H – множество маркеров.

С помощью построенной модели можно выполнить анализ устойчивости сети или анализ способности отражать реальные процессы интерактивного взаимодействия разработчика и автоматизированных средств в ходе проектирования РБД на основе существующей ИТВ.

Сеть Петри является устойчивой, если она имеет потоковое назначение $\varphi_i > 0$ для каждого $t_i \in T$. Потоковое назначение – это функция, которая приписывает каждому переходу $t_i \in T$, положительное рациональное число, называемое ее потоком [93]. Потоковое назначение для сети Петри должно удовлетворять требованиям:

- каждая дуга переносит поток, равный потоку перехода, к которому присоединена эта дуга;
- для каждого положения сумма потоков входных дуг должна равняться сумме потоков выходных дуг.

Пусть для каждого перехода сети рис. 2.14 назначен поток φ_i. Для каждого положения P_i запишем уравнения потоков, которые не должны противоречить друг другу, если данная сеть устойчива. Уравнения сведены в таблицу 4.2.

Таблица 4.2

	t1	t2	t3	t4	t5	t6	t7	t9	
P1	-φ1					-φ6+φ6			=0
P2	+φ1					-φ6			=0
P3		+φ2 -φ2	+φ3	-φ4					=0
P4		-φ2		+φ4					=0
P5			+φ3 -φ3		-φ5				=0
P6		-φ3		+φ5					=0
P7					-φ6 +φ6	-φ7			=0
P8		-φ2	-φ3			+φ7	-φ8+φ8		=0

Перепишем полученные уравнения в более компактной форме:

- φ1 - φ6 + φ6 = Ø; + φ1 - φ6 = Ø; + φ2 - φ2 + φ3 - φ4 = Ø; - φ2 + φ4= Ø;

+ φ3 - φ3 + φ5 = Ø; - φ7 - φ6 + φ6 = Ø;

- φ2 - φ3 + φ7 = Ø; - φ7 + φ7 - φ9 = Ø; - φ7 + φ9 = Ø;

Преобразовав уравнения, получим:

φ1 = Ø; φ1 = φ6; φ2 = φ4; φ7 = Ø;

φ7 = φ2 + φ3; φ9 = Ø; φ7 = φ9.

Выполнив соответствующие подстановки, получим:

φ1 = φ6 = φ7 = φ9 = Ø,

φ2 = φ4;

φ2 = -φ3;

То есть часть потоковых значений − нулевые. Это противоречит условию устойчивости системы.

В связи с этим рассмотренная сеть не является устойчивой, а, следовательно, процессы, ею описываемые, также являются неустойчивыми.

Поэтому модель нуждается в модификации, иначе методика, построенная на базе полученной модели, вероятно, будет содержать ошибки. Как видно из системы уравнений, нулевые потоковые значения, которые привели к нулевым потоковым значениям для всех положений, связаны с положениями *P1, P7, P9*. Естественно предположить то, что если исправить ситуацию для перечисленных положений, то она исправится и для сети в целом.

Вернемся к прообразам положений *P1, P7, P9* и попытаемся выяснить существо ошибки.

P1 отражает в модели *COp* − систему оценки соответствия информации табличного вида требованиям, предъявляемым к реляционным таблицам. Из рис. 4.9. видно, что *COp* по выходам связана с двумя операторами, а по входам – с одним (связь с ИТВ не является связью с оператором). Из связи такого рода можно сделать ложный вывод о том, что *COp* получается непосредственно из ИТВ без всяких преобразований. На самом деле это не так. Необходимо выполнить ряд оценок, что сопряжено с действиями и соответственно с операторами. Поэтому в операторной модели преобразования введем соответствующий оператор, а в сети Петри − соответствующий переход *t11*. (Этот оператор и другие компоненты обновленной динамической модели отражены на рис. 4.11, который построен после выполнения всех необходимых манипуляций с сетью Петри).

Прообразом положения *P7* является ИТВ, прообразом положения *P9* является *COu* − система оценки импорта. Эти компоненты модели связаны между собой, поэтому вначале попытаемся нормализовать ситуацию с одной из компонент, а другую рассмотрим позже. *COu* по входу связана с одним оператором *ОПu*, а по выходу – с двумя операторами *ОПРu* и *ОПu*. Имеется связь между ИТВ и *COu* по входу, но эта связь предполагает, что оценки необходимых действий по импорту выполняются без участия специальных средств. На самом деле это не так. В процессе импорта задействуются специализированные средства СУБД, использование которых не отражено. В

связи с этим в операторную модель рис. 4.9 необходимо добавить дополнительный оператор *OCOu* между ИТВ и *COu*, а в модель на основе сети Петри добавим соответствующий переход *t12*.

В результате выполнения модификаций получим сеть Петри, представленную на рис. 4.11**.**

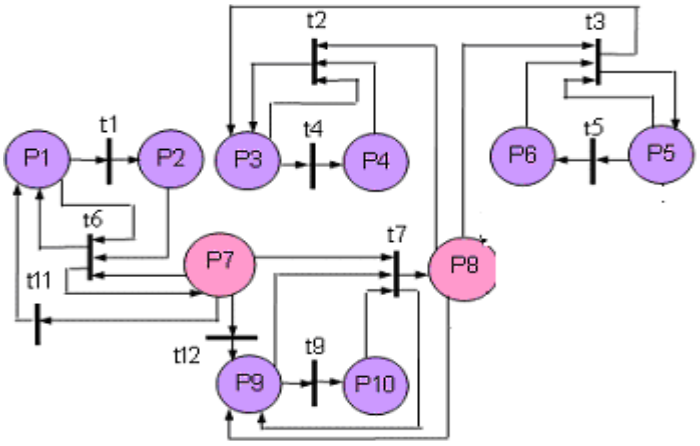


Рис. 4.11. Первая модификация сети Петри

Несмотря на выявление и исключение теперь уже очевидных ошибок, полной уверенности в том, что полученная сеть стала устойчивой, нет. Дело в том, что в сеть добавлены 3-и новых перехода. Тем самым изменено потоковое назначение сети не только для рассмотренных положений, но и для положений, связанных с ними через переходы. В связи с этим выполнен ряд модификаций операторной модели и соответствующей ее сети Петри.

Окончательная сетевая модель приведена на рис. 4.12, соответствующая ей операторная модель – на рис. 4.13.

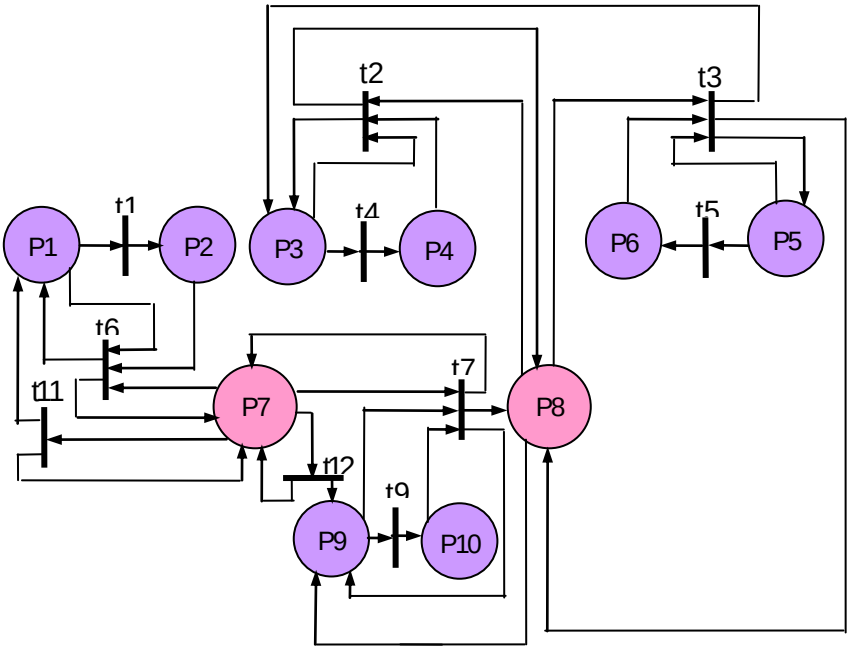


Рис. 4.12. Окончательная модификация сети Петри

Для полученной сети запишем уравнения потоков.

Уравнение потоков для положения P7 теперь выглядит следующим образом: $- \varphi 6 + \varphi 6 - \varphi 6 + \varphi 7 - \varphi 11 + \varphi 11 - \varphi 12 + \varphi 12 = \emptyset$, или $\emptyset = \emptyset$.

Таким образом, уравнение превратилось в тождество, и противоречия нет. Уравнение потоков для положения P8 теперь выглядит следующим образом: $- \varphi 2 - \varphi 3 + \varphi 3 + \varphi 7 = \emptyset$, или $\varphi 7 = \varphi 2$.

Противоречия нет.

В результате выполненных преобразований исходной сети Петри получена устойчивая сеть. Функционирование такой сети не приведет к коллизиям перемещения информационных потоков. А так как полученная сеть взаимно однозначно соответствует операторной модели (рис. 4.13), то модель методики преобразования ИТВ в РТ является устойчивой, что исключает принципиальные ошибки в методике на ранних этапах ее разработки и реализации.

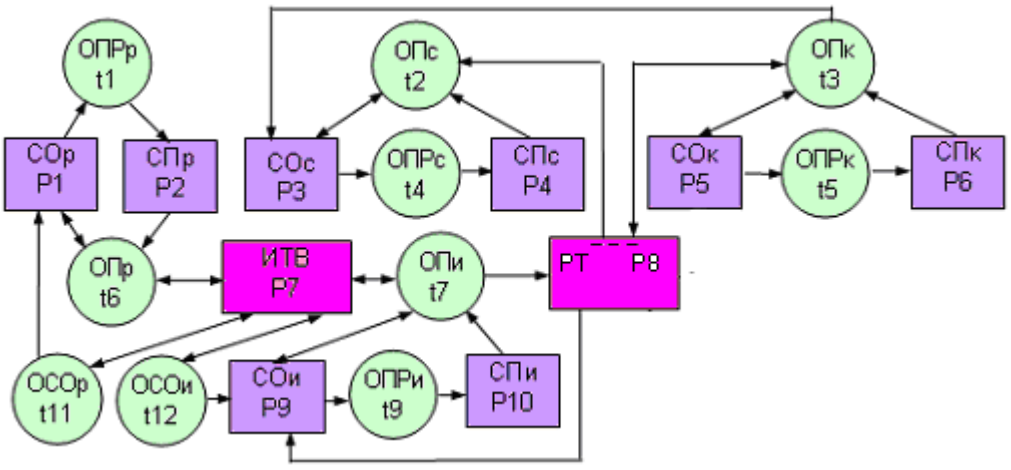


Рис. 4.13. Операторная модель преобразования ИТВ в РБД

Даже на данной стадии описания и исследования операторной модели она представляет ценность. Операторная модель позволяет сделать вывод о необходимом составе:

- операторов (способов, алгоритмов и средств), которые следует разработать для реализации полнофункциональной системы;
- системы оценок или функций (способов, алгоритмов и средств), которые следует разработать для реализации полнофункциональной системы проектирования;
- связей между средствами в разрабатываемой системе.

Кроме того, операторная модель иллюстрирует порядок и правила использования средств, задействованных в процессе решения задач преобразования ИТВ в РТ в разрабатываемой подсистеме.

Что касается динамики функционирования сети Петри и соответствующей ей операторной модели, то она в данном случае отражена лишь отчасти и сводится к ориентации информационных потоков. Их можно визуально отслеживать и в результате делать необходимые умозаключения. Однако, для более глубокого анализа динамических свойств модели и соответствующего объекта исследования оправданно использование специальных средств – маркерных сетей Петри, которые рассмотрены ниже.

4.4 Исследование динамических свойств методики

Динамику функционирования системы можно моделировать перемещением маркеров в сети в соответствии с правилами перехода:

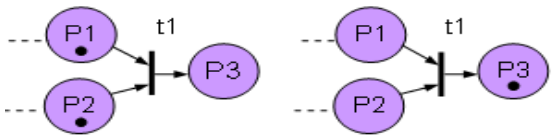
$M'(P) = M(P) - P(t_i) + H(t_i)$,

где $M(P) = (M(P1), M(P2), \ldots, M(PN))$ − разметка сети [93].

Рассмотрим процесс, описываемый сетью рис. 4.13 в динамике.

Предварительно отметим, что положения *P7* и *P8* соответствуют ИТВ и РТ, поэтому обладают особым статусом.

Инициация всего процесса преобразования начинается с ИТВ и осуществляется разработчиком. Кроме того, по достижении положения *P8*, соответствующего РТ, разработчик может инициировать процесс и в этом положении. На сетевой модели это отображается маркером, который изначально располагается в положении *P7*. По мере необходимости в *P7* может добавляться маркер. По достижении положения *P8*, в него может быть добавлен маркер. Это отражает реальные прерогативы разработчика. Собственно моделирование функционирования системы посредством маркеров состоит в следующем. Маркеры перемещаются по сети из положения в положение. При этом маркеры могут быть перемещены лишь в том случае, если сработает входной переход. А входной переход срабатывает тогда, когда во всех положениях, которые связаны с этим переходом по выходу, имеются маркеры. В качестве примера рассмотрим рис. 4.14.



а б

Рис. 4.14. Иллюстрация срабатывания перехода

Переход *t1* не сработает, пока в положении *P1* и *P2* не окажутся маркеры. Когда маркер попадают в положения *P1* и *P2*, возможно

срабатывание перехода *t1*. При этом маркеры из положений *P1* и *P2* удаляются и один маркер разместится в положении *P3*, что и проиллюстрировано на рис. 4.14,а и рис. 4.14,б.

Если данное правило поставить в соответствие реальным процессам, которые моделируются, то в нашем случае оператор выполняется лишь тогда, когда все данные, которые необходимы для его выполнения, имеются в наличии.

Перемещение маркеров по сети позволит сделать вывод о том, что:

- является ли сеть живой, т.е. есть ли принципиальная возможность срабатывания всех переходов;
- является ли сеть достижимой, т.е. достижимы ли все положения сети.

Проверка построенной сети представляет особый интерес, так как половина переходов в ней имеют больше одного входа, причем в основном они имеют по 3 входа. Возможность их срабатывания далеко неочевидна.

Для визуального отображения перемещения маркеров по сети с демонстрацией всех возможных состояний сети потребуется десятки рисунков подобных рис. 4.12 только с различной разметкой положений. В связи со значительным объемом информации, представленной в виде рисунков, и плохой обозримостью этой информации воспользуемся представлением разметки сети в виде кортежей. В этом случае порядковый номер в кортеже соответствует количеству маркеров в соответствующем положении. Начальной маркировке сети соответствует следующая разметка сети:

M1 = {0000001000}.

Срабатывание перехода *t11* приведет к очередной разметке сети. *M2 = {1000001000}.*

Срабатывание перехода *t12* приведет к очередной разметке сети. *M3 = {1000001010}.*

Срабатывание перехода *t1* приведет к очередной разметке сети. *M4 = {0100001010}.*

Срабатывание перехода *t12* приведет к очередной разметке сети. *M5 = {1000001010}.*

Срабатывание перехода *t9* приведет к очередной разметке сети. *M6 = {1000001001}.*

Срабатывание перехода *t7* приведет к очередной разметке сети. *M7 = {0000000100}.*

Срабатывание переходов может осуществляться неограниченное количество раз. Но при исследованиях выбран такой порядок перемещения маркеров, что для срабатывания всех переходов потребовалось менее сотни итераций.

В результате проверки сработали все переходы, поэтому анализируемая сеть Петри является достижимой, а значит и все состояния соответствующего объекта исследования, методики процесса преобразования ИТВ в РТ достижимы.

В реальной ситуации процесс завершается при достижении необходимого результата. Принципиальная возможность срабатывания всех переходов свидетельствует о еще одном свойстве сети – отсутствии тупиковых положений. Это в свою очередь свидетельствует о том, что в объекте моделирования отсутствуют тупиковые состояния.

Из анализа таблицы следует, что анализируемая сеть является безопасной, т.е. выполняется условие:

$\forall M(P) \subset R(P)\ \{\forall Pi\ /M\ (Pi) \leq 1\}$, где

$i = 1,n;\ n = /P/;\ R(P)$ − разметки сети.

Из этого следует, что моделируемая подсистема функционирует в стационарном режиме.

Резюмируя сказанное выше, можно сделать вывод о том, что разработанная сетевая модель обладает следующими свойствами:

устойчивая, живая, достижимая, безопасная, в ней отсутствуют тупиковые положения.

Между сетевой и операторной моделью обеспечено взаимно однозначное соответствие. Таким образом, и операторная модель методики преобразования обладает теми же свойствами. Поэтому операторная модель свободна от принципиальных ошибок и соответствует реальному процессу преобразования ИТВ в РТ.

В связи с этим разработанная операторная модель наряду с сетевой моделью может быть использована в качестве исходной формализации для разработки и реализации алгоритмов преобразования ИТВ в РТ, а также в качестве основы для разработки методики их использования.

В качестве другой формы представления методики преобразования ИТВ в РТ на рис. 4.15. приведена укрупненная схема взаимодействия разработчика и подсистемы в процессе преобразования ИТВ в РТ.

Следует обратить внимание, что под блоками “типовой процесс” подразумевается комплекс человеко-машинных процедур. Например, блок “Поиск ПК из нескольких атрибутов” включает в себя человеко-машинные процедуры поиска первичного ключа на основе двух и более атрибутов. И для каждого случая используются разные алгоритмы. Соответственно для каждого случая используется комплект программных процедур и комплект реакций разработчика, связанных с каждой процедурой.

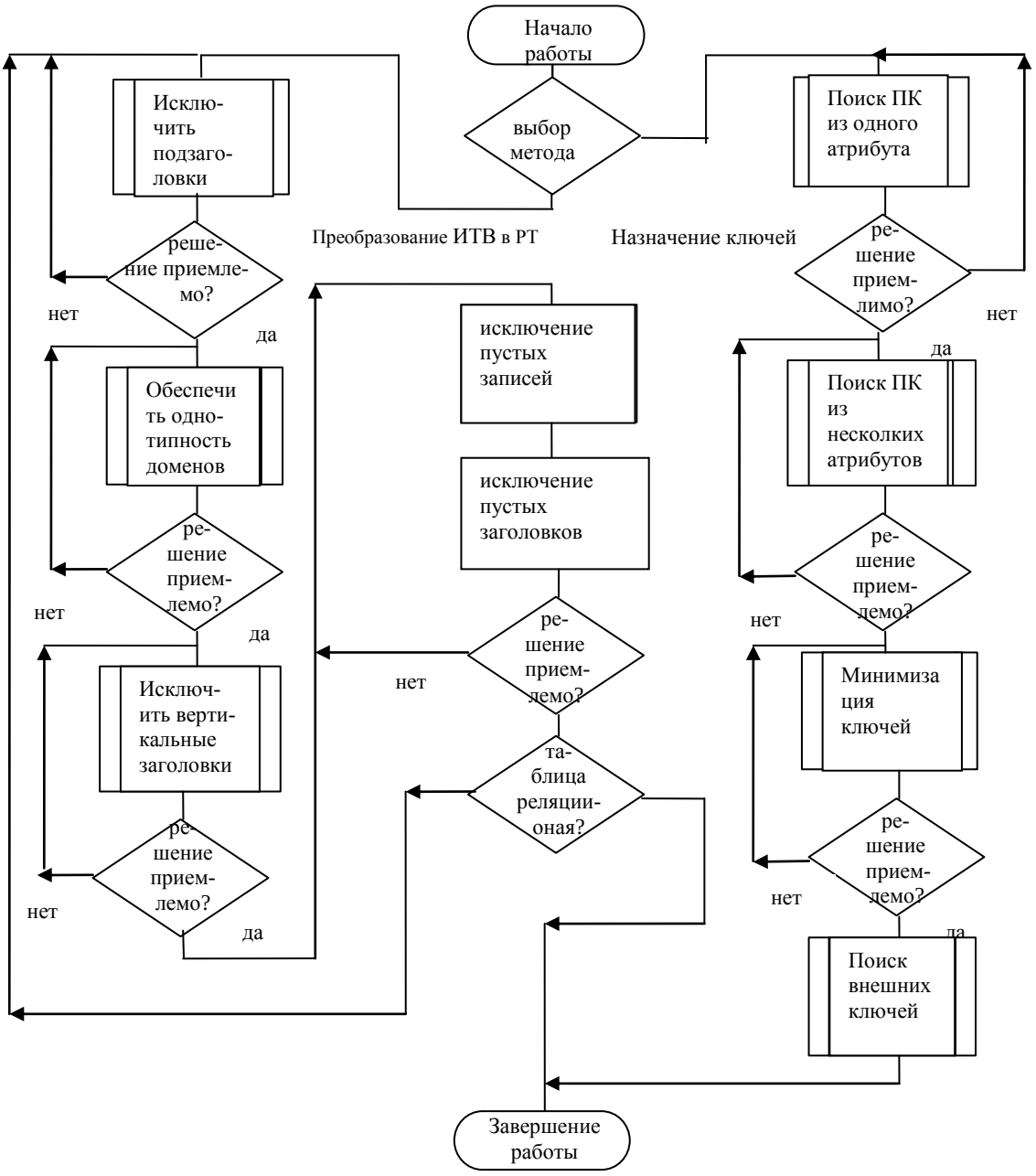


Рис. 4.15. Укрупненная схема взаимодействия разработчика и подсистемы в процессе преобразования ИТВ в РТ.

Выводы по главе 4

1. Специфика проблем, решаемых в работе, обусловила оправданность разработки модели методики преобразования ИТВ в РТ для комплексной реализации способов решения задач преобразования.

2. Детализация информационных моделей таблиц ИТВ и реляционных таблиц обеспечила их адекватность относительно задач, решаемых в процессе преобразования ИТВ в РТ.

3. Операторная модель методики позволила выявить ее основные компоненты, связи между ними, послужила в качестве исходной формализации для разработки модели методики на базе аппарата сетей Петри.

4. Модель методики в виде сети Петри позволила выявить и исключить принципиальные ошибки в проекте на ранних стадиях его разработки, исключить коллизии информационных потоков.

5. Операторная модель методики, скорректированная на основе использования сетевой модели, свободна от принципиальных ошибок и коллизий информационных потоков.

6. Построенные модели позволили исследовать динамические характеристики методики.

7. Предложенные сетевая и операторная модели методики могут быть использованы в качестве исходной формализации для разработки и реализации способов, алгоритмов и средств, ориентированных на решение общих и частных задач преобразования ИТВ в РТ.

ВЫВОДЫ И ЗАКЛЮЧЕНИЕ

В результате выполненных в диссертации исследований и разработок сделаны выводы.

1. ИТВ представляет собой информацию, которая интерпретируется заинтересованными в ней людьми двумерными таблицами.

2. В ИТВ, как правило, таблицы не удовлетворяют ни одному из требований к РТ, а потребность в использовании методов БД для работы с ИТВ существует.

3. В связи с этим сделан вывод о необходимости разработки методики преобразования ИТВ в РБД. При этом необходимо минимизировать трудоемкость преобразования и число ошибок преобразования.

4. Традиционная теория проектирования РБД это лучшее, что сегодня имеется для компактных, непротиворечивых РБД. Однако она отталкивается от схем отношений, когда таблицы не заполнены. А большинство проектных решений основывается на анализе данных, которых нет. Поэтому формализовать процесс проектирования РБД пока не удается

5. При наличии ИТВ, возможна формализация преобразования ИТВ в РТ на основе анализа имеющейся информации.

6. Методика преобразования ИТВ в РТ органично сочетается с традиционной методологией проектирования РБД. Современная методология проектирования РБД может быть использована для преобразования ИТВ в РТ как основа для формализации алгоритмов преобразования.

7. Разработчики РБД заинтересованы в наличии автоматизированных средств преобразования ИТВ в РТ. Потребители ИТВ заинтересованы в использовании преимуществ РБД при обработке информации.

8. Методика преобразования ИТВ в РТ в настоящее время в полном объеме не разработана. Современные практические разработки в области импорта, реляционных таблиц могут быть использованы только для таблиц,

представленных в реляционном виде, или как вспомогательные средства для решения задач преобразования малой размерности.

9. Для преобразования ИТВ в РТ необходимо решить следующие основные задачи: приведение ИТВ к реляционному виду; назначение первичных и внешних ключей;

10. В ИТВ нередко используются подзаголовки различного типа, что недопустимо в РТ. Разработанные в диссертации алгоритмы позволяют исключить из ИТВ подзаголовки всех типов.

11. В ИТВ, как правило, значения одного атрибута имеют различный тип, что не соответствует требованиям к реляционным таблицам. Предложенный алгоритм приведения значений атрибутов заполненных таблиц к одному типу позволяет в полном объеме обеспечить соответствующее требование к реляционным таблицам.

12. При наличии ИТВ, возможна формализация назначения ключевых полей на основе анализа имеющейся информации. Предложенный способ назначения первичных и внешних ключей на основе анализа ИТВ органично сочетается с традиционной методологией проектирования РБД.

13. В рамках методики преобразования таблиц ИТВ к РТ предлагаются структурные модели ИТВ и РТ. На основе разработанных алгоритмов предложена организация деятельности по преобразованию ИТВ к РТ. Предложены приемы по сбору, обработке и представлению нужной информации для достижения нужных результатов. В качестве корректировки новых и полученных ранее знаний в работе предлагается развитие способов назначения первичных и внешних ключей в плане использования разработанных средств формализации, использование которых оправдано при наличии ИТВ. Все это соответствует общему определению метода.

14. Разработанный комплекс программ позволяет реализовать методику, предложенную в диссертации. Аналитические и экспериментальные оценки процедур преобразования позволили сделать вывод о их применимости при работе с ИТВ средней сложности.

По результатам работы можно сделать следующее заключение о выполненных исследованиях и разработках.

1. Выполнен аналитический обзор современной теории проектирования РБД, сформулированы ее достоинства и недостатки, в результате обзора сделан вывод о том, что традиционная теория не гарантирует получение наилучшего проектного решения.

2. Определено понятие ИТВ. Сформулированы мотивы преобразования ИТВ в РТ. Выполнен анализ задач, возникающих в процессе преобразования ИТВ в РТ.

3. Выполнен анализ применимости современных теоретических и практических разработок для решения задач автоматизированного преобразования ИТВ в РТ, в результате чего сделан вывод, что эти разработки лишь отчасти применимы. Выполнена постановка проблемы формирования РТ на основе существующей ИТВ.

4. Определен состав алгоритмов, которые необходимо разработать для обеспечения эффективного решения задач преобразования ИТВ в РТ.

5. Разработан алгоритм исключения подзаголовков из ИТВ.

6. Разработан алгоритм приведения значений атрибутов заполненных таблиц к единому типу.

7. Разработан алгоритм исключения дублирования записей в ИТВ.

8. Предложен способ преобразования таблиц ИТВ в РТ на основе использования моделей ИТВ и РТ и разработанного комплекта алгоритмов.

9. Разработаны алгоритмы назначения ключевых полей в ИТВ.

10. Разработан способ назначения ключевых полей в ИТВ на основе использования моделей ИТВ и РТ и разработанного комплекта алгоритмов.

11. Проанализированы вопросы построения программного комплекса преобразования ИТВ в РТ.

12. Выполнена реализация программного комплекса преобразования ИТВ в РТ.

13. Выполнены аналитические и экспериментальные исследования емкостных и временных показателей основных модулей системы.

14. Разработана методика проектирования РТ с использованием заполненных ИТВ, которая основана на органичном использовании способа преобразования ИТВ в РТ, способа назначения ключевых полей в ИТВ и комплекта соответствующих процедур.

15. Разработанная методика прошла апробацию и внедрена для практического применения в МГТУ им. Н.Э. Баумана (использование в учебном процессе).

Отдельные вопросы диссертации более глубоко рассмотрены в работах [90 - 100].

ЛИТЕРАТУРА

1.Агальцов В.П. Базы данных. – М.: Мир, 2002. – 375 с.

2.Аграновский А.В., Арутюнян Р.Э, Хади Р.А. Современные аспекты проблемы поиска в текстовых базах данных//Телекоммуникации. – М., 2003. – №3. – С. 25– 23.

3.Ахаян Р., Горев А., Макатирипов С. Эффективная работа с СУБД. – СПб.: Питер, 1997. – 704 с.

4. Брешенков А.В. О перспективах развития информационных технологий при комплектации машин изделиями автотракторного электрооборудования//Автоэлектрооборудование. – М., 2001. –№1– С. 7–22.

5. Брешенков А.В. Методы решения задач проектирования реляционных баз данных на основе использования существующей информации табличного вида – М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. – 154 с.

6. Балдин А.В., Брешенков А.В. Анализ проблемы проектирования реляционных баз данных на основе использования существующей информации табличного вида//Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия Приборостроение. – М., 2007. – №2. – С. 66–80.

7. Балдин А.В., Брешенков А.В. Исследование временных свойств системы проектирования реляционных баз данных на основе использования информации табличного вида//Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия Приборостроение. – М., 2007. – №3. – С. 9–23.

8. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2000. – 384 с.

9. Григорьев Е. А. Представление идентифицируемых сложных объектов в реляционной базе данных//Открытые системы.– М., 2000.–№ 1– 2.

10. Григорьев Ю.А., Ревунков Г.И. Банки данных: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 320 с.

11. Дейт К. Дж. Введение в системы баз данных: Пер. с англ. – М.: Наука, 1980. – 464 с.

12. Дейт К. Дж. Введение в системы баз данных. 6-е изд.: Пер. с англ. – Киев: Диалектика, 1998. – 784 с.

13. Дейт К. Дж. Введение в системы баз данных. 7-е изд.: Пер. с англ. – М.: Вильямс, 2001. – 1072 с.

14. Замулин А.В. Системы программирования баз данных и знаний. – Новосибирск: Наука, 1990. – 352 с.

15. Корнеев В.В. и др. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2000. – 162 с.

16. Розмахов О.Г. Основы проектирования баз данных. – М.: Московский авиационный институт, 1993. – 24 с.

17. Chen P. P. – S.The Entity – Relationship Model – Toward Unified View of Data//ACM TODS. – March 1976. – 1, № 1. (Переиздано: M. Stonebraker (ed.) Readings in Database Systems. – San Mateo, Calif.: Morgan Kaufmann, 1988.)

18. Cleaveland J.C. An Introduction to Data Types//Reading, Mass.: Addison-Wesley, 1986.

19. Codd E.F. Data Base Sublanguage Founded on the Relational Calculus//Proc. 1971 ACM SIGFIDET Workshop on data Description, Access and Control. – San Diego, Calif. – November, 1971.

20. Codd E.F. A relational model data for larger shared data banks//Comm; ACM. – 1970. V.13 – № 6 – P. 377– 387.

21. Codd E.F. Further normalization of the database relational model, in data base systems (R. Rustin, ed.). Prentice Hall, Endlewood Cliffs, NJ, 1972.

22. Codd E.F. Recent Investigations into Relational Data Base Eystems//Proc IFIP Congress – Stockholm, Sweden, 1974

23. Codd E.F. The Relational Model For Database Management Version 2.

Reading, Mass.: Addison – Wesley, 1990.

24. Паспорт специальности 05.13.17. Теоретические основы информатики.

25. Информатика: Учебник//Под общ. ред. А.Н. Данчула. – М.: Изд-во РАГС, 2004. – 528 с.

26. Брешенков А.В. Неформальная постановка проблемы преобразования информации табличного вида в файлы баз данных//Сб. трудов АУ МВД России "Актуальные вопросы технологий в деятельности органов внутренних дел". – М., 2004. – С. 55– 70.

27. Брешенков А.В. Избавление от сложных атрибутов в заполненных нереляционных таблицах//Сб. трудов кафедры ИУ-6 – М.: Эликс+, 2006. – С. 10–15.

28. Брешенков А.В. Преобразование заполненных таблиц ко второй нормальной форме//Инженерное образование, 2007. – №2. – 16 с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0420700025/0005.)

29. Брешенков А.В. Приведение заполненных таблиц к третьей нормальной форме//Инженерное образование, 2007. – №4. – 15с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0420700025/0016.)

30. Брешенков А.В. Разработка модели методики проектирования реляционных баз данных на основе использования информации табличного вида//Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия Приборостроение. – М., 2007. – №2. – С. 40–56.

31. Брешенков А.В. Исследование методики проектирования реляционных баз данных на основе сетевой модели//Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия Приборостроение. – М., 2007. – №3. – С. 55–70.

32. Балдин А.В., Брешенков А.В. Анализ проблемы проектирования реляционных баз данных на основе использования существующей информации табличного вида//Вестник Московского государственного

технического университета им. Н.Э. Баумана. Серия Приборостроение. – М., 2007. – №2. – С. 66–80.

33. Брешенков А.В., Балдин А.В. Исследование временных свойств системы проектирования реляционных баз данных на основе использования информации табличного вида//Вестник Московского государственного технического университета им. Н.Э.Баумана. Серия Приборостроение. – М., 2007. – №3. – С. 9 – 23.

34. Брешенков А.В. Преобразование заполненных таблиц к первой нормальной форме // Инженерное образование, 2007. – №2. – 14 с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0420700025/0005.)

35. Брешенков А.В. Приведение заполненных таблиц к четвертой нормальной форме // Инженерное образование, 2007. – №4. – 15с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0420700025/0017.)

36. Брешенков А.В., Бараков Д. Д. Вопросы преобразования электронных таблиц в таблицы реляционных баз данных//Современные информационные технологии. Сб. трудов кафедры ИУ–6. – М.: Эликс +, 2004. – С. 44–50.

37. Брешенков А.В. Методология проектирования реляционных баз данных с использованием данных табличного вида. Дис. доктор техн. наук (05.25.05) – М., 2007

38. Хоменко А.Д., Цыганков В.М, Мальцев М.Г. Базы данных: Учебник для высших учебных заведений // Под ред. проф. А.Д. Хомоненко – 6-е изд. – СПб КОРОНА-Век: Бином-Пресс, 2007. – 736 с.

39. Date C. J. Why Quantifier Order Is Important // Date C. J. and Hugh Darwen. Relational Database Writings 1989 – 1991. – Reading, Mass.: Addison-Wesley, 1992.

40. Date C. J. What’s Wrong with SQL? // Date C. J. Relational Database Writings 1985 – 1989. – Reading, Mass.: Addison-Wesley, 1990.

41. Date C. J. How We Missed the Relational Boat // Date C. J. Relational Database Writings 1991 – 1994. – Reading, Mass.: Addison-Wesley, 1995.

42. Date C. J. Why Relational? // C.J. Date. Relational Database Writings 1985 – 1989. – Reading, Mass.: Addison-Wesley, 1990.

43. Date C. J.: “There’s Only One Relational Model!”, http://www.dbdebunk.com (February 2001).

44. Date C. J. What Not How: The Business Rules Approach to Appllication Devolopment.-Reading, Mass.: Addison-Wesley, 2000

45. Гэри Хансен, Джэймс Хансен. Базы данных: разработка и управление: Пер. с англ. – М.: Бином, 1999. – 699 с.

46. Ульман Дж. Основы систем баз данных: Пер. с англ. М.Р. Когаловского и В.В. Когутовского.– М.: Финансы и статистика, 1983. – 334 с.

47. Ульман Д., Уидом Д. Введение в системы баз данных: Пер. с англ. – М.: Лори, 2000. – 319 с.

48. Тихомиров Ю.В. Microsoft SQL Server 7.0. – СПб.: БХВ-Петербург, 1999. – 720 с.

49. Тихомиров Ю.В. Micrjsoft SQL Server 7.0: разработка приложений. – СПб.: БХВ – Петербург, 1999. – 352 с.

50. Аткинсон М., Бансилон Ф., Девитт Д., Дитрих К., Майнер Д., Здоник С. Манифест систем объектно-ориентированных баз данных СУБД. – М., 1995. – №4.

51. Атре Ш. Структурный подход к организации баз данных: Пер. с англ. – М.: Финансы и статистика, 1983. – 317 с.

52. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.

53. Брешенков А.В. Базы данных. Проектирование баз данных на основе информации табличного вида. – Germany: LAP LAMBERT Academic Publishing GmbH & Co. KG Dudweiler, rbr, 66123 Saarbrucken, 2011. – 394 с.

54. Бабанов А.М. Теория семантически значимых отображений и ее применение для проектирования реляционных баз данных: Дисс. … канд.

техн. наук (05.13.11). – М., 2005. – 182 с.

55. Берзтисс А.Т. Структуры данных. – М.: Статистика, 1974. – 408 с.

56. Арсеньев Б.П., Яковлев С.А. Интеграция распределенных баз данных. – СПб.: Лань, 2001. – 461 с.

57. Буре Р. XML и базы данных // Открытые системы. – М., 2000. – № 10. – С. 62– 65.

58. Eisenberg A., Melton J/ SQL: 1999, Formerly Known as SQL3 // ACM SIGMOD Record. – March 1999. – 28, № 4.

59. Брешенков А.В. Выявление и формирование связей один - к одному между заполненными реляционными таблицами: Сборник трудов №5 молодых ученых, аспирантов и студентов ”Информатика и системы управления в XXI веке” (часть 2). – М.: МГТУ им. Н.Э. Баумана, 2007. – С. 61– 65.

60. Брешенков А.В. Выявление и формирование связей один-ко многим в заполненных реляционных таблицах//Современные информационные технологии: Сб. трудов кафедры ИУ–6. – М.: Эликс+, 2010. Том 1. – С. 48–51.

61. Брешенков А.В. Выявление и формирование связей многие-ко многим в заполненных реляционных таблицах//Современные информационные технологии: Сб. трудов кафедры ИУ–6. – М.: Эликс+, 2010. Том 2– С. 60–69.

62. Брешенков А.В., Белоус В.В. Метод назначения первичных ключей в информации табличного вида//Инженерное образование, 2010. – №4. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0321000195)

63. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – 2-е изд. перераб. и доп.– М.: Энергоатомиздат, 1988. – 480 с.

64. Вентцель Е.С. Теория вероятностей. – М.: Высшая школа, 2001. – 576 с.

65. Колесников А. EXCEL 97 (русифицированная версия). – Киев:

BHV, 1998. – 480 с.

66. Энциклопедия пользователя. Oracle8.: Пер. с англ.//Компания Advanced Information Systems и др. – Киев: ДиаСофт, 1999. – 864 с.

67. Урман С. Oracle 8. Программирование на языке PL/SQL. – М.: Лори, 1999. – 607 с.

68. Тихомиров Ю.В. Microsoft SQL Server 7.0. – СПб.: БХВ-Петербург, 1999. – 720 с.

69. Тихомиров Ю.В. Micrjsoft SQL Server 7.0: разработка приложений. – СПб.: БХВ – Петербург, 1999. – 352 с.

70. Брешенков А.В. Разработка и исследование метода проектирования регистровых структур в интерактивном режиме: Дисс. … канд. техн. наук (05.13.12). – М., 1988. – 192 с.

71. Брешенков А.В. Интерактивный анализ регистровых структур // Тез. доклада Всесоюзной школы-семинара "Разработка и применение в народном хозяйстве ЕС ЭВМ /ЕС ЭВМ-85/". – Кишинев, 1985. – С. 17– 19.

72. Брешенков А.В. Структура программного обеспечения диалоговой системы анализа операционных устройств ЭВМ на уровне регистровых передач // Труды МВТУ им. Н.Э. Баумана. – М., 1987. – Вып. 482. – С. 31– 40.

73. Брешенков А.В., Павлов Ю.Е. Интерактивные средства генерации описания моделей регистровых структур // Тез. доклада Всесоюзной научно-технической конференции “Актуальные проблемы информатики, управления и вычислительной техники”. – М., 1987. – С. 19– 26.

74. Норенков И.П. Разработка систем автоматизированного проектирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 1994. – 207 с.

75. Норенков И.П. Основы автоматизированного проектирования: Учебник для вузов. – 2-е изд., переработ. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 336 с.

76. Брешенков А.В., Гудзенко Д.Ю., Казаков Г.И. Проектирование

реляционных баз данных на основе информации табличного типа: Учебное пособие – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 150 с.

77. Система управления базами данных. Руководство по проектированию структур данных. – М.: Информ Икс, 1997. – 14 с.

78. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981. – 368 с.

79. Зиглер К. Методы проектирования программных систем. – М.: Мир, 1985. – 328 с.

80. Иванова Г.С. Технология программирования. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. – 320 с.

81. Иванова Г.С., Ничушкина Т.Н. Проектирование программного обеспечения: Методическое пособие по выполнению и оформлению курсовых, дипломных и квалификационных работ. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 83 с.

82. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. – М.: Мир, 1992. – 406 с.

83. Хьюз Дж., Мичтом Дж. Структурный подход к программированию. – М.: Мир, 1980. – 278 с.

84. Бекаревич Ю.Б., Пушкина Н. В. Microsoft Access 2000. – СПб.: БХВ – Петербург, 2001. – 480 с.

85. Боровиков В.В. Microsoft Access 2002: Программирование и разработка баз данных и приложений. – М.: Солон – Р, 2002. – 560 с.

86. Харитонова И.А., Михеева В.Д. Microsoft Access 2000. – СПб.: БХВ-Петербург, 2001. – 819 с.

87. Харитонова И.А., Михеева Л.В. Рудикова. Microsoft Access 2007. – СПб.: БХВ-Петербург, 2008. – 1280 с.

88. Овчинников В.А. Автоматизация комбинаторно-оптимизационных задач при проектировании ЭВМ и систем: Учеб. для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 288 с.

89. Формирование связей многие ко многим в заполненных

реляционных таблицах//Вестник МГТУ им. Н.Э. Баумана. Серия Приборостроение. – М., 2011. – С. 105– 117

90. Брешенков А. В., Мин Т. Т. Мотивы разработки метода преобразования информации табличного вида в реляционное представление. // Инженерное образование, 2012. – №3 – 13 с (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0421200025.)

91. Брешенков А. В., Мин Т. Т. Аналитический обзор традиционного подхода формирования реляционных таблиц с учетом использования существующей информации табличного вида // Инженерное образование, 2012. – №8. – 16 с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0421200025.)

92. Брешенков А. В., Мин Т. Т. Модели реляционных таблиц и информации табличного вида// Инженерное образование, 2012. – №7. – 10 с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0421200025.)

93. Брешенков А. В., Мин Т. Т. Алгоритмы назначения первичных ключей в заполненных таблицах// Инженерное образование, 2012. – №6. – 14 с. (Наука и образование: Эл. науч. издание. Номер гос. регистрации 0421200025.)

94. Брешенков А.В., Мин Т.Т. Преобразование нереляционных таблиц к реляционному виду без использования сложных атрибутов//Вестник Московского государственного технического университета им. Н.Э.Баумана. – М.: МГТУ им. Н.Э.Баумана, 2012. – №2. – С. 59– 60.

95. Брешенков А.В., Мин Тхет Тин. Электронная цифровая подпись. Современные информационные технологии//Сб. трудов кафедры ИУ-6. – М.: НИИ РЛ МГТУ им. Н.Э. Баумана, 2011. – С. 112–115.

96. Брешенков А.В., Мин Тхет Тин. Исключение внутренних подзаголовков и избавление от сложных атрибутов при преобразовании нереляционных таблиц к реляционному виду // Современные информационные технологии: Сб. трудов кафедры ИУ-6. – М.: НИИ РЛ

МГТУ им. Н.Э. Баумана, 2011. – С. 176–183.

97. Брешенков А.В., Мин Т.Т. Вычислительная сложность процедур назначения первичных ключей в заполненных таблицах // Информатика и системы управления в XXI веке: Сб. трудов МГТУ им. Н.Э. Баумана. –М.: МГТУ им. Н.Э. Баумана, 2012.– №9. – С. 136–142.

98. Мин Тхет Тин, Брешенков А.В., Гудзенко Д.Ю. Назначение внешних ключей в заполненных реляционных таблицах//Современные компьютерные системы и технологии: Сб. трудов кафедры ИУ-6. – М.: МГТУ им. Н.Э. Баумана, 2012. – С. 128–135.

99. Мин Тхет Тин, Брешенков А.В., Гудзенко Д.Ю. Анализ типов атрибутовинформации табличного вида//Современные компьютерные системы и технологии: Сб. трудов кафедры ИУ-6. – М.: МГТУ им. Н.Э. Баумана, 2012. – С. 15—23.

100. Мин Т. Т. Анализ проблем разработки методики формирования реляционных таблиц на основе использования информации табличного вида. Россия в XXI веке: проблемы, тенденция, перспективы // Материалы XIV Международного симпозиума “Уникальные феномены и универсальные ценности культуры”: Сборник научных статей. – М.: МГТУ им. Н.Э. Баумана, 2012. – С. 270–272.

ПРИЛОЖЕНИЕ 1.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДИКИ ФОРМИРОВАНИЯ РЕЛЯЦИОННЫХ ТАБЛИЦ НА ОСНОВЕ ИНФОРМАЦИИ ТАБЛИЧНОГО ВИДА

1. Общие принципы разработки комплекса программ преобразования ИТВ в РТ

Так как программное обеспечение (ПО), с одной стороны, реализует закладываемые в него способы и алгоритмы преобразования ИТВ в РТ, а с другой стороны, организует общение разработчика и средств преобразования, эффективность ПО в значительной степени сказывается на эффективности использования средств автоматизированного проектирования [74, 75]. Ввиду того, что взаимодействие разработчика и комплекса программ осуществляется посредством экранных форм, а само взаимодействие ориентировано на обработку ИТВ, разработка ПО тесно связана с разработкой лингвистического и информационного обеспечений.

Так как программы преобразования ИТВ в РТ имеют значительный объемом и сложность, трудоемкость их разработки может составлять несколько человеко-лет, а объем – тысячи операторов языка высокого уровня [77]. С целью сокращения сроков разработки ПО системы интерактивного проектирования РБД на основе ИТВ использовался подход, изложенный в [53]. Основные его положения следующие:

- разработка и исследование комплекса программ и его отдельных компонент таким образом, который обеспечит выявление принципиальных ошибок в нем на ранних этапах его создания, и исключит ситуации, приводящие к необходимости перепроектирования;
- всестороннее использование опыта предыдущих разработок подобных комплексов, а также систем интерактивного взаимодействия как в плане применения эффективных методов создания сложных программных комплексов, так и в плане использования готовых программных модулей;

- проведение анализа требований к разрабатываемому комплексу программ в рамках всех этапов ее разработки, чтобы исключить возможность распространения ошибок проектирования от предыдущих этапов к последующим;
- детальное определение спецификаций подсистемы, унификация лингвистического и информационного обеспечений с целью обеспечения совместимости программных модулей;
- использование методов структурного проектирования и структурного программирования для распараллеливания процесса кодирования программного комплекса и сокращения времени кодирования.

Так как строго детерминированный диалог зачастую сводит на нет основные достоинства интерактивных процедур, разработаны средства, позволяющие изменять число проектных процедур, задаваемых директивами, выполнять переход к любому уровню диалога [74].

Сама структура диалога позволяет пользователю построить непротиворечивую модель поведения системы. Ясности поведения системы также способствуют информационные сообщения, соответствующие директивам диалога [37].

Структура информационных областей и методы доступа к ним в значительной степени определяют производительность средств преобразования ИТВ к РТ. При разработке логического и физического представления данных в информационных областях интерактивных систем особо остро встает проблема разрешения противоречий между необходимостью минимизации времени доступа к информационным областям и минимизацией занимаемого ими объема оперативной памяти. При разработке информационных областей системы эти противоречия были учтены [53].

Как и всякая программная система, интерактивные средства преобразования ИТВ в РТ должна удовлетворять таким требованиям, как

правильность, точность, совместимость, надежность, универсальность, защищенность, полезность, проверяемость и адаптируемость [75].

Исходными данными для разработки программного обеспечения системы являются:

- математическое обеспечение системы (глава 2, глава 3);
- информационные и управляющие кадры терминала.

Общие требования к программному обеспечению сложных программных систем, также как и средства их реализации, широко освещены в литературе, в частности [78, 79-83]. Они всесторонне учитывались при разработке ПО системы.

Анализ существующих средств преобразования информации табличного вида в файлы РТ

Во всех инструментальных СУБД предусмотрены средства импортирования данных. В частности, MS Access позволяет импортировать данные, представленные в форматах MS Access, HTML, MS Excel, текстовом формате и др. [84, 86]. Это существенно упрощает решение проблемы преобразования отдельных таблиц в файлы БД. В случае, если на основе имеющихся таблиц создается новая БД, то необходимо создать БД, импортировать в нее таблицы и, используя средства СУБД, построить необходимые связи между таблицами. Если в существующую БД добавляются новые таблицы, необходимо открыть БД, импортировать в нее таблицы и, используя средства СУБД, построить необходимые связи между таблицами [37].

Однако таблицы, построенные вне СУБД, как правило, нереляционные, а импортирование таблиц такого рода практически невозможно. Тем не менее инструментальные средства существующих СУБД могут быть полезны как при решений задач импорта, так и при преобразования ИТВ в РТ.

В системе Oracle предусмотрено специализированное средство для перемещения данных – SQL*Loader [66]. В нем используются файлы отвергнутых и некорректных записей – записей, которые по каким-либо причинам не могут быть включены в БД. Это, с одной стороны, говорит о том, что проблема преобразования информации табличного вида в файлы РТ актуальна, а с другой стороны, эта проблема не всегда имеет корректное решение.

В Microsoft SQL Server существует служба DTS (Data Transformation Services), предназначенная для преобразования данных. Структура DTS основана на библиотеке Microsoft OLE DB, которая представляет собой набор, предлагающий общий интерфейс различным типам источников данных. DTS в сочетании с OLE DB позволяет установить связь с любым источником данных, для которого имеется драйвер OLE DB или ODBC (Open Data Base Connectivity). [68, 69] Это – набор соглашений, которые позволяют получить доступ к информации из баз данных В SQL Server 2000 предусмотрены драйверы OLE DB для SQL Server, Oracle, Excel, Access, текстовых файлов ASCII-файлов и других источников данных, поддерживающих ODBC.

Однако и здесь есть недостатки. Служба DTS работает с нормализованными данными, поэтому требуются предварительные работы по подготовке данных с помощью средств SQL Server, но они не всегда приводят к положительному результату [68, 69].

Подводя итоги, можно сделать вывод, что средства импорта СУБД, а также другие механизмы обмена данными до определенной степени обеспечивают преобразование ИТВ в РТ, однако их зачастую бывает недостаточно.

Разрабатываемый комплекс программ предназначен для круга пользователей средней квалификации, с начальными знаниями в области БД. В связи с этим в качестве инструментальной СУБД оправдано выбрать доступную систему, обладающую малой субъективной сложностью и

удовлетворяющую необходимым требованиям. К такой инструментальной СУБД относится Microsoft Access.

MS Access включает в себя интегрированную среду объектно-ориентированного программирования Visual Basic for Applications (VBA), позволяющую реализовать любые программные решения.

В VBA БД рассматривается как совокупность объектов (таблиц, форм, отчетов, их элементов и т.д.). Эти объекты имеют свойства и методы, которые реализуют действия над объектами.

Среда VBA – одна из лучших сред систем программирования. Она объединяет наглядные инструменты: редактор VBA, окно разрабатываемого проекта, окно свойств объектов проекта, окно просмотра объектов, отладчик и др. Все инструменты унифицированы и являются общими для всех продуктов Microsoft Office [84, 86].

2. Программная реализация методики назначения ключевых полей

Может вызвать недоумение, что программная реализация назначения ключевых полей обсуждается до программной реализации преобразования ИТВ в РТ. Таким образом, специально подчеркивается, что методы по сути независимы и самодостаточны. Дело в том, что ключевые поля могут быть не назначены в таблицах существующих БД. И очень вероятно, что возникнет необходимость их назначить. В этом случае процедуры назначения ключевых полей могут оказаться полезными вне зависимости от процедур преобразования ИТВ в РТ.

2.1. Назначение первичных ключей на базе одного атрибута

Постановка задачи звучит предельно просто – необходимо в таблице найти домены, значения атрибутов которых уникальны.

В первую очередь напрашивается визуальный анализ всех столбцов таблицы. И это оправдано для таблиц с небольшими степенями и мощностями. Например, степень таблицы 10 столбцов, а ее мощность 50

строк. Опыт показывает, что таблицы большей размерности проанализировать “вручную” без ошибок очень сложно.

Во вторую очередь естественно попытаться использовать существующие инструментальные средства. Рассмотрим такие средства. В MS Acccess предусмотрен мастер просмотра повторяющихся записей. Он позволяет выбрать любое сочетание атрибутов таблицы и просмотреть, сколько значений связанных с выбранным атрибутом повторяются. Ниже на рис. 1 - 6 приведены шаги мастера для анализа двух полей “Имя” и “Отчество” таблицы “Сотрудники”.

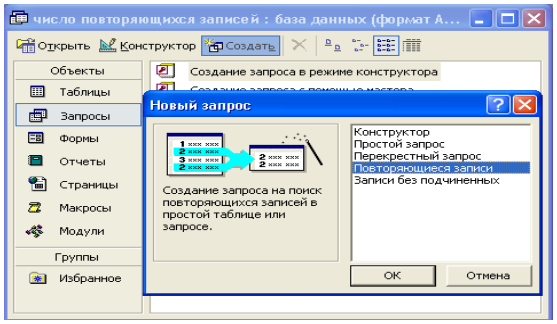


Рис. 1. Первый шаг мастера

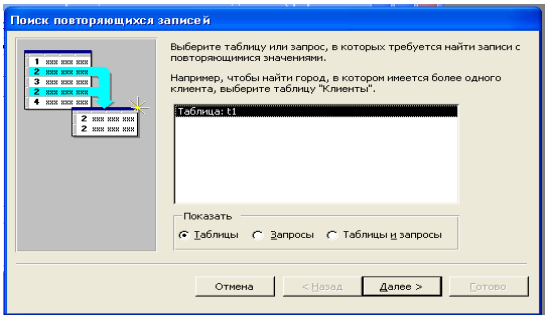


Рис. 2. Второй шаг мастера

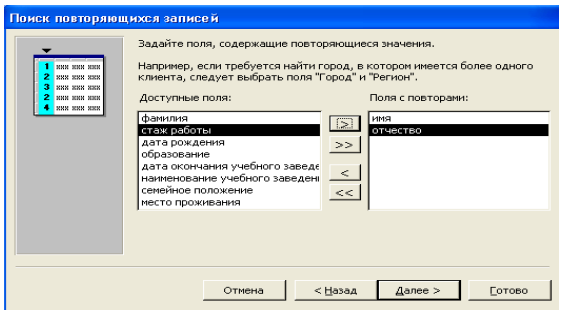


Рис. 3. Третий шаг мастера

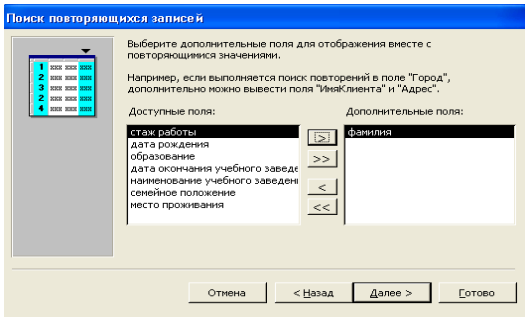


Рис. 4. Четвертый шаг мастера

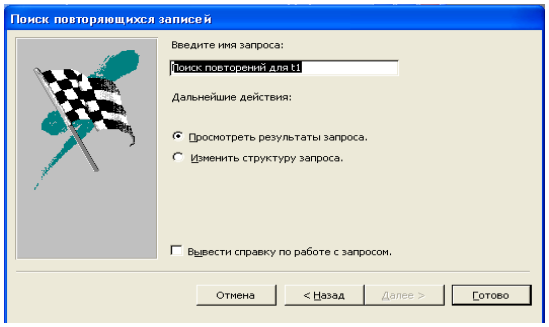


Рис. 5. Пятый шаг мастера

Рис. 6. Шестой шаг мастера

Первый шаг мастера позволяет выбрать мастер, второй – выбрать анализируемую таблицу, третий – выбрать атрибуты для анализа, четвертый

– выбрать отображаемые поля, пятый – указать имя запроса для отображения повторяющихся полей, шестой – выводит повторяющиеся поля. В нашем случае значения полей “Имя” и “Отчество” повторяются два раза. Таким образом, эти атрибуты не могут использоваться в качестве первичного ключа.

Следует обратить внимание на то, что эти 6 шагов нужно выполнять для каждого сочетания полей. В случае анализа одного атрибута для 30 столбцов манипуляции нужно выполнить 30 раз. Но для такой же таблицы при анализе двух атрибутов манипуляции придется проделать 435 раза, при анализе трех атрибутов – 4060 раз, при анализе четырех атрибутов – 27405 раза. Это связано с числом возможных сочетаний атрибутов.

В конечном случае все шаги мастера приводят к формированию следующего запроса на языке SQL.

SELECT [имя] FROM [Сотрудники] As Tmp GROUP BY [имя],[отчество] HAVING Count(*)>1 And [отчество] = [t1].[отчество])

Но написать несколько сотен запросов для каждой таблицы задача непростая.

В связи с этим реально использование существующих средств только при анализе одного атрибута. Конечно, удобнее получить исчерпывающую информацию за несколько секунд, чему и посвящены разработанные процедуры.

На рис. 7 приведен фрагмент таблицы, на которой будут продемонстрированы разработанные процедуры.

фамилия	имя	отчество	стаж работы	дата рождения	образование	дата окончания учебного заведения	наименование учебного
Агеев	Александр	Семенович	11	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Андреев	Алексей	Александрович	11	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Акылов	Антон	Алтуфьевич	10	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Брыкалов	Афиноген	Андреевич	10	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Борисов	Борис	Алексеевич	9	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Брейдо	Виктор	Абрамович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Витин	Иван	Антипович	11	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Волгин	Петр	Борисович	9	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Выдрин	Сидор	Викторович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана
Гусев	Владимир	Власович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баумана

Рис .7. Фрагмент анализируемой таблицы

В целях обеспечения возможности обработки таблицы и использования в процедурах типовых имен необходимо выполнить запрос на добавление (рис. 8).

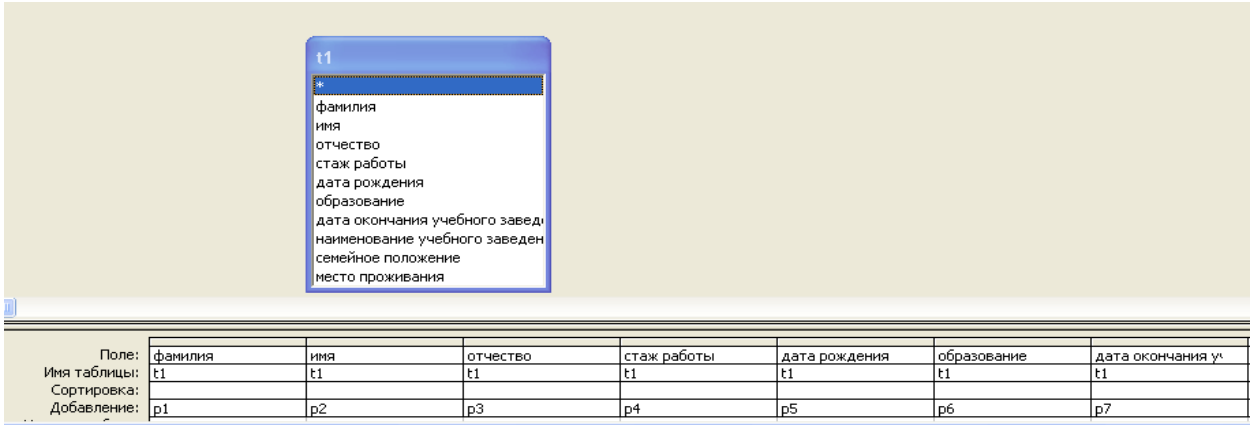


Рис. 8. Запрос на добавление исходной таблицы

В формате SQL запрос выглядит следующим образом:

INSERT INTO t2 (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10)
SELECT t1.фамилия, t1.имя, t1.отчество, t1.[стаж работы], t1.[дата рождения], t1.образование, t1.[дата окончания учебного заведения], t1.[наименование учебного заведения], t1.[семейное положение], t1.[место проживания]
FROM t1;

В результате выполнения запроса на добавлении сформируется рабочая таблица (рис. 9)

p1	p2	p3	p4	p5	p6	p7	p8
Агеев	Александр	Семенович	11	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Андреев	Алексей	Александрович	11	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Акылов	Антон	Алтуфьевич	10	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Брыкалов	Афиноген	Андреевич	10	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Борисов	Борис	Алексеевич	9	01.01.1980	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Брейдо	Виктор	Абрамович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Витин	Иван	Антипович	11	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Волгин	Петр	Борисович	9	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Выдрин	Сидор	Викторович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум
Гусев	Владимир	Власович	10	02.02.1978	высшее	07.07.1993	МГТУ им. Н.Э. Баум

Рис. 9. Рабочая таблица

После этого пользователю остается только воспользоваться интерфейсом представленном на рис. 10.

Рис. 10. Интерфейс пользователя для поиска первичных ключей

Ниже приведен фрагмент процедуры, обеспечивающей поиск первичного ключа на базе одного атрибута. Команды для удобства пояснений пронумерованы.

```
1. Private Sub Кнопка0_Click()
2. Dim mdb As Database  ' mdb объявляется как база данных
3. Dim rst As Recordset ' rst объявляется как массив записей
4. Dim a(10) As Variant ' Массив заголовков исходной таблицы
5. Dim kp As String
6. Dim np As String
7. Set mdb = CurrentDb  ' mdb назначается текущая база данных
8. Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из
   таблицы t1
9. DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается
   форма t1 для таблицы t1
10.DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
11.kp = ""
12.np = ""
13.n = 0 ' счетчик имен столбцов исходной таблицы
14.For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для
   их запоминания
15.' MsgBox fld.Name
16.n = n + 1
17.a(n) = fld.Name
18.Next fld
19.' DoCmd.Close ' закрывается форма t1
20.DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
21.DoCmd.GoToRecord acForm, "t2", acLast
22.DoCmd.GoToRecord , , acFirst
23.c = Forms!t2!cpol ' запоминается число полей
24.f = 0
25.For I = 1 To c - 1
```

```
26.p11 = Forms!t2!p1
27.'MsgBox (p11)
28.For j = I + 1 To c
29.DoCmd.GoToRecord , , acNext
30.p22 = Forms!t2!p1
31.'MsgBox (p22)
32.If p11 = p22 Then
33.f = 1
34.If f = 1 Then
35.'   MsgBox a(1)
36.MsgBox ("столбец не ключевой: " & a(1))
37.np = np & ", " & a(1)
38.End If
39.Exit For
40.End If
41.Next j
42.If f = 1 Then Exit For
43.DoCmd.GoToRecord , , acFirst
44.For m = 1 To I
45.DoCmd.GoToRecord , , acNext
46.Next m
47.If f = 1 Then
48.'   MsgBox a(1)
49.MsgBox ("столбец не ключевой: " & a(1))
50.np = np & ", " & a(1)
51.End If
52.Next I
53.If f = 0 Then
54.'    MsgBox a(1)
55.MsgBox ("столбец ключевой: " & a(1))
56.kp = kp & ", " & a(1)
57.End If
58.'MsgBox (" НЕ КЛЮЧЕВЫЕ ПОЛЯ: " & np & "  КЛЮЧЕВЫЕ ПОЛЯ: "
   & kp)
59.MsgBox ("ПОЛЯ, ПРЕТЕНДУЮЩИЕ НА ПЕРВИЧНЫЙ КЛЮЧ: " &
   kp)
60.DoCmd.Close acForm, "t1"
61.End Sub
```

Дадим краткое пояснение основной компоненты при процедуре назначения первичных ключей на базе одного атрибута.

В командах 5, 6 объявляются строковые переменные, в которых накапливаются сообщения о ключевых и неключевых полях. В Заголовков

таблицы запоминаются заголовки таблицы. Они впоследствии потребуются при формировании сообщений о первичных ключах.

Посредством оператора 20 открывается рабочая форма. С помощью операторов 21, 22 она “перетряхивается”.

В форме “t2” имеется поле “cpol”, где подсчитывается число записей таблицы. Это поле используется для организации цикла просмотра всех записей, а в их рамках – полей, соответствующих анализируемому столбцу.

В 24 операторе обнуляется флажок, который фиксирует отсутствие ключевых полей.

Оператор 25 организует цикл для сравнения каждого значения атрибута со всеми остальными. Все остальные значения перебираются посредством цикла, который начинается 28-м оператором.

В операторах 26 и 29 значениям переменных присваиваются значения полей таблицы, которые сравниваются в операторе 32.

С помощью оператора 37 накапливаются имена неключевых атрибутов. Операторы 43 – 46 позволяют перейти (вернуться) к следующей сравниваемой записи, ведь в цикле по “j” мы достигаем конца таблицы.

Оператор 59 формирует итоговое сообщение со списком полей, которые могут претендовать на ключевые поля.

Кратко рассмотрим пример функционирования полного комплекта процедур выявления первичного ключа на базе одного атрибута после того, как пользователь нажмет соответствующую кнопку (рис. 10).

На рис. 11 приведен пример одного из текущих сообщений процедуры.

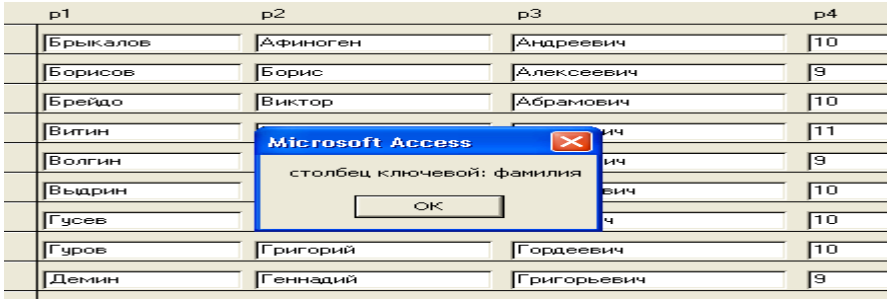


Рис. 11. Пример одного из текущих сообщений процедуры.

На рис. 12 приведен пример другого типа текущих сообщений процедуры.

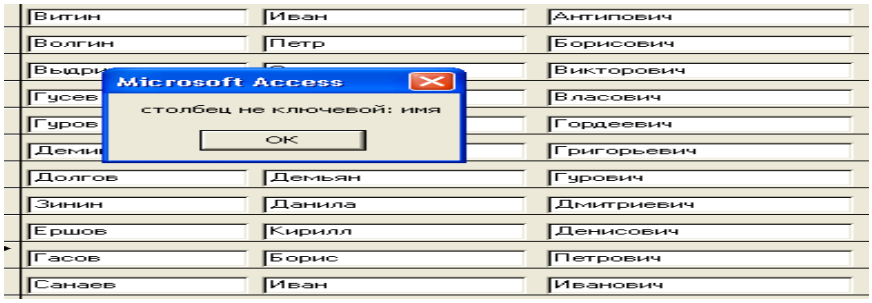


Рис. 12. Пример другого типа текущих сообщений процедуры.

На рис. 13 приведено итоговое сообщение процедуры о полях таблицы, которые могут претендовать на роль первичного ключа.

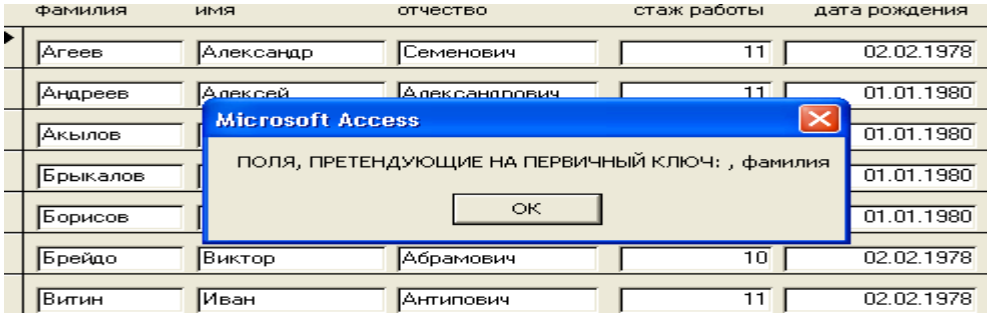


Рис. 13. Итоговое сообщение процедуру о полях таблицы, которые могут претендовать на роль первичного ключа.

2.2 Назначение первичных ключей на базе двух атрибутов

Ниже приведен фрагмент процедуры, обеспечивающей поиск первичного ключа на базе двух атрибутов. Команды для удобства пояснений пронумерованы.

```
1. Private Sub Кнопка2_Click()
2. Dim mdb As Database  ' mdb объявляется как база данных
3. Dim rst As Recordset ' rst объявляется как массив записей
4. Dim a(10) As Variant ' Массив заголовков исходной таблицы
5. Dim kp As String
6. Dim np As String
7. Set mdb = CurrentDb  ' mdb назначается текущая база данных
8. Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из
   таблицы t1
9. DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается
   форма t1 для таблицы t1
10.DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
11.n = 0 ' счетчик имен столбцов исходной таблицы
12.For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для их
   запоминания
13.' MsgBox fld.Name
14.n = n + 1
```

```
15.a(n) = fld.Name
16.Next fld
17.' DoCmd.Close ' закрывается форма t1
18.DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
19.DoCmd.GoToRecord acForm, "t2", acLast ' перетряхивание
20.DoCmd.GoToRecord , , acFirst          ' формы
21.c = Forms!t2!cpol ' запоминается число полей
22.kp = ""
23.np = ""
24.f = 0
25.For I = 1 To c - 1
26.p11 = Forms!t2!p1 & " " & Forms!t2!p2
27.'MsgBox (p11)
28.For j = I + 1 To c
29.DoCmd.GoToRecord , , acNext
30.p22 = Forms!t2!p1 & " " & Forms!t2!p2
31.'MsgBox (p22)
32.If p11 = p22 Then
33.f = 1
34.If f = 1 Then
35.'MsgBox a(1) & " " & a(2)
36.MsgBox ("столбцы не могут составить ключевой столбец: " & a(1)) & " "
   & a(2)
37.np = np & a(1) & " и " & a(2) & ", "
38.End If
39.Exit For
40.End If
41.Next j
42.If f = 1 Then Exit For
43.DoCmd.GoToRecord , , acFirst
44.For m = 1 To I
45.DoCmd.GoToRecord , , acNext
46.Next m
47.If f = 1 Then
48.'MsgBox a(1) & " " & a(2)
49.MsgBox ("столбцы не могут составить ключевой столбец: " & a(1)) & " "
   & a(2)
50.np = np & a(1) & " и " & a(2) & ", "
51.End If
52.Next I
53.If f = 0 Then
54.'MsgBox a(1) & " " & a(2)
55.MsgBox ("столбцы  могут составить ключевой столбец: " & a(1)) & " " &
   a(2)
```

```
56.kp = kp & a(1) & " и " & a(2) & ", "
57.End If
58.DoCmd.Close acForm, "t2"
59.MsgBox ("ПОЛЯ, ПРЕТЕНДУЮЩИЕ НА ПЕРВИЧНЫЙ КЛЮЧ: " & kp)
60.End Sub
```

По структуре и составу операторов этот фрагмент похож на предыдущий фрагмент. Принципиальным отличием является то, что анализируется пара полей, а не одно поле. Это отражено в операторах 26, 30, 35– 37, 48– 50, 54, 56.

Промежуточные сообщения могут быть полезны, но в принципе их можно убрать. При проверке конкатенации нескольких атрибутов на принадлежность первичному ключу их может быть очень много – сочетание атрибутов велико.

На рис. 14 приведено итоговое сообщение процедуры о парах полей таблицы, которые могут претендовать на роль первичного ключа. Промежуточные сообщения в данном случае опущены.

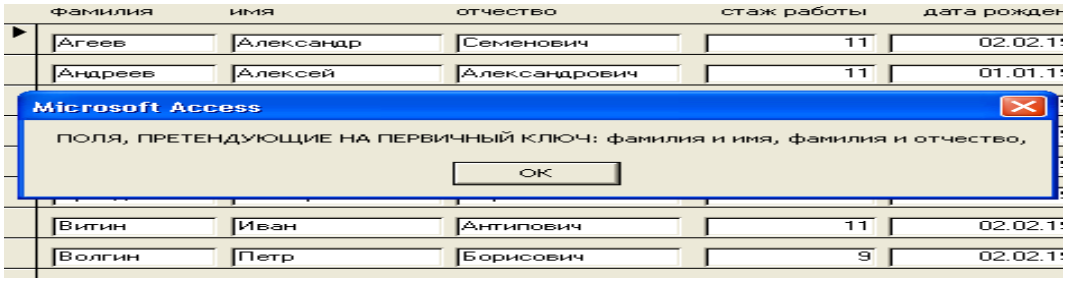


Рис. 14. Итоговое сообщение процедуру о парах полей таблицы

2.3. Назначение первичных ключей на базе трех атрибутов

Ниже приведен фрагмент процедуры, обеспечивающей поиск первичного ключа на базе трех атрибутов. Команды для удобства пояснений пронумерованы.

```
1. Private Sub Кнопка5_Click()
2. Dim mdb As Database  ' mdb объявляется как база данных
3. Dim rst As Recordset ' rst объявляется как массив записей
4. Dim a(10) As Variant ' Массив заголовков исходной таблицы
5. Dim kp As String
6. Dim np As String
```

```
Set mdb = CurrentDb  ' mdb назначается текущая база данных
Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из таблицы t1
DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается форма t1 для таблицы t1
DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
n = 0 ' счетчик имен столбцов исходной таблицы
For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для их запоминания
' MsgBox fld.Name
n = n + 1
a(n) = fld.Name
Next fld
' DoCmd.Close ' закрывается форма t1

DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
DoCmd.GoToRecord acForm, "t2", acLast ' перетряхивание
DoCmd.GoToRecord , , acFirst          ' формы
c = Forms!t2!cpol ' запоминается число полей
kp = ""
np = ""
f = 0
For I = 1 To c - 1
p11 = Forms!t2!p1 & " " & Forms!t2!p2 & " " & Forms!t2!p3
'MsgBox (p11)
For j = I + 1 To c
DoCmd.GoToRecord , , acNext
p22 = Forms!t2!p1 & " " & Forms!t2!p2 & " " & Forms!t2!p3
'MsgBox (p22)
If p11 = p22 Then
f = 1
If f = 1 Then
'MsgBox a(1) & " " & a(2)& " " & a(3)
MsgBox ("столбцы не могут составить ключевой столбец: " & a(1)) & " " & a(2) & " " & a(3)
np = np & a(1) & " " & a(2) & " " & "  " & a(3) & ", "
End If
Exit For
End If
Next j
If f = 1 Then Exit For
DoCmd.GoToRecord , , acFirst
For m = 1 To I
DoCmd.GoToRecord , , acNext
```

```
46.Next m
47.If f = 1 Then
48.'MsgBox a(1) & " " & a(2) & " " & a(3)
49.MsgBox ("столбцы не могут составить ключевой столбец: " & a(1)) & "
    " & a(2) & " " & a(3)
50.np = np & a(1) & " " & a(2) & " " & " " & a(3) & ", "
51.End If
52.Next I
53.If f = 0 Then
54.'MsgBox a(1) & " " & a(2) & " " & a(3)
55.MsgBox ("столбцы  могут составить ключевой столбец: " & a(1)) & " "
    & a(2) & " " & a(3)
56.kp = kp & a(1) & " " & a(2) & " " & " " & a(3) & ", "
57.End If
58.DoCmd.Close acForm, "t2"
59.'MsgBox (" НЕ КЛЮЧЕВЫЕ ПОЛЯ: " & np & "  КЛЮЧЕВЫЕ ПОЛЯ: "
    & kp)
60.MsgBox ("ПОЛЯ, ПРЕТЕНДУЮЩИЕ НА ПЕРВИЧНЫЙ КЛЮЧ: " &
    kp)
61.DoCmd.Close acForm, "t1"
62.End Sub
```

По структуре и составу операторов этот фрагмент похож на предыдущий фрагмент. Принципиальным отличием является то, что анализируется пара полей, а не одно поле. Это отражено в операторах 26, 30, 35– 37, 48– 50, 54, 56.

На рис. 15 приведено одно из промежуточных сообщений процедуры.

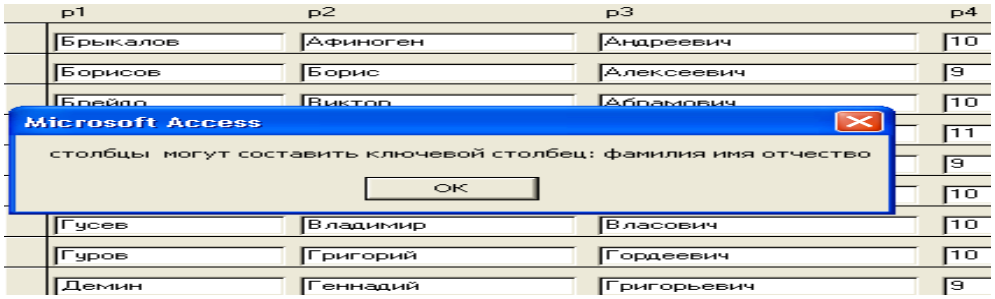


Рис .15. Одно из промежуточных сообщений процедуры.

На рис. 16 приведено итоговое сообщение процедуры о тройках полей таблицы, которые могут претендовать на роль первичного ключа.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10

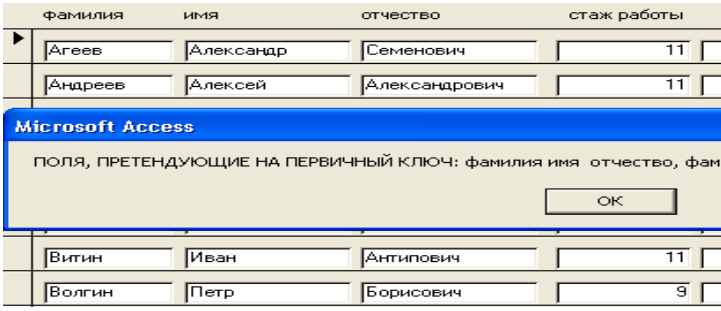


Рис. 16. Итоговое сообщение процедуры о тройках полей таблицы, которые могут претендовать на роль первичного ключа.

2.4. Оценка вычислительной сложности процедур назначения первичных ключей

Выполним аналитическую оценку вычислительной сложности разработанных программ.

В качестве базовой для аналитической оценки вычислительной сложности процедур операции выбрана операция сравнения. Во-первых, эта операция одна из самых длительных, а во-вторых, она встречается внутри всех циклов. Остальные компоненты программы добавляют в общее время выполнения программы незначительные значения.

Важной компонентой оценки является число сочетаний. Ведь для анализируемой таблицы необходимо проверить все сочетания полей.

Число возможных сочетаний C_n^k = n!/(n-k)!/k!, где n – общее число атрибутов таблицы, а k – количество проверяемых атрибутов на принадлежность к первичному ключу.

Для предварительного анализа написана небольшая процедура, которая позволила подсчитать количества сочетания для различных n и k. Результаты сведены в таблицы. Таблица 1– число сочетаний из 1– 10 по 1, 2, 3, 4. Таблица 2 – число сочетаний из 11– 20 по 1, 2, 3, 4. Таблица 3 - Число сочетаний из 21– 30 по 1, 2, 3, 4.

Таблица 1

2	0	1	3	6	10	15	21	28	36	45
3	0	0	1	4	10	20	35	56	84	120
4	0	0	0	1	5	15	35	70	126	210

Таблица 2

	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	11	12	13	14	15	15	17	18	19	20
2	55	66	78	91	105	120	136	153	171	190
3	165	220	286	364	455	560	680	816	969	1140
4	330	495	715	1001	1365	1820	2380	3060	3876	4845

Таблица 3

	21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
1	21	22	23	24	25	26	27	28	29	30
2	210	231	253	276	300	325	351	378	406	435
3	1330	1540	1771	2024	2300	2600	2925	3276	3654	4060
4	5985	7315	8855	10626	12650	14950	17550	20475	23751	27405

Важно отметить, что сравнивать необходимо значения полей всех записей таблицы, при этом необходимо сравнивать поля каждой записи со всеми остальными. Поэтому к числу сочетаний добавляется сомножитель. Определим его эмпирическим методом.

Например, возьмем таблицу из одиннадцати записей. Тогда для записи 1-й потребуется 10 проверок, для 2-й – 9, для 3-й – 8, для 4-й – 7, для 5-й – 6, для 6-й – 5, для 7-й – 4, для 8-й – 3, для 9-й – 2, для 10-й – 1. Это следует из фрагмента программы – каждая текущая запись сравнивается со всеми остальными. И с увеличением номера текущей записи уменьшается число записей, с которыми ее необходимо сравнить. Таким образом, общее количество сравнений будет 10+9+8+7+6+5+4+3+2+1 = 10+1 + 9+2 + 8+3 + 7+4 + 6+5 = 55. Итого мы получили 5 пар или (11- 1)/2 или (m-1)/2, где m – число записей таблицы. В каждой паре сумма равна 11 или m. Таким образом, число сравнений будет равным m(m-1)/2. Получено аналитическое

выражение для оценки временной сложности алгоритма: VS = m(m-1)/2n!/(n-k)!/k!, где

n – общее число атрибутов таблицы;

k – количество проверяемых атрибутов на принадлежность к первичному ключу;

m – число записей таблицы.

Оценка VS впечатляет и внушает пессимизм. Для дальнейшего анализа вычислительной сложности программ построены графики (рис. 17 и рис. 18).

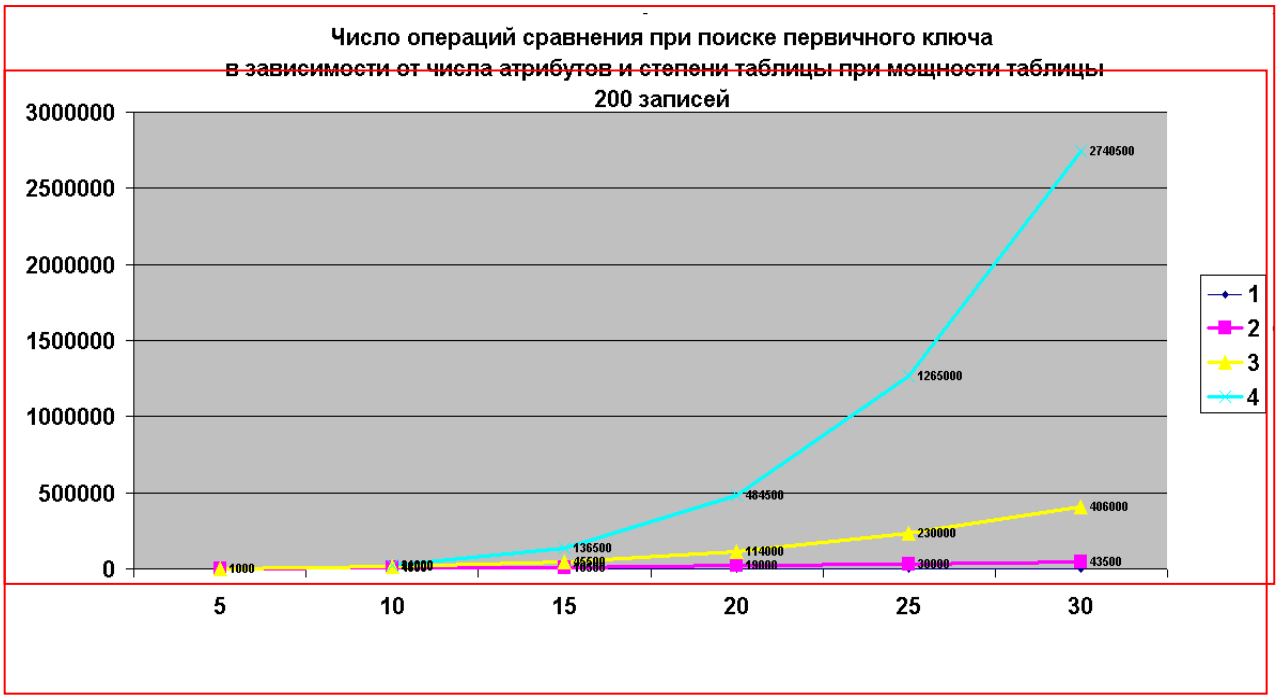


Рис.17. График изменения числа сравнений для таблицы из 200 записей.

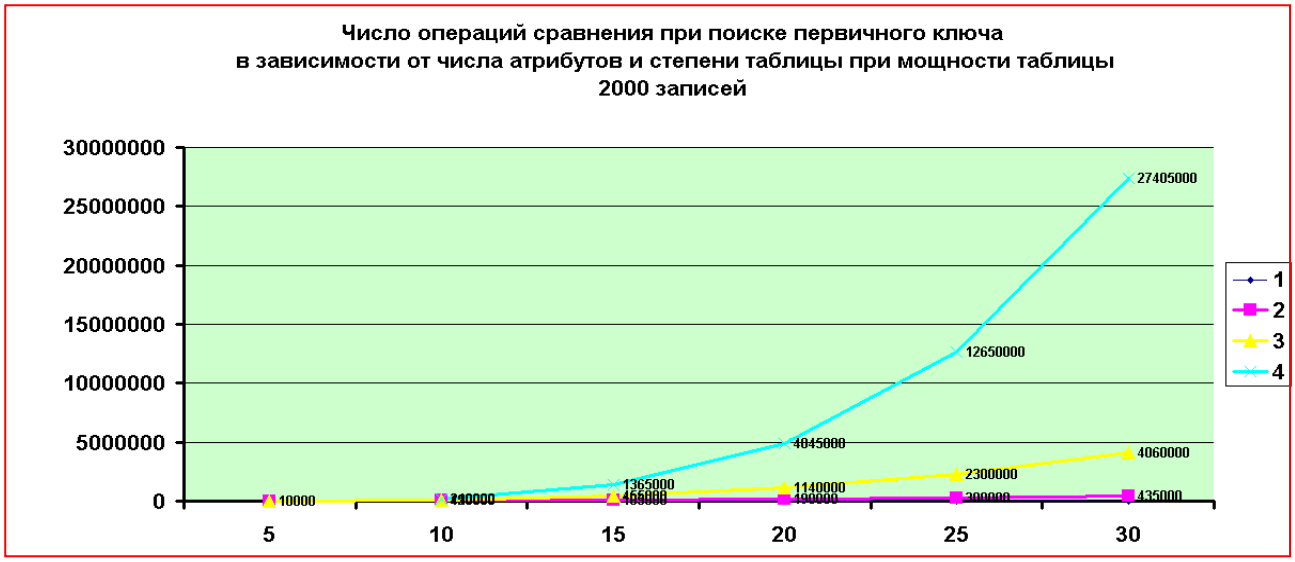


Рис.18. График изменения числа сравнений для таблицы из 2000 записей.

Самое большое число, которое можно увидеть на графиках это 27.450.000. В качестве утешения можно сказать, что такая ситуация встречается исключительно редко. Кроме того, специальный эксперимент показал, что 100.000.000 сравнений на среднем компьютере (1,6 ГГц, 2 ядра) выполняется за 7 секунд. В качестве доказательства приводится код программы и результат ее выполнения (рис. 5.19).

```
Private Sub Кнопка4_Click()
Forms!даты!дата1 = Time
For i = 1 To 100000000
    If i = 1000000 Then r = 0
Next i
Forms!даты!дата2 = Time
End Sub
```

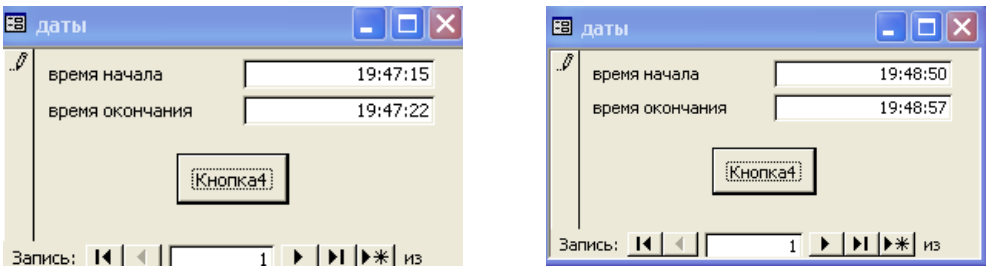


Рис.19. Результаты измерения времени 100.000.000 сравнений

Выполним экспериментальную оценку вычислительной сложности разработанных программ.

Посредством экспериментов выполнены оценки времен выполнения процедур обработки 2000 записей при количестве анализируемых атрибутов 1, 2, 3, 4 и общем числе атрибутов 5, 10, 15, 20, 25, 30, которые сведены в таблицу 4.

Таблица 4

	5	**10**	**15**	**20**	**25**	**30**
1	5000	10000	15000	20000	25000	30000
2	10000	45000	105000	190000	300000	435000
3	100000	120000	455000	1140000	2300000	4060000
4	5000	210000	1365000	4845000	12650000	27405000

Таблица 4 взята за основу для построения, графиков, приведенных на рис. 18.

Рис.18. Графики времени выполнения процедур в зависимости от числа столбцов и числа анализируемых атрибутов.

Как видно из графика, времена вполне приемлемы. При этом характеристики таблиц и числа атрибутов, как показывает опыт разработок, исключительно редко превышают рассмотренные значения.

Однако бывают и исключения. В связи с этим далее анализируются границы применимости предлагаемого способа и соответствующих процедур.

Подсчитаны числа сочетаний для 100 столбцов таблицы:

Из 100 по 2 – 4950

Из 100 по 3 – 161600

Из 100 по 4 – 3921225

Число необходимых итераций для анализа 2000 записей равно

1999*2000/2= 1999000

Число необходимых операций сравнений для 100 столбцов и 3-х анализируемых атрибутов для таблицы в 2000 записей равно:

161600*1999000 = 9.905.045.000

Экспериментальная проверка показала, что для анализа таблицы из 100 столбцов и 2000 записей проверка всех сочетаний по 3-и атрибута, претендующих в своем составе на первичный ключ, необходимо 4 мин. 45 сек. Это вполне приемлемое время, учитывая то, что выполнить миллиарды сравнений вручную просто невозможно.

Для анализа таблицы из 100 столбцов и 2000 записей проверка всех сочетаний по 4-и атрибута, претендующих в своем составе на первичный ключ, необходимо 153 мин. 15 сек. Такая ситуация скорее всего гипотетическая. В основном из-за того, что для назначения первичного ключа очень редко используют 4-е атрибута. Но, как известно, в целях решения важных задач зачастую идут и на такие затраты.

Таким образом, можно сделать вывод, что предложенный способ и соответствующие процедуры могут успешно использоваться для анализа таблиц из несколько десятков столбцов, нескольких тысяч записей и одного, двух , трех атрибутов, используемых в качестве ключевых.

3. Программная реализация метода преобразования ИТВ в РТ

3.1 Анализ типов атрибутов таблицы

В ИТВ типы элементов в пределах любого столбца таблицы могут различаться. Такие столбцы недопустимо использовать в РТ БД. Ниже описаны программные средства, которые позволяют разработчику выявить наиболее вероятный тип для каждого столбца ИТВ и принять правильное решение.

Все столбцы ИТВ имеют тип строковый. Поэтому задача состоит в том, чтобы проанализировать в каждом столбце все его элементы и по внешнему виду элементов определить его тип. Предложен следующий подход.

Определяется не тип элементов, а наоборот непринадлежность к типу. Формируются данные о том, сколько и какие элементы столбца не соответствуют каждому из базовых типов. После этого нетрудно сделать вывод о том, какого типа столбец. Могут создаться неочевидные ситуации. Тогда решение принимает разработчик, после чего вид элементов он приводит в соответствии с выбранным типом, а затем назначает выбранный тип в целевой таблице. При этом рассматриваются базовые типы: дата-время, числовой и строковый.

Сначала кратко рассмотрим, что сделано по этому поводу, а потом – как это сделано.

На рис. 20 приведен фрагмент интерфейса для анализа типов ИТВ.

Рис. 20. Фрагмент интерфейса для анализа типов ИТВ.

После нажатия на соответствующую кнопку сформируется сообщение о том, сколько полей не являются датами для анализируемого столбца. В данном случае это “фамилия” (рис. 21).

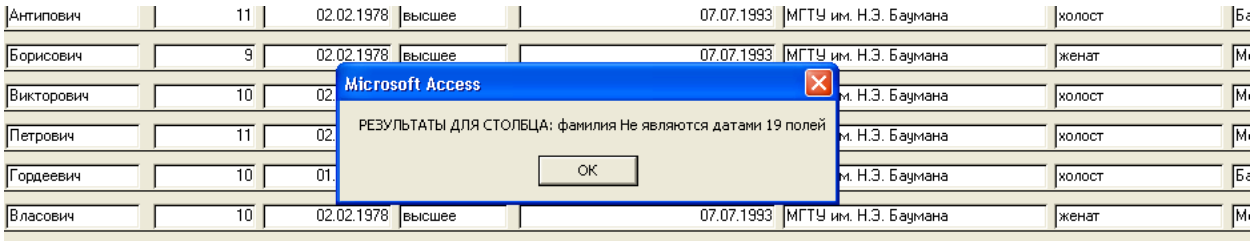


Рис. 21. Сообщение о том, сколько полей не являются датами.

Затем сформируется сообщение о том, в каких записях фамилия не является датой. Это показано на рис. 22.

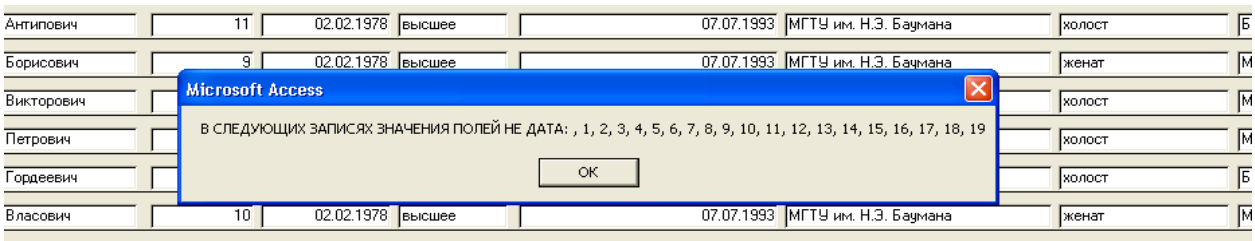


Рис. 22. Сообщение о том, в каких записях фамилия не является датой.

Далее сформируется сообщение о том, сколько полей не являются числами для анализируемого столбца (рис. 23).

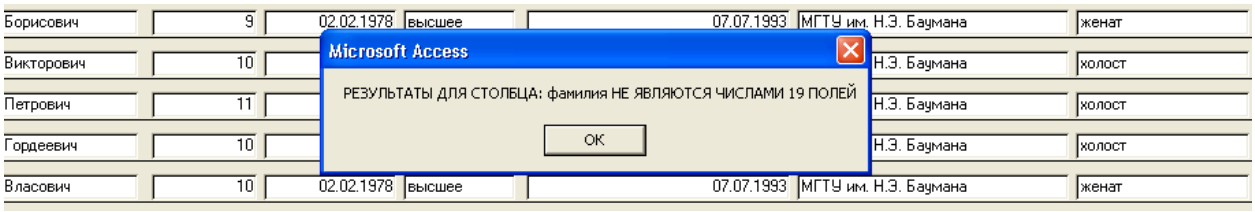


Рис. 23. Сообщение о том, сколько полей не являются числами для анализируемого столбца.

Затем сформируется сообщение о том, в каких записях фамилия не является датой. Это показано на рис. 24.

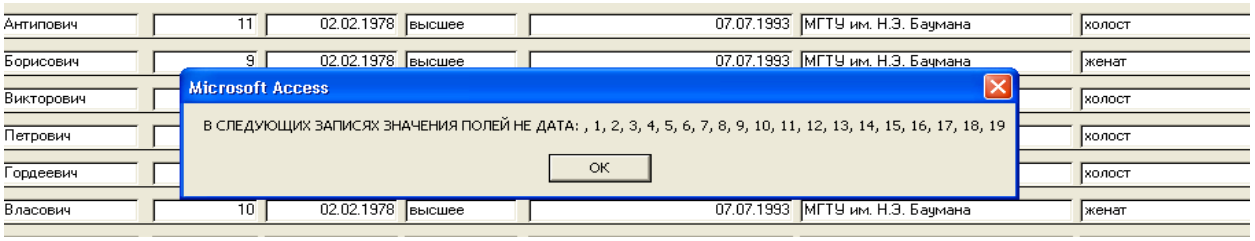


Рис . 24. Сообщение о том, в каких записях фамилия не является датой.

Учитывая то, что в таблице всего 19 полей, и все они не являются ни датами ни числами, легко сделать вывод о том, что столбец “фамилия” строкового типа.

Рассмотрим противоположный случай. Анализируется 5-й столбец с датами. В процессе анализа этого столбца сформируется сообщение о том, сколько полей не являются датами для анализируемого столбца (рис. 25).

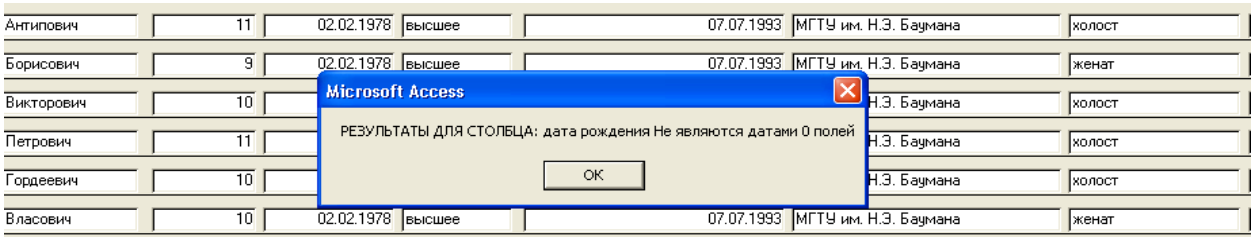


Рис. 25. Сообщение о том, сколько полей не являются датами для анализируемого столбца.

Затем сформируется сообщение о том, в каких записях фамилия не является датой. Это показано на рис. 26.

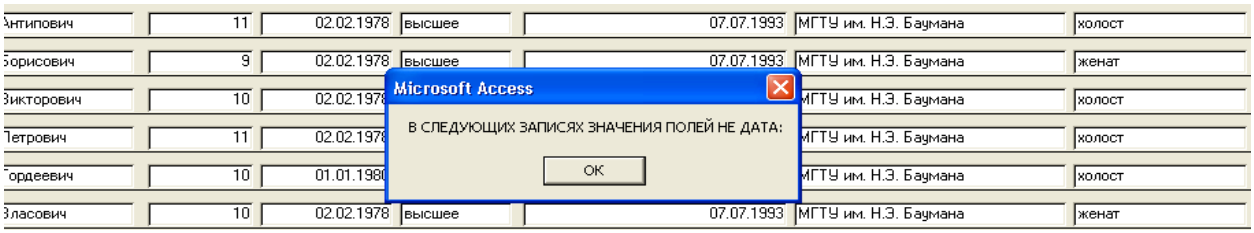


Рис. 26. Сообщение о том, в каких записях поля не являются датой.

Легко сделать вывод о том, что столбец “дата рождения” имеет тип дата.

Рассмотрим ситуацию менее очевидную. Например, в столбце “дата рождения” в некоторых полях указана информация следующего вида: 13.10.55 года. Строго говоря, такая запись не удовлетворяет типу “дата”. При анализе такой ситуации сформируется сообщение, представленное на рис. 27.

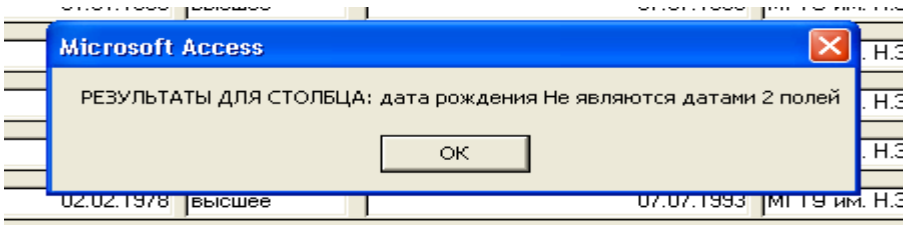


Рис. 27. Сообщение о несоответствии дате 2-х полей

Далее сформируется сообщение (рис. 28).

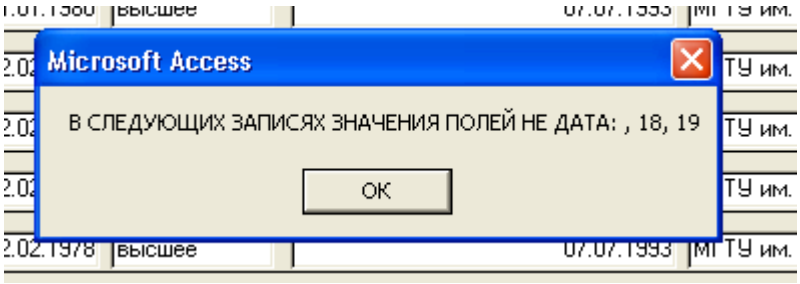


Рис. 28. Сообщение о номерах записей, где поле “дата рождения” не соответствует типу “дата”

В соответствии с полученными сообщениями разработчик может принять решение о том, что поля в записях 18 и 19 необходимо исправить.

Событию “нажатие кнопки” “Анализ типов элементов столбцов” поставлена в соответствие процедура, фрагмент которой приведен ниже. Для удобства разъяснения основных приемов операторы пронумерованы.

```
1.  Private Sub Кнопка0_Click()
2.  On Error GoTo nd
3.  cnd = 0
4.  Dim mdb As Database  ' mdb объявляется как база данных
5.  Dim rst As Recordset ' rst объявляется как массив записей
6.  Dim a(10) As Variant ' Массив заголовков исходной таблицы
7.  Dim kp As String
8.  Dim np As String
9.  Set mdb = CurrentDb  ' mdb назначается текущая база данных
10. Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из
    таблицы t1
11. DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается
    форма t1 для таблицы t1
12. DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
13. kp = ""
14. np = ""
15. n = 0 ' счетчик имен столбцов исходной таблицы
16. For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для
    их запоминания
17. ' MsgBox fld.Name
18. n = n + 1
19. a(n) = fld.Name
20. Next fld
21. ' DoCmd.Close ' закрывается форма t1
22. DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
23. DoCmd.GoToRecord acForm, "t2", acLast
24. DoCmd.GoToRecord , , acFirst
```

```
25.c = Forms!t2!cpol ' запоминается число полей
26.cnd = 0
27.For i = 1 To c
28.p11 = Forms!t2!p1
29.v = FormatDateTime(p11)
30.GoTo d
31.nd: cnd = cnd + 1
32.np = np & ", " & i
33.Resume Next
34.GoTo d
35.d: DoCmd.GoToRecord , , acNext
36.Next i
37.DoCmd.Close acForm, "t2"
38.MsgBox ("РЕЗУЛЬТАТЫ ДЛЯ СТОЛБЦА: " & a(1) & " Не являются
   датами " & cnd & " полей")
39.MsgBox ("В СЛЕДУЮЩИХ ЗАПИСЯХ ЗНАЧЕНИЯ ПОЛЕЙ НЕ
   ДАТА: " & np)
40.DoCmd.Close acForm, "t1"
41.funnnunber (0)
42.End Sub
```

Большинство операторов совпадает с операторами процедуры, рассмотренной в предыдущем параграфе. Рассматриваются принципиальные отличия.

Все построено на выявлении ошибок преобразования типа, фиксирования количества этих ошибок, номеров строк и имен столбцов. Для этого используется оператор 2 (On Error GoTo nd). Ошибка преобразования может возникнуть в операторе 29 v = FormatDateTime(p11). Здесь осуществляется попытка преобразования типа текущего поля в тип “дата”. Если преобразование невозможно, то возникает ошибка и переход на метку nd (см. предыдущий оператор). В операторе 31, помеченным этой меткой, увеличивается счетчик недат. В операторе 32 запоминается номер строки, в которой анализируемое поле не является дата. В операторе 33 выполняется сброс ошибки. В операторе 34 выполняется переход к дальнейшей обработке записей. Выбирается следующая запись и начинается следующая итерация цикла перебора записей.

Если ошибки преобразования не возникло после выполнения оператора 29, то осуществляется переход к дальнейшей обработке записей. Выбирается следующая запись и начинается следующая итерация цикла перебора записей.

Таким образом, в результате работы процедуры будут выявлены все поля анализируемого столбца, которые не являются датой и сохранятся номера соответствующих записей.

После окончания цикла закрывается вспомогательная таблица “t2” и выводятся сообщения о числе полей, не являющимися датами и сообщение о номерах соответствующих строк.

Но поля проверены только на соответствие типу “дата”. Проверку на соответствие типу “число” в данном контексте выполнить невозможно, т.к. неизвестно по какому поводу возникла ситуация On Error.

Для выхода из этой ситуации написана функция, которая вызывается в операторе 42 (funnnunber (0)).

Ниже представлена эта функция.

```
1. Private Function funnnunber(r) As Integer
2. On Error GoTo nd
3. cnd = 0
4. Dim mdb As Database  ' mdb объявляется как база данных
5. Dim rst As Recordset ' rst объявляется как массив записей
6. Dim a(10) As Variant ' Массив заголовков исходной таблицы
7. Dim kp As String
8. Dim np As String
9. Set mdb = CurrentDb  ' mdb назначается текущая база данных
10.Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из
   таблицы t1
11.DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается
   форма t1 для таблицы t1
12.DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
13.kp = ""
14.np = ""
15.n = 0 ' счетчик имен столбцов исходной таблицы
16.For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для
   их запоминания
17.' MsgBox fld.Name
```

```
18.n = n + 1
19.a(n) = fld.Name
20.Next fld
21.' DoCmd.Close ' закрывается форма t1
22.DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
23.DoCmd.GoToRecord acForm, "t2", acLast
24.DoCmd.GoToRecord , , acFirst
25.c = Forms!t2!cpol ' запоминается число полей
26.cnd = 0
27.For i = 1 To c
28.p11 = Forms!t2!p1
29.v = FormatNumber(p11)
30.GoTo d
31.nd: cnd = cnd + 1
32.np = np & ", " & i
33.Resume Next
34.GoTo d
35.d: DoCmd.GoToRecord , , acNext
36.Next i
43.DoCmd.Close acForm, "t2"
44.MsgBox ("РЕЗУЛЬТАТЫ ДЛЯ СТОЛБЦА: " & a(1) & " НЕ
   ЯВЛЯЮТСЯ ЧИСЛАМИ " & cnd & " ПОЛЕЙ")
45.MsgBox ("В СЛЕДУЮЩИХ ЗАПИСЯХ ЗНАЧЕНИЯ ПОЛЕЙ НЕ
   ЧИСЛА: " & np)
46.DoCmd.Close acForm, "t1"
47.End Function
```

По сути, эта функция практически не отличается от вызывающей процедуры. Главное отличие в операторе 29 (v = FormatNumber(p11)). В этом операторе осуществляется попытка преобразования текущего поля в число. Далее работа программы выполняется аналогично. Конечно, в конце функции формируются соответствующие ей сообщения.

3.2. Избавление от дублирующихся записей

Избавление от дублирующихся записей является несложной процедурой. Для ее реализации необходимо 3-и запроса: запрос на создание новой таблицы на базе исходной, в которой отсутствуют повторяющиеся записи; запрос на очистку исходной таблицы; запрос на добавление таблицы, в которой отсутствуют повторяющиеся записи в исходную таблицу.

Продемонстрируем разработки, выполненные в этом направлении.

На рис. 29 продемонстрирована таблица, в которой 3-и последние записи повторяются.

	Выдрин	Сидор	Викторович	10	02.02.1978
	Гасов	Борис	Петрович	11	02.02.1978
	Гуров	Григорий	Гордеевич	10	01.01.1980
	Гусев	Владимир	Власович	10	02.02.1978
	Демин	Геннадий	Григорьевич	9	02.02.1978
	Долгов	Демьян	Гурович	10	02.02.1978
	Ершов	Кирилл	Денисович	10	02.02.1978
	Зинин	Данила	Дмитриевич	9	02.02.1978
	Санаев	Иван	Иванович	11	01.01.1980
	Сидоров	Петр	Сергеевич	12	01.01.1979
	Фомин	Сидор	Михайлович	11	02.02.1978
	Фомин	Сидор	Михайлович	11	02.02.1978
	Фомин	Сидор	Михайлович	11	02.02.1978

Рис. 29. Таблица, в которой 3-и последние записи повторяются.

Далее на рис. 30 приведен фрагмент интерфейса для исключения дублирования записей.

Рис . 30. Фрагмент интерфейса для исключения дублирования записей.

После нажатия кнопки “Исключение дублирования записей” сформируется уведомляющее сообщение (рисунок 31).

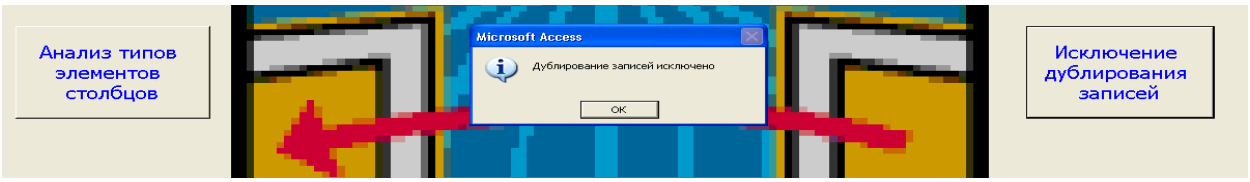


Рис . 31. Уведомляющее сообщение.

После выполнения устранения дублирования исходная таблица примет вид (рис. 32).

	Выдрин	Сидор	Викторович	10	02.02.1978
	Гасов	Борис	Петрович	11	02.02.1978
	Гуров	Григорий	Гордеевич	10	01.01.1980
	Гусев	Владимир	Власович	10	02.02.1978
	Демин	Геннадий	Григорьевич	9	02.02.1978
	Долгов	Демьян	Гурович	10	02.02.1978
	Ершов	Кирилл	Денисович	10	02.02.1978
	Зинин	Данила	Дмитриевич	9	02.02.1978
	Санаев	Иван	Иванович	11	01.01.1980
	Сидоров	Петр	Сергеевич	12	01.01.1979
	Фомин	Сидор	Михайлович	11	02.02.1978

Рис. 32. Вид таблицы после устранения дублирования записей

Ниже приведены сформированные запросы.

Запрос на создание новой таблицы на базе исходной, в которой отсутствуют повторяющиеся записи.

```
SELECT DISTINCT t1.* INTO t3
FROM t1;
```

Запрос на очистку исходной таблицы.

```
DELETE t2.*
FROM t2;
```

Запрос на добавление таблицы, в которой отсутствуют повторяющиеся записи в исходную таблицу.

```
INSERT INTO t1
SELECT t3.*
FROM t3;
```

Все эти запросы сгруппированы в макрос и выполняются автоматически друг за другом при обращении к макросу. Макрос поставлен в соответствии событию “нажатие кнопки” “Исключение дублирования записей”.

3.3. Оценка вычислительной сложности процедур анализа типов атрибутов таблицы и избавления от дублирующихся записей

Оценка вычислительной сложности процедур анализа типов

Базовым оператором для оценки вычислительной сложности является оператор преобразования типа. Во-первых, это один из самых долго выполняемых операторов, во-вторых, он присутствует во всех циклах.

Главная составляющая оценки – это число записей таблицы или ее мощность (m). Действительно необходимо проверить поля всех записей.

Второй составляющей оценки является число столбцов или степень таблицы (n), ведь необходимо проверить все столбцы.

Третья составляющая оценки является число преобразуемых таблиц (k).

Таким образом, вычислительная сложность процедуры анализа типов составит VS = $\sum_k m_k n_k$.

Эта вычислительная сложность для данной предметной области небольшая. Ведь для 100 таблиц с 1000 записями и 100 столбцами (БД более чем средней сложности) число операций преобразований типа будет 10.000.000. Как показал эксперимент, данное количество сравнений на компьютере с частотой 1,6 МГц выполняется за 11 секунд.

Самой большой временной составляющей в процессе анализа и формирования типов столбцов ИТВ являются временные затраты на действия разработчика, которые трудно оценить. Но, как показывает опыт разработок, при использовании разработанных процедур время назначения типов столбцов в ИТВ сокращается в десятки раз.

ПРИЛОЖЕНИЕ 2.

АКТ ВНЕДРЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ