Министерство образования и науки Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Факультет информатики
Кафедра программной инженерии (ПИ)

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП
д-р техн. наук, профессор
_____ С.П. Сущенко
«_10_»_____06_____2016 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА КОМПОНЕНТА ПРЕОБРАЗОВАНИЯ «ERM-МОДЕЛЬ – РЕЛЯЦИОННАЯ МОДЕЛЬ» В СРЕДЕ ORACLE DESIGNER

по основной образовательной программе подготовки бакалавров
направление подготовки 02.03.03 - Математическое обеспечение и администрирование информационных систем

Петров Евгений Денисович

Руководитель ВКР, канд. техн.
наук, доцент кафедры ПИ
_____А.М. Бабанов
подпись
«_1_» _____июня_____2016 г.

Автор работы
студент группы №
_____Е.Д. Петров
подпись

Томск-2016

Реферат

Выпускная квалификационная работа 43 с., 17 рис., 14 источников, 2 прил.

ORACLE DESIGNER, CASE-СИСТЕМА, ERM-МОДЕЛЬ, ERM-СХЕМА, РЕЛЯЦИОННАЯ МОДЕЛЬ, РЕЛЯЦИОННАЯ СХЕМА, ТРАНСЛЯЦИЯ СХЕМ, PL-SQL

Объект исследования – процесс трансляции ERM-схемы в реляционную схему в среде Oracle Designer.

Цель работы – разработать компонент для среды Oracle Designer, обеспечивающий преобразование «ERM-модель – реляционная модель».

Результат работы – получен исследовательский прототип компонента преобразования «ERM-модель – реляционная модель» в среде Oracle Designer.

Дальнейшие исследования – дополнение компонента новыми функциями и возможностями; разработка графического редактора для ERM-схем.

ОГЛАВЛЕНИЕ

Перечень терминов и условных обозначений

CASE-средство – (англ. Computer-Aided Software Engineering) набор инструментов проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

ER-модель – модель «Сущность – Связь» (англ. Entity – relationship model).

ERM-модель, ERMM – модель «Сущность – Связь – Отображение» (англ. Entity – relationship – mapping model).

OD – Oracle Designer.

RON – Repository Object Navigator.

МинКЧ – минимальное кардинальное число.

МаксКЧ – максимальное кардинальное число.

СУБД – система управления базами данных.

ПрО – предметная область.

Введение

Сегодня, наиболее распространённой технологией для хранения данных в информационном мире являются базы данных. Базы данных используются во всех сферах деятельности человека, позволяют хранить и эффективно обрабатывать большие объёмы информации.

Наиболее трудоёмким этапом в создании эффективной базы данных является проектирование. Облегчить работу проектировщика на этом этапе позволяет семантическая методика проектирования баз данных, которая состоит из следующих этапов: проектирование семантической схемы базы данных с использованием той или иной модели, трансляция полученной схемы в реляционную модель, нормализация полученной реляционной схемы. Самой распространённой моделью данных является ER-модель, однако она, как и многие другие модели, не имеет больших описательных возможностей, необходимых для представления всех законов и ограничений предметной области. В связи с этим, проектировщику часто приходится дорабатывать ограничения вручную путём написания триггеров, функций и процедур.

Для проектирования баз данных рекомендуется использовать модель данных, элементы которой наиболее точно отображают объекты реального мира. Примером такой модели является ERM-модель или как её ещё называют модель «Сущность – Связь – Отображение». Богатство её описательных возможностей позволяет говорить о том, что с её помощью выявление семантики предметной области и полная её формализация может быть осуществлена один раз на этапе анализа предметной области [2].

ERM-модель существует уже продолжительное время и сейчас идёт разработка полноценного CASE-средства, которое сможет обеспечить всю полноту её описательных возможностей. На сегодняшний день мы имеем только репозиторий, созданный путём расширения репозитория Oracle Designer и способный хранить объекты ERM-модели, а также механизм проверки ERM-схем на непротиворечивость.

Цель данной дипломной работы – разработка компонента преобразования «ERM-модель – реляционная модель» в среде Oracle Designer, используя правила преобразования.

Для достижения данной цели были выделены следующие задачи:

- реализация механизма для работы с элементами реляционной модели;
- определение правил преобразования «ERM-модель – реляционная модель»;
- реализация механизма преобразования «ERM-модель – реляционная модель».

1 Используемые модели данных

1.1 ERM-модель

ERM-модель – семантическая модель данных, являющаяся преемницей модели «Сущность-Связь» (ER-модели), однако существенно расширяющая её возможности. По сравнению с ER-моделью, ERMM имеет более высокий уровень абстракции. Это позволяет говорить о том, что ER-модель является частным случаем модели ERM. Наиболее полно модель описана А.М. Бабановым в статье «Семантическая модель «Сущность – Связь – Отображение» [2].

ERM-модель имеет существенные отличия по сравнению со своими предшественниками. В качестве базовых концепций данной модели выступают новые понятия «класс» и «отображение». Они существенно расширяют выразительные возможности модели по сравнению с ER-моделью и обеспечивают должный уровень абстракции, доведённой в теории семантически значимых отображений (ТСЗО) до формальной системы [1].

Все понятия ER-модели (например, такие, как: множество сущностей, множество связей, роль, атрибут, множество значений) могут быть выражены через понятия ERM-модели. Однако в дополнение к этому в ERM-модель введены и некоторые понятия ER-модели. Они обеспечивают более понятные человеку формы восприятия данных. Подобный подход позволяет проектировщику оперировать в основном знакомыми понятиями, прибегая к использованию новых форм лишь в случае недостаточной выразительности первых. Возможны ситуации, когда при проектировании ERM-схемы используются только правила структуризации и задания ограничений целостности, целиком принадлежащие ER-модели, а новые возможности явно не применяются [7].

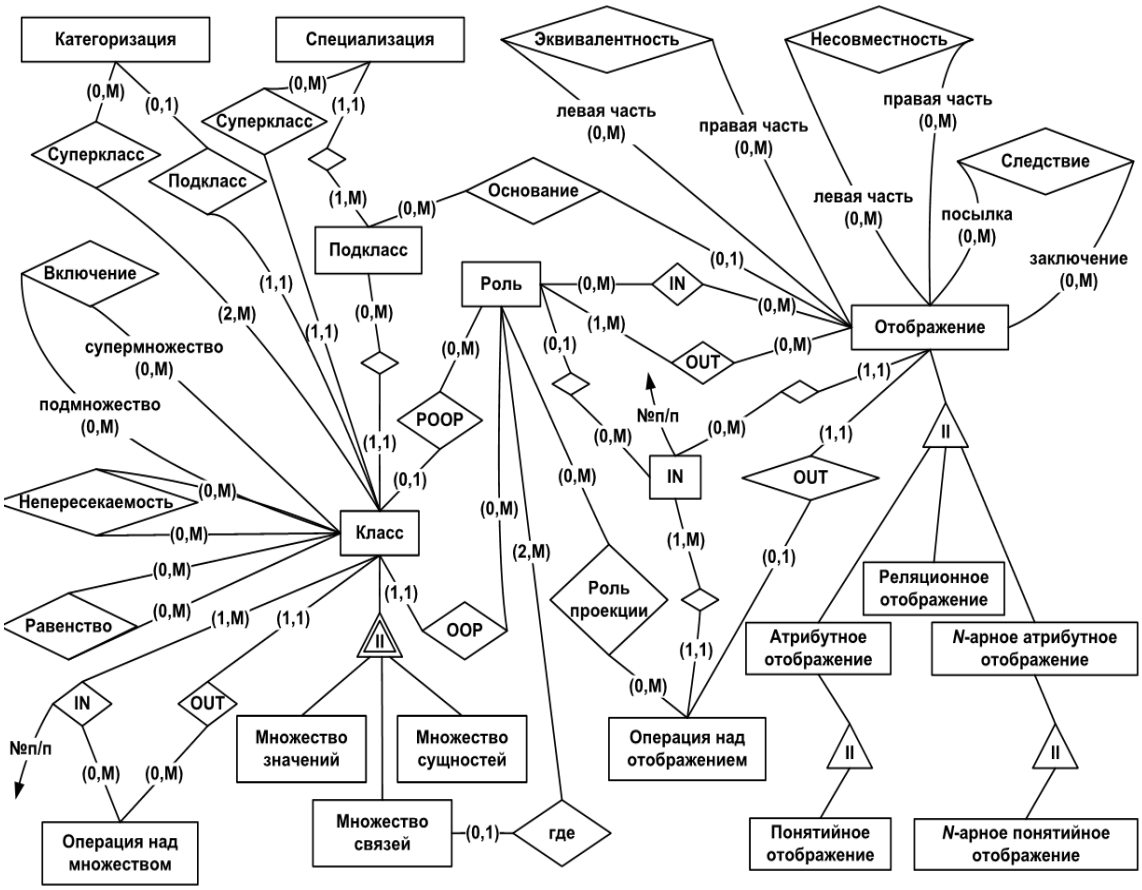


Рисунок 1. Метасхема ERM-модели

Помимо всего перечисленного ERM-модель предлагает следующие нововведения:

- «класс» как обобщение понятий «множество сущностей», «множество связей» и «множество значений»;
- «атрибутивные» и «реляционные» отображения;
- операции следствия, несовместности и эквивалентности между отображениями;
- отношения равенства, включения и непересекаемости для классов;
- алгебра отображений – набор операций, задаваемых на множестве отображений;
- основание деления – отображение, определяющее к какому подклассу специализации, принадлежит объект суперкласса;
- специализации и категоризации подняты на уровень класса.

Все понятия ERM-модели делятся на базовые и производные. Базовые понятия также делятся на структурные (класс и отображение) и дополнительные (специализация и категоризация, роли объектов-прообразов и объектов-образов в экземплярах отображений, а также операции и отношения). Производными же понятиями являются понятия ER-модели, расширенные понятием специализации.

Обе системы структурных понятий (понятия ER-модели и базовые понятия ERM-модели) не являются независимыми, их связывают отношения обобщения, т.е. понятия ER-модели являются специализациями базовых понятий ERM-модели. По мере надобности проектировщик вправе использовать подходящие структурные понятия обеих групп [6].

1.2 Реляционная модель

Реляционные БД появились ещё в начале 1970-х годов и со временем обрели большую популярность. На сегодняшний день реляционная модель является развитой системой и имеет следующие основные структурные понятия:

- тип данных;
- домен;
- атрибут;
- кортеж;
- отношение.

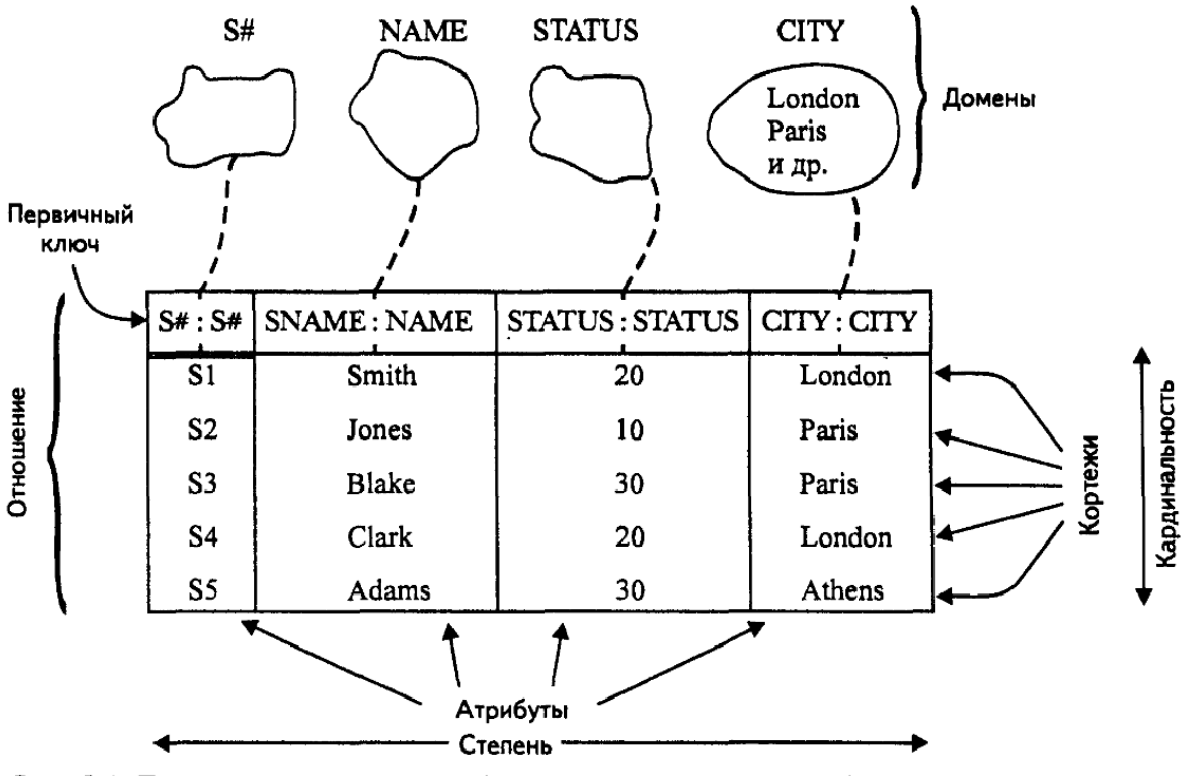


Рисунок 2 Пример реляционной модели данных

Понятие **типа данных** (*data type*, или сокращённо **типа**) является фундаментальным; каждое значение, каждая переменная, каждый параметр, каждый оператор, предназначенный только для чтения, и особенно каждый реляционный атрибут относится к тому или иному типу. Кроме всего прочего, он представляет собой множество значений. К примерам таких типов относятся INTEGER (множество всех целых чисел), CHAR (множество всех символьных строк) и др. Отсюда следует то, что типы можно также называть **доменами** (*domain*), особенно когда речь идёт о реляционной модели [9].

Если дана коллекция типов *Ti* (*i* = 1, 2, . . ., *n*), которые не обязательно все должны быть разными, то значением **кортежа** (tuple, или кратко **кортежем**), определённым с помощью этих типов (назовём его t), является множество упорядоченных троек в форме <*Ai*,*Ti*,*vi*>, где *Ai* — *имя атрибута*, *Ti* — *имя типа* и *vi* — *значение типа Ti* [9].

Значение (*value*) представляет собой «отдельно взятую константу», например, конкретную константу, которая выражается в виде целого числа 3. Для значения не определено место во времени или в пространстве. Но значе-

ния могут быть представлены в памяти с помощью некоторого метода кодирования и такие представления или проявления характеризуются определённым местоположением во времени и пространстве [9].

Термин **отношение** (*relation*) – математическое название **таблицы** (*table*) (точнее определённого вида таблиц). В настоящее время в неформальном контексте термины **отношение** и **таблица** принято считать синонимами. На практике в подобном контексте термин **таблица** используется гораздо чаще, чем термин **отношение**. [9].

В верхней части таблицы-отношения задаётся перечень наименований **атрибутов**. **Атрибуты** (*attribute*) отношения выполняют функцию наименований его столбцов и содержательно описывают смысл и назначение элементов данных, располагаемых в соответствующих ячейках [8].

Неопределённое значение (NULL) указывает, что значение атрибута в настоящий момент неизвестно или неприемлемо для этого кортежа.

Суперключ (*superkey*). Атрибут или множество атрибутов, которое единственным образом идентифицирует кортеж данного отношения.

Потенциальный ключ (*candidate key*). Суперключ, который не содержит подмножества, также являющегося суперключом данного отношения.

Первичный ключ (*primary key*). Потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения. Поскольку отношение не содержит кортежей-дубликатов, всегда можно уникальным образом идентифицировать каждую его строку. Это значит, что отношение всегда имеет первичный ключ. В худшем случае все множество атрибутов может использоваться как первичный ключ, но обычно, чтобы различить кортежи, достаточно использовать несколько меньшее подмножество атрибутов. Потенциальные ключи, которые не выбраны в качестве первичного ключа, называются **альтернативными ключами**.

Суррогатный ключ (*surrogate key*) – это искусственный атрибут отношения, не имеющий связей с какой-либо естественной характеристикой явлений ПрО и вводимый в схему отношения исключительно для организации связей кортежей этого отношения с кортежами других отношений.

Внешний ключ (*foreign key*). Атрибут или множество атрибутов внутри отношения, который соответствует потенциальному (как правило, первичному) ключу некоторого (может быть, того же самого) отношения и является его подмножеством. Если некий атрибут присутствует в нескольких отношениях, то его наличие обычно отражает определённую связь между кортежами этих отношений.

2 Среда и инструменты разработки

2.1 Oracle Designer

Oracle Designer (*OD*) – CASE инструмент фирмы Oracle Corporation, используемый для проектирования и создания информационных систем. После создания информационной системы проектировщик имеет возможность редактировать сгенерированный код при помощи Oracle Developer Suite (набор инструментов разработки, выпущенный корпорацией Oracle, основными компонентами которого являются: Oracle Forms, Oracle Reports и JDeveloper).

Для разработки генератора и создания объектов репозитория используются следующие компоненты OD:

- Repository Object Navigator (RON) – инструмент для создания, хранения и редактирования прикладных систем и их объектов, также предоставляет возможности администрирования репозитория. RON предоставляет хранящиеся в репозитории элементы в виде иерархии тем самым позволяя быстро и удобно создавать, находить и редактировать объекты.

– Design Editor – инструмент для проектирования базы дынных и приложений. Предоставляет графический способ просмотра и манипулирования хранящимися в репозитории представлениями объектов.

Так же для работы с объектами репозитория и создания пакетов и функций использовался Alt SQL Developer. Alt SQL Developer представляет собой интегрированную среду разработки для объектов базы данных Oracle и предоставляет удобные возможности для создания, редактирования, тестирования, отладки и оптимизации SQL запросов, хранения процедур написанных на PL/SQL, включая пакеты, функции, триггеры, определяемые пользователем типы данных, и т.п.

2.2 Oracle Designer API

Oracle Designer API – это набор представлений базы данных и пакетов PL/SQL в схеме владельца репозитория, которые обеспечивают возможность безопасного доступа к данным репозитория. API содержит множество представлений таблиц репозитория, в которых отражаются реальные объекты репозитория (например сущности и атрибуты). Эти представления являются важной частью API, так как они позволяют исследовать описания, создаваемые в прикладных системах[11].

Вторая составляющая API – пакеты PL/SQL, позволяющие безопасно изменять содержимое таблиц из средства, внешнего по отношению к OD. С помощью этих пакетов можно дополнить средства построения диаграмм и утилиты OD собственными программами или программными конструкциями. Кроме того, инструментальные средства OD используют API для ввода, обновления, удаления и выбора данных из репозитория.

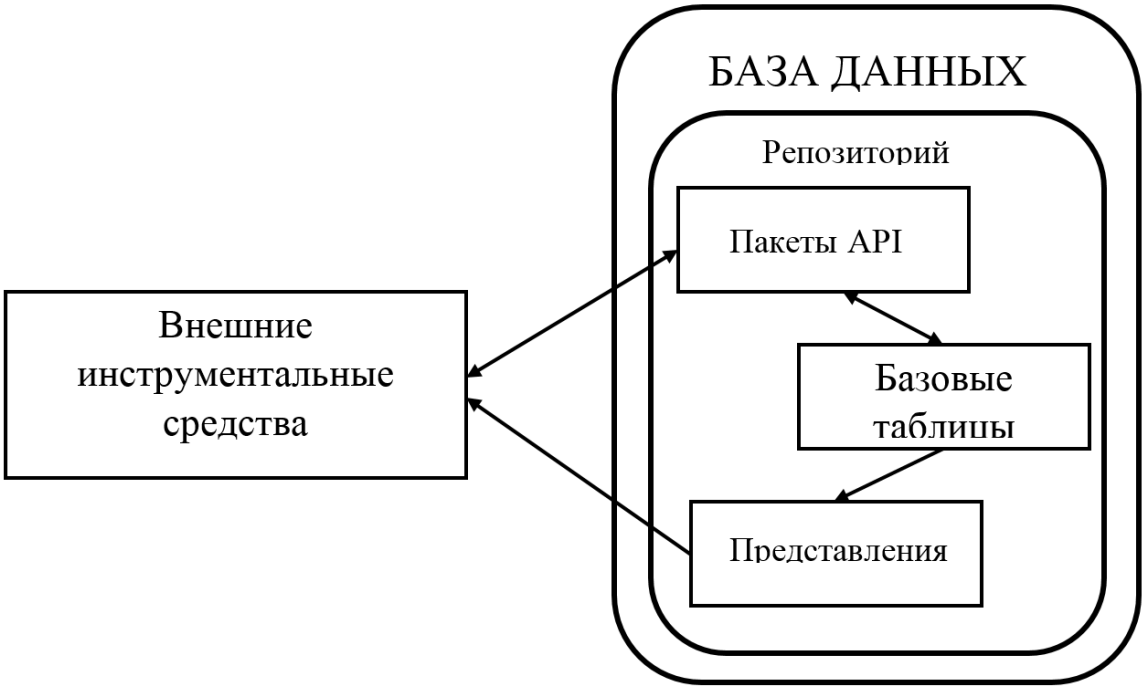


Рисунок 3 Обращение API к репозиторию

Посредством вызовов API реализуется **модель транзакции**, определяющая логическое начало и окончание единицы работы. Эта модель отчасти похожа на стандартную модель транзакций SQL, в которой для обозначения и отката транзакции применяются операторы COMMIT и ROLLBACK. Однако в модели транзакции API применяется несколько дополнительных методов для обработки ошибок[11]. Для каждого типа объектов, который может хранить репозиторий, Oracle автоматически создаёт:

- таблицу, в которой будут храниться все созданные объекты этого типа;
- представление (имя которого имеет следующий вид – *CI_<тип_объекта>*), которое представляет данные из таблицы в более читабельном виде;
- пакет PL/SQL (имя которого имеет вид – CIO<*тип_объекта*>), для безопасного изменения содержимого таблицы. Перечисленные элементы используются для работы с каким-либо объектом, который может храниться и обрабатываться репозиторием.

Этапы транзакции API (осуществляется посредством пакета CDAPI):

1. Инициализация (объявляется какая прикладная система, и её версия используется);
2. Открытие (начало транзакции);
3. Загрузка значений переменной записи:
4. Загрузка указателей переменной записи;
5. Выполнение операции DML (обновляется описание репозитория);
6. Подтверждение правильности;
7. Сообщение об ошибках;
8. Закрытие (подтверждаются данные и состояние в конце);
9. Прерывание из-за сбоя.

Пример модели транзакции:

```
1. CDAPI.INITIALIZE(CURR_APP_SYS_NAME => SCHEMA_NAME);
2. CDAPI.OPEN_ACTIVITY;
3. TAB_DEF.V.NAME := TABLE_DEFINITION_NAME;
4. TAB_DEF.I.NAME := TRUE;
5. CIOTABLE_DEFINITION.INS(ID => NULL,PL => TAB_DEF);
6. CDAPI.VALIDATE_ACTIVITY(V_STATUS, V_WARNING);
7. CDAPI.INSTANTIATE_MESSAGE;
8. CDAPI.CLOSE_ACTIVITY(ACT_STATUS => V_STATUS);
9. CDAPI.ABORT_ACTIVITY;
```

2.3 Пользовательский репозиторий, поддерживающий ERM-модель

Как говорилось ранее, сегодня активно идёт разработка CASE – средства, которое сможет поддерживать ERM-модель. На данный момент готов репозиторий, способный хранить и обрабатывать все понятия ERM-модели. Разработан репозиторий на базе Oracle Designer, путём добавления в последний пользовательских типов и API для работы с ними.

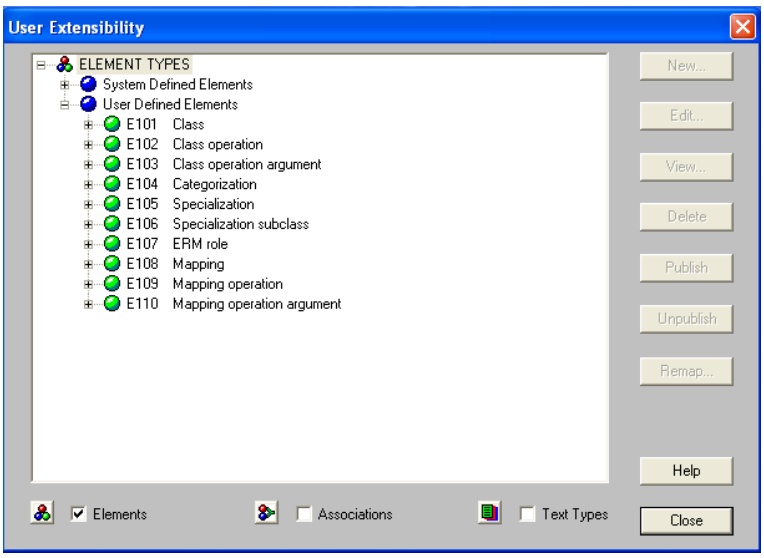


Рисунок 4 Список поддерживаемых объектов ERM-модели

API для работы с понятиями ERM-модели, сохранёнными в репозитории, представлен пакетом ERM_API, который содержит в себе все необходимые процедуры (*SELECT* – для выборки, *INSERT* – для вставки, *UPDATE* – для обновления, *DELETE* – для удаления) для работы с каждым понятием модели. Так же имеется пакет ERM_CHECK, который служит для проверки схем на непротиворечивость.

3 Правила трансляции схем ERM-модели в реляционную модель

Непрекращающиеся поиски решения проблемы проектирования БД привели к появлению семантических моделей данных, которые обеспечивают высокоуровневое, формальное представление ПрО. Они должны были устранить первые два недостатка классической методики – неестественность представлений для человека и неприменимость для анализа сложных ПрО[4].

Но, поскольку коммерческие СУБД не собираются поддерживать напрямую семантические модели данных, необходима трансляция схем БД с языка этих моделей на язык коммерческих СУБД и в первую очередь – реляционных. Примерно так, практически с первых публикаций по ER-модели стала выри-

совываться основная схема семантической методики проектирования реляционных БД[4]. Один из основных этапов семантической методики проектирования – перевод полученной схемы в реляционную модель с применением подходящего набора правил трансформации и получение множества предварительных отношений.

ERM-модель имеет огромные описательные возможности и способна существенно облегчить жизнь проектировщику, однако до настоящего времени не было ни одного CASE-средства, способного поддерживать модель. Сегодня уже есть репозиторий способный хранить и обрабатывать понятия ERM-модели. Следующий этап разработки CASE-средства является разработка генератора реляционных схем из ERM-схем. Было принять начать разработку генератора с реализации правил трансляции, из методики преобразования «ERM-схема – реляционная схема».

Методика преобразования «ERM-схема – реляционная схема» осуществляется по правилам, которые были получены Бабановым А. М. в ходе его научных исследований. Для разработки генератора были использованы следующие правила, касающиеся в основном производных понятий ERM-модели.

Правила порождения структур:

Правило 1. Каждому множеству сущностей соответствует своё отношение, включающее следующие атрибуты:

- суррогатный первичный ключ;
- однозначные атрибуты этого множества связей;

Имя отношения совпадает с именем множества сущностей. Имя первичного ключа образуется добавлением к имени отношения суффикса «_ID». Имена атрибутов совпадают с именами соответствующих отображений.

Правило 2. Если у множества связей имеются однозначные или многозначные атрибуты, то для такого множества связей строится своё отношение, включая следующие атрибуты:

- суррогатный первичный ключ;
- однозначные атрибуты этого множества связей;

Имя отношения совпадает с именем множества связей. Имя первичного ключа образуется добавлением к имени отношения суффикса «_ID». Имена атрибутов совпадают с именами соответствующих отображений.

Правило 3. Каждой многозначной характеристике каждого класса (множества сущностей или множества связей) соответствует своё бинарное отношение с атрибутами:

- внешний ключ, представляющий собой дубликат первичного ключа класса;
- атрибут для значения характеристики;

Имя такого отношения составляется из имени характеристики и имени класса в родительном падеже. Имя внешнего ключа совпадает с именем первичного ключа класса. Имя второго атрибута – имя соответствующего многозначного отображения - характеристики.

Правило 4. Каждому множеству связи степени *n* соответствует своё отношение с *n* атрибутами, каждый из которых представляет собой внешний ключ, который является дубликатом первичного ключа отношения, построенного для соответствующего множества сущностей. Именем каждого такого отношения становится соответствующий предикатор. Имена внешних ключей совпадают с именами первичных ключей множеств сущностей. В том случае, когда одно и то же множество сущностей играет сразу несколько ролей во множестве связей, имя роли становится дополнительным префиксом имени внешнего ключа.

Правила порождения ограничений целостности:

Правило 5. Ограничения целостности на отображения-характеристики и обратные им отображения.

- Если отображение-характеристика имеет МинКЧ = 1 и МаксКЧ = 1 (полное функциональное), то соответствующий ей атрибут принадлежит отношению класса и снабжается описателем *NOT NULL;*

– Если отображение-характеристика имеет МинКЧ = 0 и МаксКЧ = 1 (частично функциональное), то соответствующий ей атрибут принадлежит отношению класса и снабжается описателем *NULL;*
– Если отображение-характеристика имеет МаксКЧ = ∞ (неограниченно), то соответствующий ей атрибут принадлежит специальному дополнительному отношению и снабжается описателем *NOT NULL (*как и внешний ключ, представляющий собой дубликат первичного ключа класса, которому принадлежит атрибут);
– Если отображение, обратное однозначному отображению-характеристике, имеет МаксКЧ = 1 (функционально), то соответствующий атрибут отношения класса объявляется *UNIQUE* (возможный ключ);
– Если отображение, обратное однозначному отображению-характеристике, имеет МаксКЧ = ∞ (неограниченно), то соответствующий атрибут отношения класса объявляется, как *NOT UNIQUE*;
– Если отображение, обратное многозначному отображению-характеристике, имеет МаксКЧ = 1 (функционально), то соответствующий атрибут дополнительного отношения объявляется *PRIMARY KEY* (первичный ключ);
– Если отображение, обратное многозначному отображению-характеристике, имеет МаксКЧ = ∞ (неограниченно), то оба атрибута дополнительного отношения (внешний ключ и сам атрибут) объявляются *PRIMARY KEY* (первичный ключ);

Правило 6. Ограничения целостности на реляционные отображения и обратные им отображения между классами объектов.

– Если реляционное отображение имеет МинКЧ = 1 и МаксКЧ = 1 (полное функциональное) и при этом отображение, обратное реляционному имеет МинКЧ = 1 и МаксКЧ = 1 (полное функциональное), то для такой пары классов создаётся новое отношение. Новое отношение образуется путём слияния отношений, уже созданных

для соответствующих классов. Имя отношения будет образованно из имён рассматриваемых классов по шаблону «*имя первого класса _ имя второго класса*». Возможным ключом (*UNIQUE)* нового отношения является группа, состоящая из обоих первичных ключей отношений, и оба они имеют описатель *NOT NULL*.

- Если реляционное отображение имеет МинКЧ = 1 и МаксКЧ = 1 (полное функциональное) и при этом отображение, обратное реляционному имеет МинКЧ = 0 и МаксКЧ = 1 (частично функциональное), то для такой пары классов первичный ключ из отношения для первого класса дублируется в отношение, построенное для второго класса. Соответствующий внешний ключ объявляется *UNIQUE* и имеет описатель *NOT NULL*.
- Если реляционное отображение имеет МинКЧ = 0 и МаксКЧ = 1 (частично функциональное) и при этом отображение, обратное реляционному имеет МинКЧ = 0 и МаксКЧ = 1 (частично функциональное), то для такой пары классов создаётся дополнительное отношение, куда дублируются первичные ключи отношений, построенных для соответствующих классов. Каждый из этих внешних ключей является возможным ключом (*UNIQUE)* нового отношения и имеет описатель *NOT NULL*.
- Если реляционное отображение имеет МинКЧ = 1 и МаксКЧ = 1 (полное функциональное) и при этом отображение, обратное реляционному имеет МаксКЧ = ∞ (неограниченно), то для такой пары классов первичный ключ из отношения для первого класса дублируется в отношение, построенное для второго класса и имеет описатель *NOT NULL*.
- Если реляционное отображение имеет МинКЧ = 0 и МаксКЧ = 1 (частично функциональное) и при этом отображение, обратное реляционному имеет МаксКЧ = ∞ (неограниченно), то для такой пары классов создаётся дополнительное отношение, куда дублируются

первичные ключи отношений, построенных для соответствующих классов. Внешние ключи отношения имеют описатель *NOT NULL*. Внешний ключ, являющийся дубликатом первичного ключа из второго отношения, объявляется *UNIQUE*.

– Если реляционное отображение имеет МаксКЧ = ∞ (неограниченно) и при этом отображение, обратное реляционному имеет МаксКЧ = ∞ (неограниченно), то для такой пары классов создаётся дополнительное отношение, куда дублируются первичные ключи отношений, построенных для соответствующих классов. Возможным ключом нового отношения является группа, состоящая из обоих внешних ключей, и оба они имеют описатель *NOT NULL*.

4 Практическая реализация трансляции схем

4.1 Описание генератора

Прежде чем объяснять устройство и принцип работы компонента преобразования (далее просто генератор) следует сделать одно замечание. Генератор работает в пределах одной прикладной системы (*Application System*). Это означает, что прежде чем запустить генератор необходимо создать прикладную систему и задать ERM-схему. После того, как генератор закончит свою работу, итоговая реляционная схема будет добавлена в ту же прикладную систему. При этом пользователю будут доступны обе схемы для редактирования.

Состав разрабатываемого генератора:

1. Пакет *R_PKG_GENERATOR* (предназначен для описания процедур и функций, необходимых для работы с ERM-моделью).
2. Пакет *API_PKG_GENERATOR* (предназначен для описания процедур и функций, необходимых для работы с реляционной моделью, посредством Oracle Designer API).

3. Набор объектных типов (объектные типы описывают элементы реляционной модели и используются пакетом *API_PKG_GENERATOR).*

Пакеты в Oracle Designer состоят из двух частей – спецификация (*Spec*) и тело (*Body*). Первая описывает все компоненты и переменные пакета, предоставляя, таким образом, краткое описание последнего. Вторая же представляет реализацию функций и процедур, описанных в спецификации.

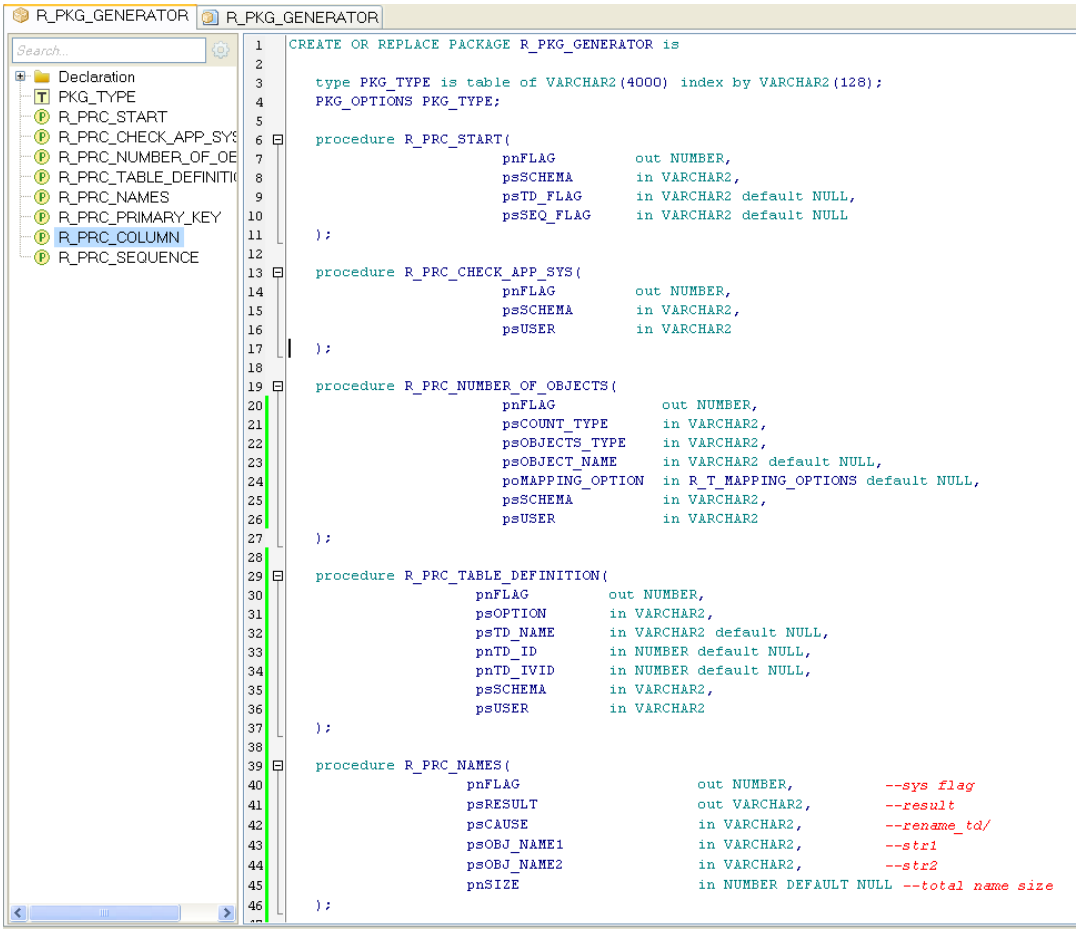


Рисунок 5 Пример пакета в Alt SQL Developer

В пакете *R_PKG_GENERATOR* реализованы следующие процедуры:

1. Процедуры генерации:
   a. *R_PRC_START* (проверка заданных параметров, инициализация переменных, запуск генератора);
   b. *R_PRC_RUN* (процедура трансляции схем);
2. Вспомогательные процедуры:

a. *R_PRC_NAMES* (процедура, для генерации имён элементов реляционной модели);
b. *R_PRC_NUMBER_OF_OBJECTS* (проверяет существует ли элемент модели с такими параметрами; подсчитывает общее количество элементов модели в прикладной системе).

3. Шаблонные процедуры для элементов реляционной модели (реализация наиболее часто используемых действий над объектами).

В пакете *API_PKG_GENERATOR* реализованы процедуры для добавления, удаления и обновления, посредством Oracle Designer API, следующих элементов реляционной модели:

1. Table Definition.
2. Primary Key.
3. Sequence.
4. Column.
5. Unique Key.
6. Foreign Key.
7. Index.

4.2 Устройство и принцип работы генератора

На вход генератора подаётся имя прикладной системы и набор флагов, позволяющих определить поведение генератора в спорных моментах. Процедура запуска генератора имеет следующий вид:

```
begin
    R_PKG_GENERATOR.R_PRC_START(pnFLAG, psSCHEMA, [,psFLAGS])
end;
```

В данном примере используются следующие имена и переменные:

– *R_PKG_GENERATOR* – имя вызываемого пакета;
– *R_PRC_START* – имя вызываемой процедуры;

– *pnFLAG* – возвращаемое значение процедуры (возвращает числовое значение в зависимости от итога работы генератора);
– *psSCHEMA* – имя прикладной системы, показывает откуда берётся ERM – схема и записывается реляционная схема;
– *psFLAGS* – список флагов, определяющих поведение генератора в спорных моментах (например что делать с уже созданными Table Definitions или Sequences). Может принимать значения – *delete/rename/ignore*. Значение флага по умолчанию – *rename*.

После запуска стартовый скрипт проверяет заданные параметры и создаёт необходимое для генератора окружение: массивы, временные таблицы, пользовательские типы. Если заданные параметры корректны и необходимое окружение успешно создано, скрипт запускает сам генератор.

Пользователь, запустивший скрипт может перейти во вкладку *Output* окна, в котором был запущен скрипт, и наблюдать за ходом выполнения трансляции (рис. 6). Для вывода данной информации используется стандартный пакет PL/SQL *DBMS_OUTPUT*, предназначенный для вывода информации во временный буфер.

Сам генератор работает по следующей схеме:

```
procedure R_PRC_RUN(pnFLAG out number) is
  «Инициализация переменных»
begin
  if «количество классов в прикладной системе» > 0 then
   while «не просмотрены все классы» do
     if "тип класса Мн.Сущ." then
          «выполняются правила 1-3»
     elsif "тип класса Мн.Св." then
          «выполняются правила 1-3»
     elsif "тип класса Мн.Зн." then
          null;
     end if;
   ...
     «повторный запуск цикла для остальных правил»
   ...
  else
     "Выход"
  end if;
end;
```

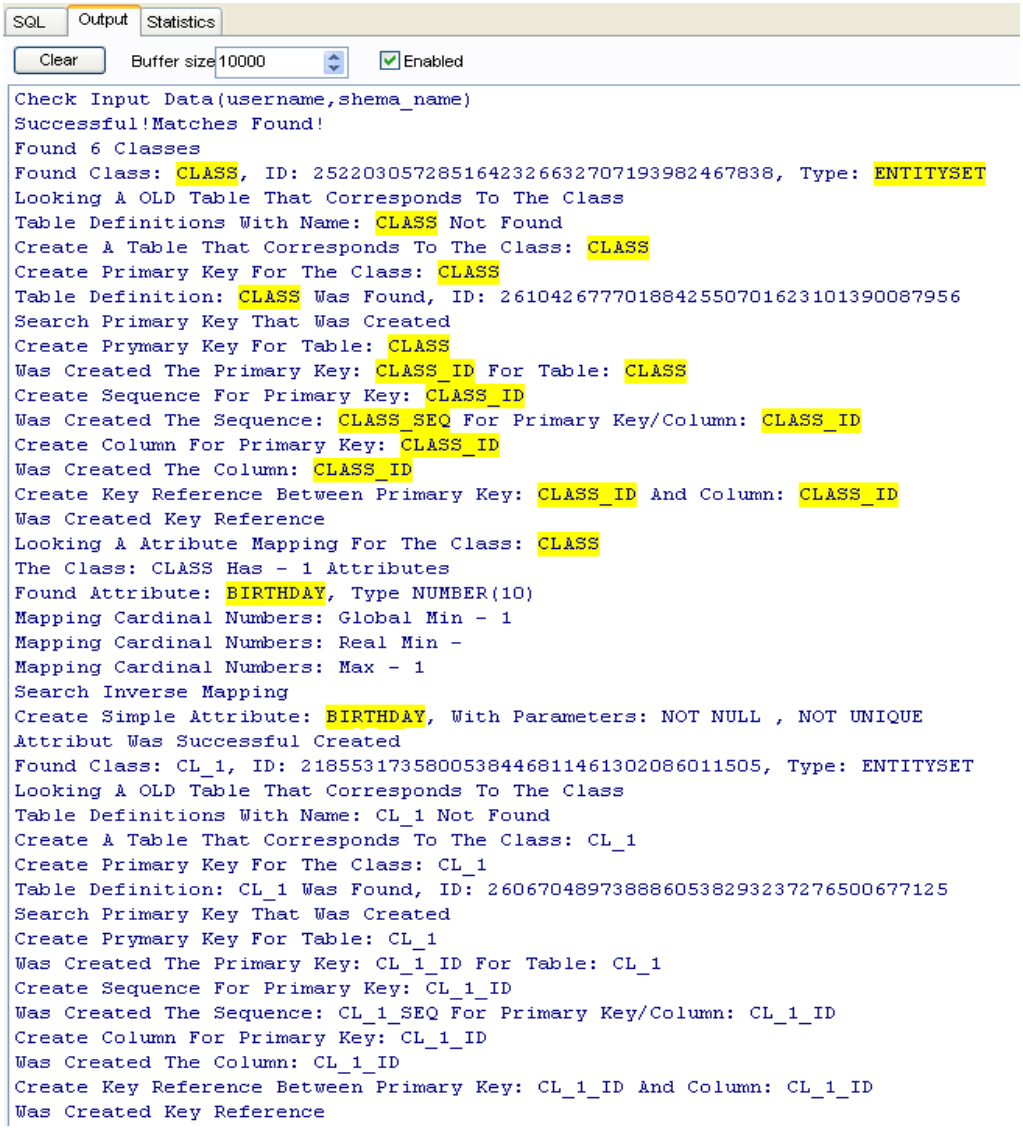


Рисунок 6 Вид вкладки Output во время работы транслятора

Как работает генератор для правил 1-3, описано в приложении Б.

Для удобной работы с объектами реляционной модели в репозитории был разработан отдельный пакет – *API_PKG_GENERATOR*, поддерживающий все возможные операции над элементами реляционной модели. Для каждого элемента реляционной модели в пакете создано 3 процедуры для добавления, удаления и обновления элемента. Вызов таких процедур осуществляется следующим образом (пример вызова процедуры для создания столбцы таблицы):

```
API_PKG_GENERATOR.A_PRC_INS_COL(
                pnFLAG, PKG_OPTIONS('SCHEMA'), oCOLUMN_OPTIONS
);
```

Рисунок 7 Пример вызова API процедуры

В данном примере используются следующие имена и переменные:

- *API_PKG_GENEARTOR* – вызываемый пакет;
- *A_PRC_INS_COL* – вызываемая процедура (имена процедур составляются следующим образом:
  - A – определяет принадлежность к пакету (***A*** – API пакет или ***R*** – пакет генератора);
  - PRC – определяет что вызываем (***PRC*** – процедура или ***FNC*** – функция);
  - INS/UPD/DEL – для чего вызываем;
  - COL/TAB/PK/SEQ... – для какого элемента реляционной модели вызывается.
- *pnFLAG* – флаг для проверки корректности работы процедуры;
- *psSCHEMA* – имя прикладной системы в рамках которой работает генератор;
- *oCOLUMN_OPTIONS* – переменная объектного типа, для описания параметров создаваемого объекта реляционной модели.

Пример процедуры вызываемой выше можно увидеть на рис. 8. Подробнее об использовании Oracle Designer API можно прочесть в главе 2.2.

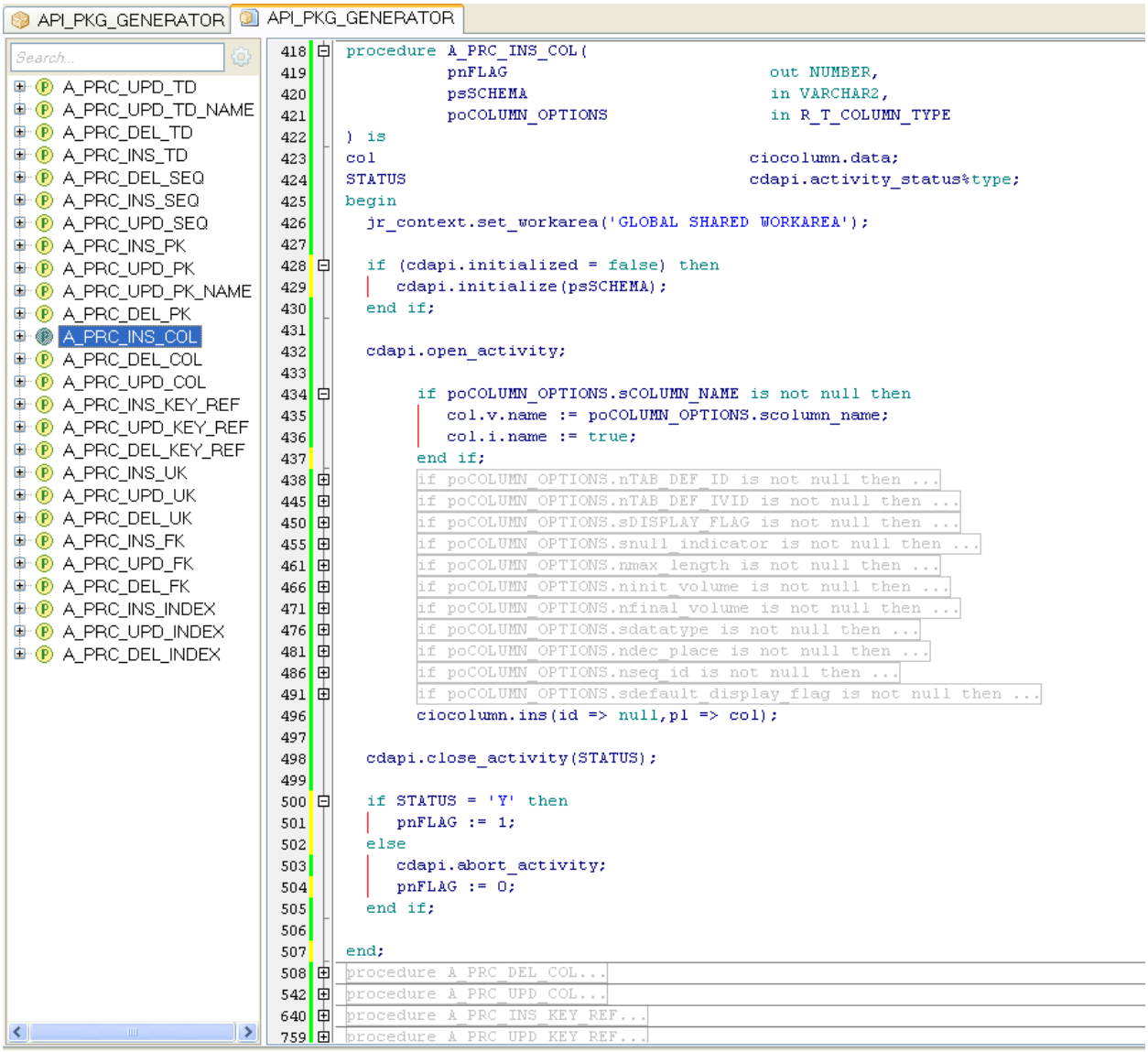


Рисунок 8 Пример процедуры для добавления столбца

Для удобства работы с пакетом API и упрощения его логики, каждому элементу реляционной схемы был задан свой объектный тип. Имена атрибутов в объектном типе соответствуют именам столбцов в системной таблице, в которой хранится элемент реляционной модели. Пример пользовательского типа (для столбцов):

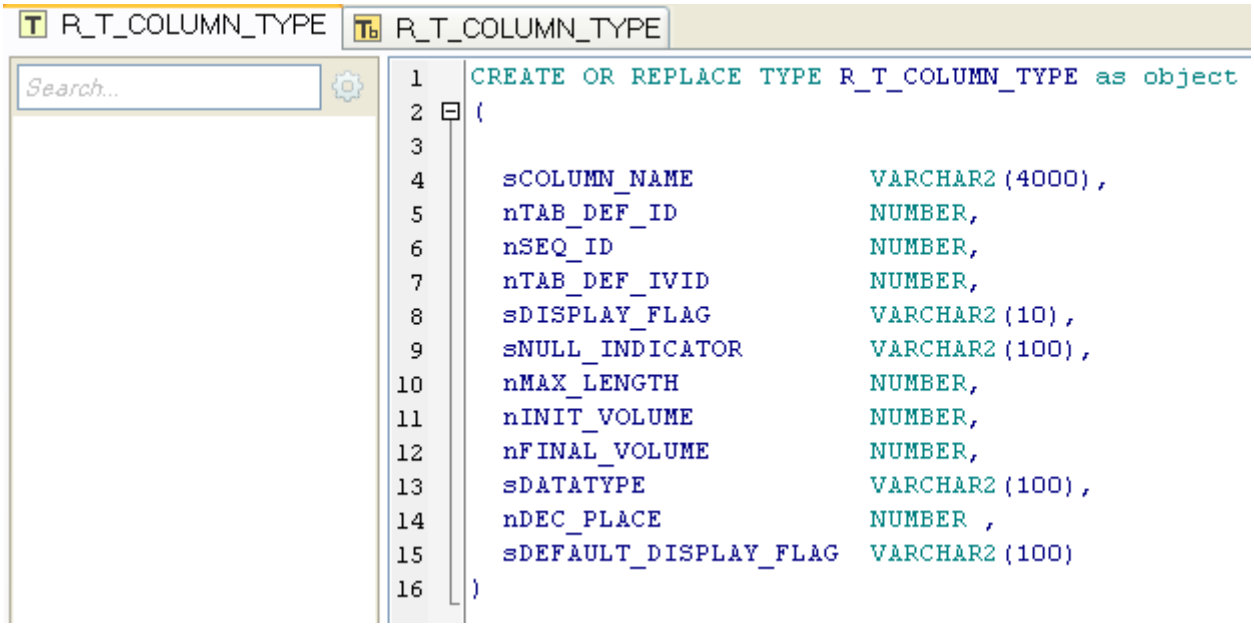


Рисунок 8 Пример описания объектного типа для столбца

Объявление переменной объектного типа происходит следующим образом (пример для столбца):

```
oCOLUMN_OPTIONS := (null);
oCOLUMN_OPTIONS := R_T_COLUMN_TYPE(
                              sCOLUMN_NAME => sCOL_NAME,
                              nTAB_DEF_ID => to_number(PKG_OPTIONS('CURRENT_TD_ID')),
                              nSEQ_ID => to_number(PKG_OPTIONS('CURRENT_SEQ_ID')),
                              nTAB_DEF_IVID => to_number(PKG_OPTIONS('CURRENT_TD_IVID')),
                              sDISPLAY_FLAG => 'N',
                              sNULL_INDICATOR => 'NOT NULL',
                              nMAX_LENGTH => 10,
                              nINIT_VOLUME => 100,
                              nFINAL_VOLUME => 100,
                              sDATATYPE => 'NUMBER',
                              nDEC_PLACE => 0,
                              sDEFAULT_DISPLAY_FLAG => null
);
```

Рисунок 9 Объявление объектного типа

Все описанные выше пакеты, процедуры и типы не имеют жёсткой привязки и могут вызываться, откуда угодно, достаточно лишь указать для какого класса или отношения следует применить выбранную процедуру. Обычно одна процедура соответствует одному правилу. Так же для ряда элементов реляционной модели были реализованы шаблонные процедуры, реализующие методы для изменения определённых свойств элемента.

Например, при необходимости переименовать отношение (Table Definition), без использования шаблонной процедуры, нам необходимо будет проделать следующие шаги:

– для текущего отношения получить новое имя,
– создать переменную-словарь,
– вызвать транзакцию.

Однако вместо этого можно просто вызвать шаблонную процедуру для текущего отношения.

```
R_PRC_TABLE_DEFINITION(
            pnFLAG => pnFLAG,
            psOPTION => 'RENAME',
            psTD_NAME => PKG_OPTIONS('CURRENT_TD_NAME'),
            pnTD_ID => to_number(PKG_OPTIONS('CURRENT_TD_ID')),
            pnTD_IVID => to_number(PKG_OPTIONS('CURRENT_TD_IVID')),
            psSCHEMA => PKG_OPTIONS('SCHEMA'),
            psUSER => PKG_OPTIONS('USER')
);
```

Рисунок 10 Вызов шаблонной процедуры для переименования Table Definition

Подводя итог можно сделать следующий вывод. В большинстве случаев работа генератора будет сводиться к следующим шагам:

– выбрать очередной класс,
– определить правило, под которое подходит выбранный класс,
– в соответствии с правилом определить набор создаваемых объектов реляционной схемы,
– для каждого создаваемого объекта задать переменную-словарь,
– вызвать соответствующую процедуру API или шаблонную процедуру.

В дальнейшем такое устройство генератора, должно облегчить доработку CASE-средства для ERM-модели позволяя добавлять новый функционал, используя подготовленные шаблоны и процедуры.

4.3 Конечный результат работы генератора

По окончанию работы генератора мы получаем полностью построенную реляционную схему, которую можно просмотреть или внести правки при помощи RON (*Repository Object Navigation*), или Design Editor. После этого пользователь может по полученной схеме сгенерировать таблицы и начать вносить данные.

Ниже продемонстрированы результаты работы с разными объектами ERM-модели. Для проверки работоспособности использовался набор ERM-

схем показанных на рисунках ниже. Данные схемы используются только для демонстрации возможностей генератора и не отображают какие-либо объекты реального мира.

Выбранные ERM-схемы затрагивают следующие правила трансляции:

- Рисунок 11 – правило 1 и правило 5.
- Рисунок 12 – правило 1, правило 3 и правило 5.
- Рисунок 13 – правило 1, правило 4, правило 6.
- Рисунок 14 – правило 1, правило3, правило 5, правило 6.

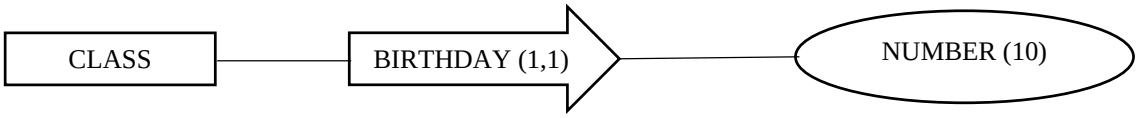


Рисунок 11 ERM-схема для однозначного атрибута

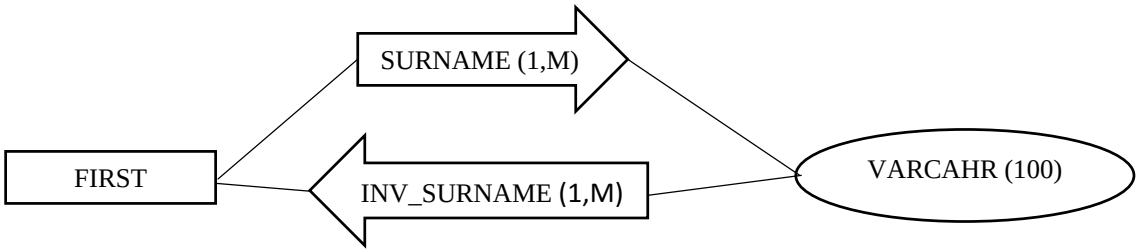


Рисунок 12 ERM-схема для многозначного атрибута

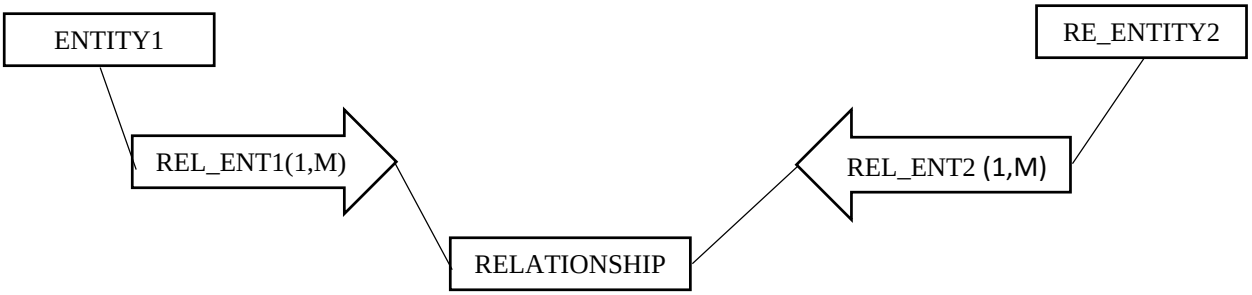


Рисунок 13 ERM-схема для реляционного отображения с участием множества связей

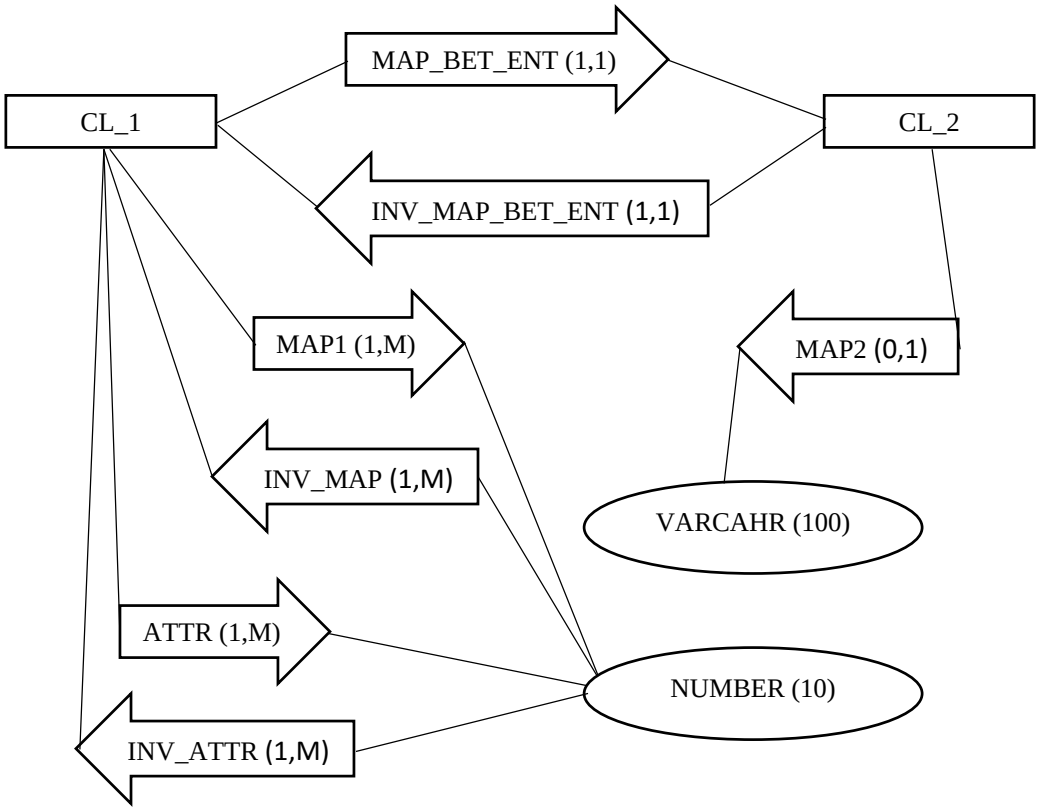


Рисунок 14 ERM-схема для реляционного отображения

На данный момент ERM-схемы можно задать только при помощи RON. Как это выглядит можно увидеть на рисунке 15.

После окончания работы генератора, полученную в результате работы реляционную схему можно посмотреть и изменить все в том же RON. Как это выглядит можно увидеть на рисунке 16.

Также просмотреть или изменить полученную реляционную схему можно в Design Editor, построив Server Model Diagram для выбранной прикладной системы. Как это выглядит можно увидеть на рисунке 17.

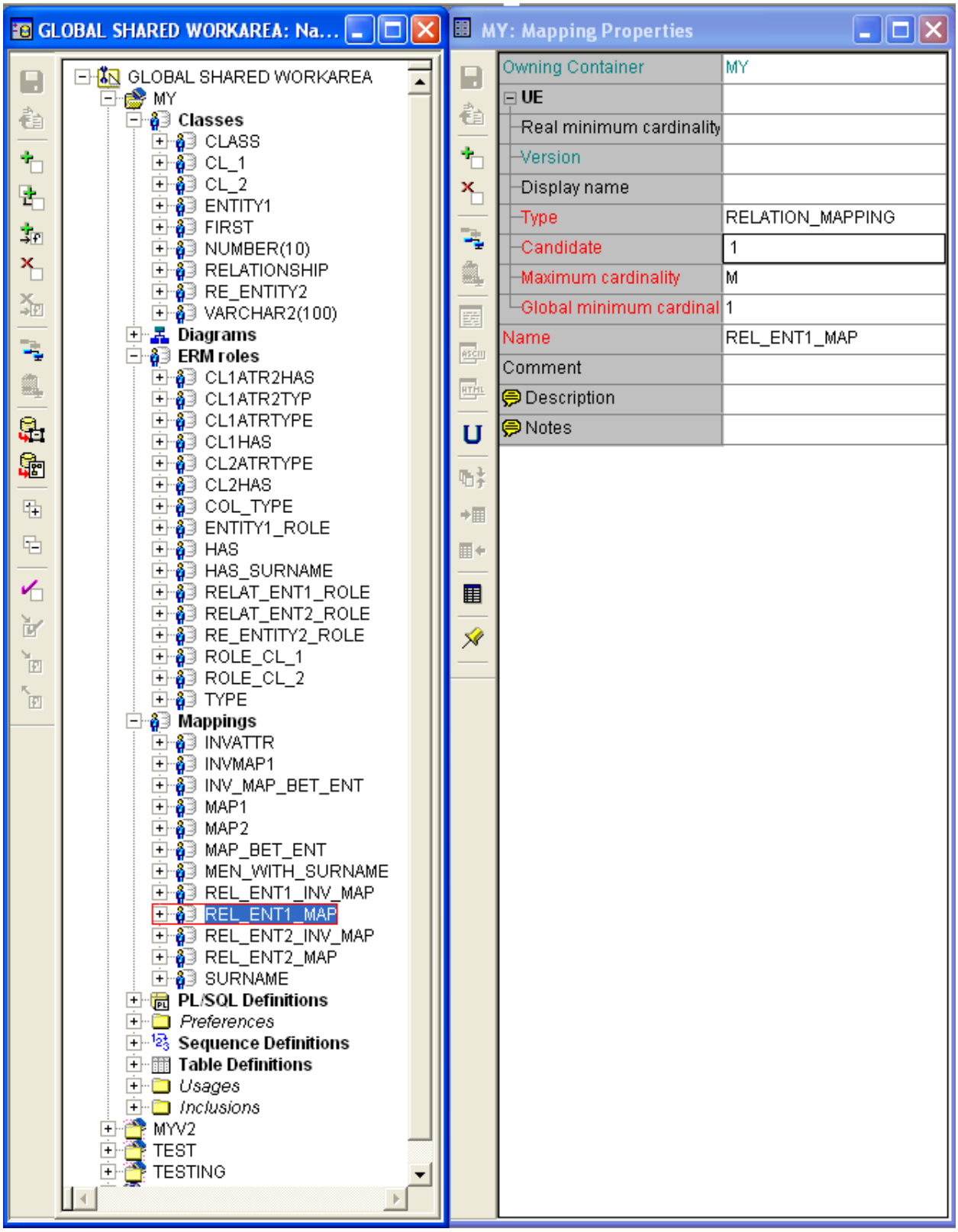


Рисунок 15 Исходная ERM-схема в RON

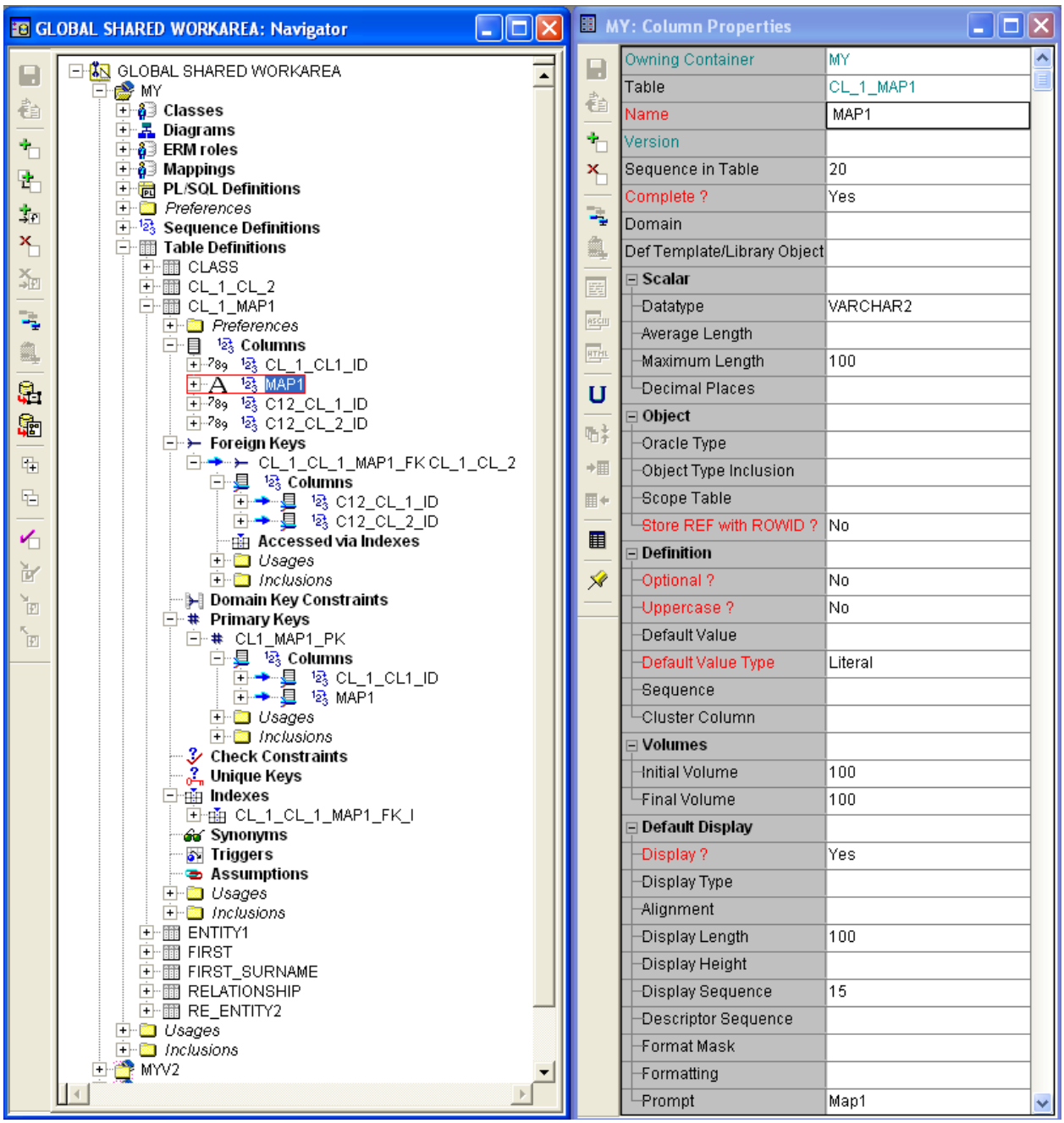


Рисунок 16 Итоговая реляционная схема в RON

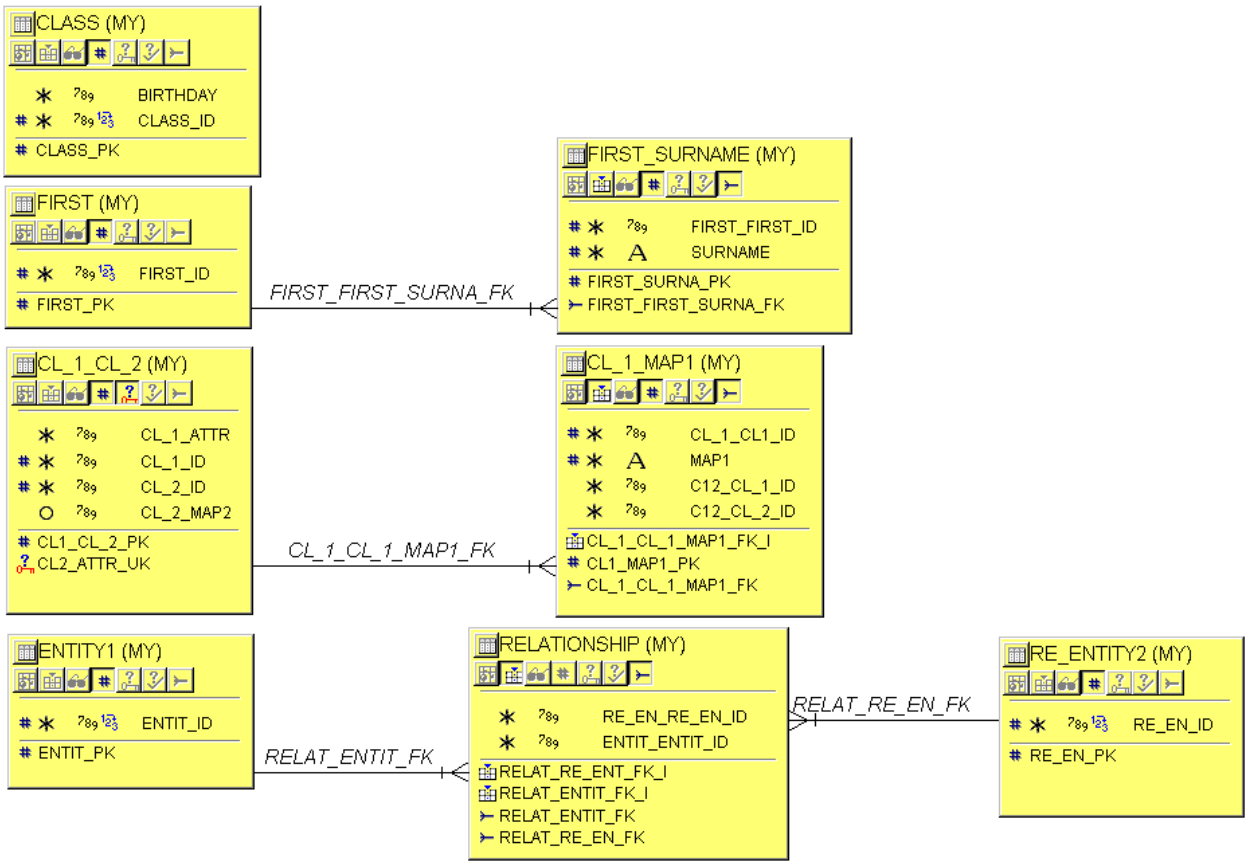


Рисунок 17 Server Model Diagram, построенная по итоговой реляционной схеме

Заключение

В ходе работы был реализован компонент для среды Oracle Designer, обеспечивающий преобразование «ERM-модель – реляционная модель». Компонент состоит из двух пакетов и набора объектных типов. Один пакет предназначен для работы с элементами реляционной модели. При помощи данного пакета можно изменять, удалять и добавлять все часто используемые элементы реляционной модели. Второй пакет, согласно правилам преобразования, реализует механизм преобразования «ERM-модель – реляционная модель». Таким образом, все поставленные задачи были решены и цель работы была достигнута.

В качестве планов на дальнейшую разработку данного компонента поставлены цели, связанные с разработкой графического редактора ERM-схем и расширением данного компонента, путём добавления в него новых механизмов и операций: реализация операций между отображениями и классами и реализация правил перехода от базовых понятий к производным.

Список используемых источников и литературы

1. Бабанов А.М. Формальная система теории семантически значимых отображений // Вестн. Том. Гос. ун-та. Математика. Кибернетика. Информатика. – 2006. – № 290. – С. 261–263.
2. Бабанов А. М. Семантическая модель «Сущность – Связь – Отображение» // Вестн. Том. Гос. ун-та. Управление, вычислительная техника и информатика. – 2007. – №1. – С. 77–91.
3. Бабанов А. М. Графическая нотация для модели «Сущность – Связь – Отображение» / Скачкова А. С. // Вестн. Том. Гос. ун-та. Управление, вычислительная техника и информатика. - 2008. - № 1(2). - С. - 97-105.
4. Бабанов А. М. Перспективы семантической методики проектирования БД, открывающиеся с применением ERM-модели данных // Информационные технологии и мат. моделирование (ИТММ - 2009): материалы 8-ой Всероссийской научно-практ. конф. с международным участием. - Томск. 2009. - Ч. 1. - С. 107-110.
5. Бабанов А. М. Базовые и производные понятия ERM-модели данных и их роль в процессе проектирования схем баз данных / Скачкова А. С. // Информационные технологии и математическое моделирование (ИТММ - 2010). Материалы 9-ой Всероссийской научно-практ. конф. с международным участием – Томск. 2010. - Ч. 2. - С. - 16-18.
6. Бабанов А. М. Базовые и производные структурные понятия ERM-модели данных и изоморфное отношение между ними // Вестн. Том. Гос. ун-та. Управление, вычислительная техника и информатика. - 2012. - № 4 (21). - С. 117-126.
7. Бабанов А. М. Синонимия в ERM-модели и проблемы обеспечения непротиворечивости и пополнения ERM-схем / Скачкова А. С. // Вестн. Том. Гос. ун-та. Управление, вычислительная техника и информатика. – 2012. – №3. – С. 121–128.

8. Гарсиа-Молина Гектор. Системы баз данных. Полный курс / Джеффри. Д. Ульман, Дженнифер Уидом.: Пёр с англ. – М.: Издательский дом Вильямс. 2003. – 1088 с.

9. Дейт К. Дж. Введение в системы баз данных. – 8-е издание. : Пёр. с англ. – М. : Издательский дом Вильямс. 2005. – 1328 с.

10. Крёнке Д. Теория и практика построения баз данных. 8-е изд. – СПб.: Питер, 2003. – 800 с.

11. Колетски П. Oracle Designer. Настольная книга пользователя / Дорси П. – М.: Лори, - 1999. – 592 с.

12. Фейерштейн С. Oracle PL/SQL. Для профессионалов/ Прибыл Б.- 6-е изд. — СПб.: Питер, 2015. — 1024 с.

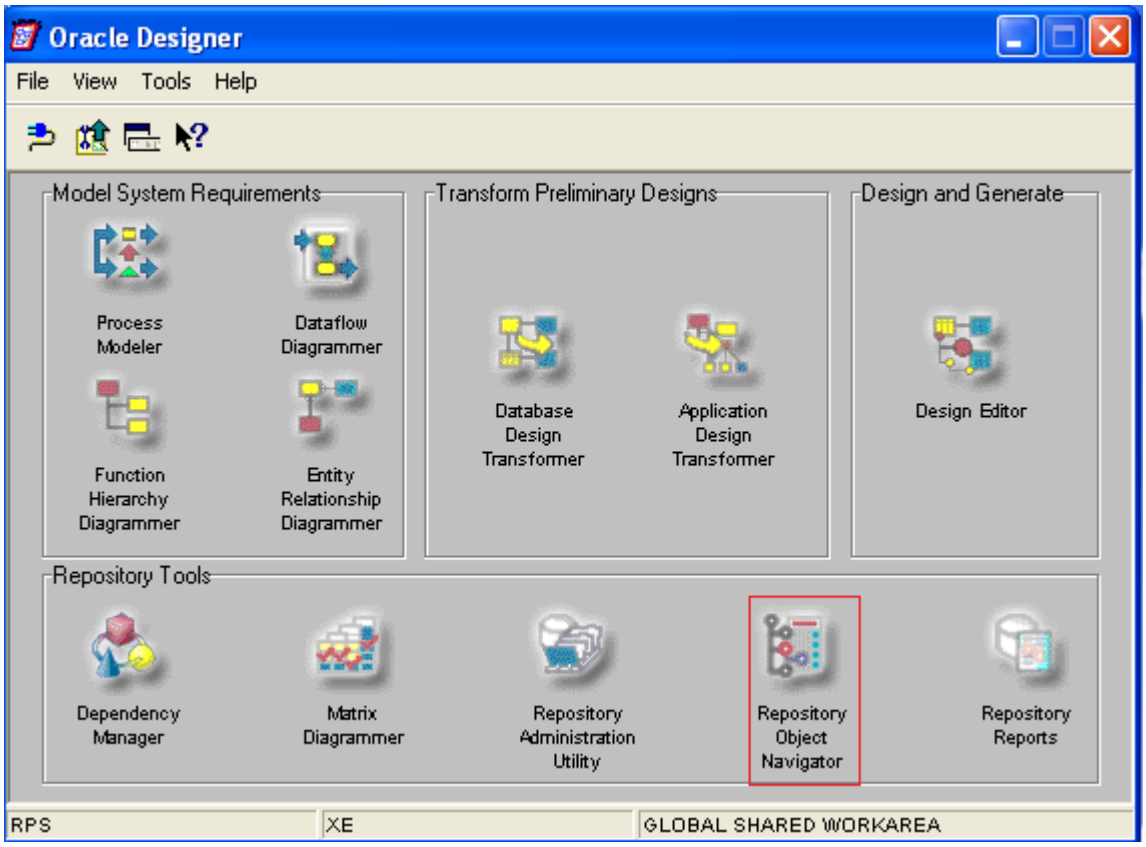
13. Oracle PL/SQL для профессионалов: практические решения / Коннор МакДональд, Хаим Кац, Бек Кристофер [и др.]; Пёр. с англ. - СПб.: ДиаСофтЮП, 2005. — 560 с.

14. ORACLE Help Center [Electronic Resource] // Oracle Database Online Documentation 11g Release 1 (11.1). URL: http://docs.oracle.com/cd/B28359_01/index.htm (access date: 04.2015)

Приложение А. Руководство пользователя

Создание ERM-схемы

Создание ERM-схемы осуществляется при помощи RON. Для этого необходимо запустить Oracle Designer и запустить Repository Object Navigator.

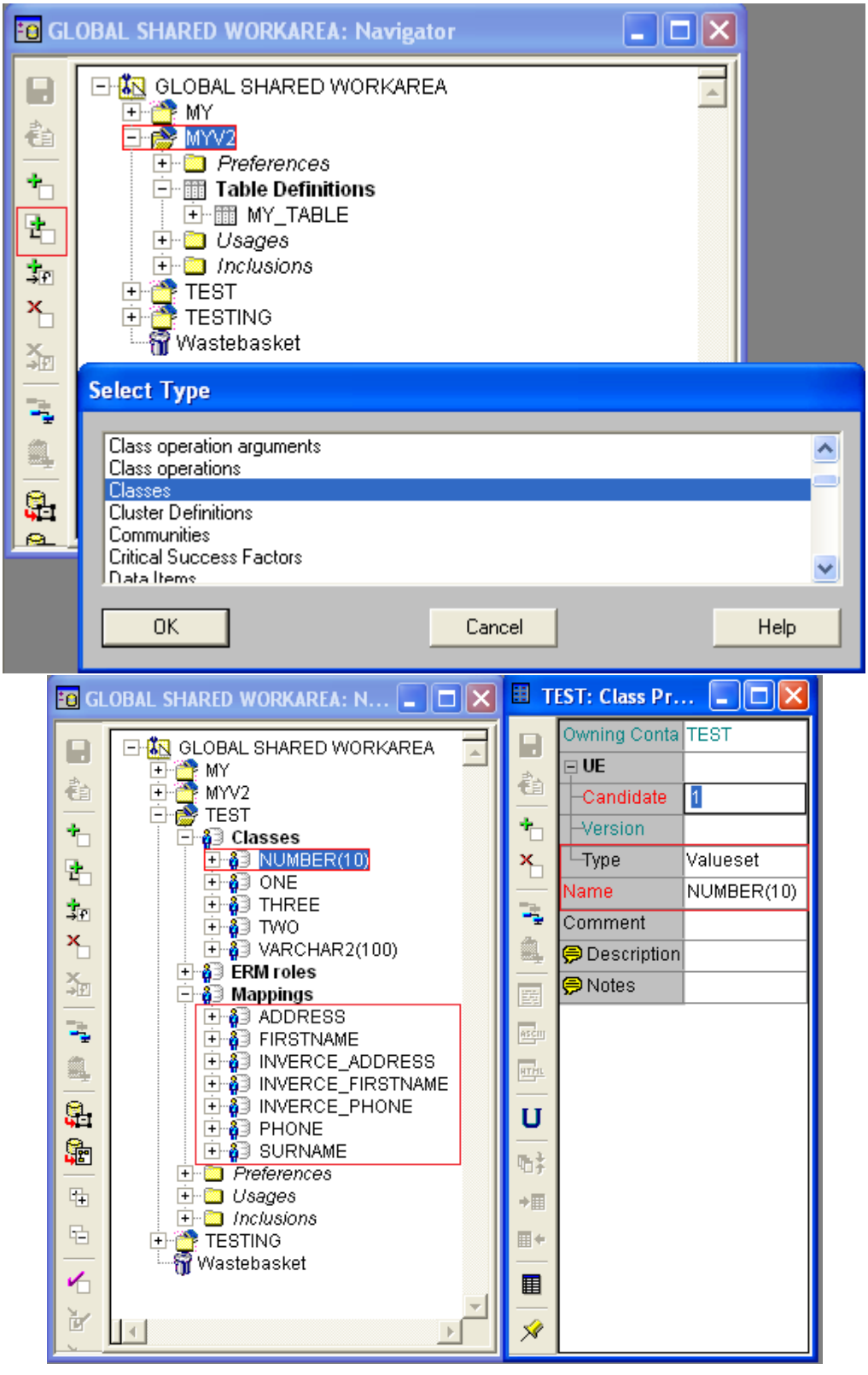


Далее необходимо выбрать прикладную систему (*Application System*) в рамках которой будет создана ERM-схема. Если созданной прикладной системы нет, то её можно создать при помощи контекстного меню ПКМ (Правая Кнопка Мыши) – Create Child, или нажать на кнопку Create as Child на панели инструментов.

После выбора прикладной системы, при помощи контекстного меню на выбранной прикладной системе ПКМ – Create Child (или нажатием на кнопку Create as Child на панели инструментов), можно создавать объекты ERM-модели.

Внимание!: Для классов обязательно необходимо указывать его тип (ENTITYSET, VALUESET, RELATIONSHIPSET). Для классов типа

VALUESET необходимо задавать имя по шаблону – «тип атрибута» («размерность»). В названии обратного атрибутного отображения должно содержаться имя отображения, для которого создаётся обратное. Например: для отображение SURNAME, обратное может выглядеть, как INVERCE_SURNAME.



Запуск механизма преобразования

Для запуска генератора, необходимо выполнить в любом, удобном для пользователя месте, следующий PL/SQL скрипт.

SQL | Output | Statistics | Monitor

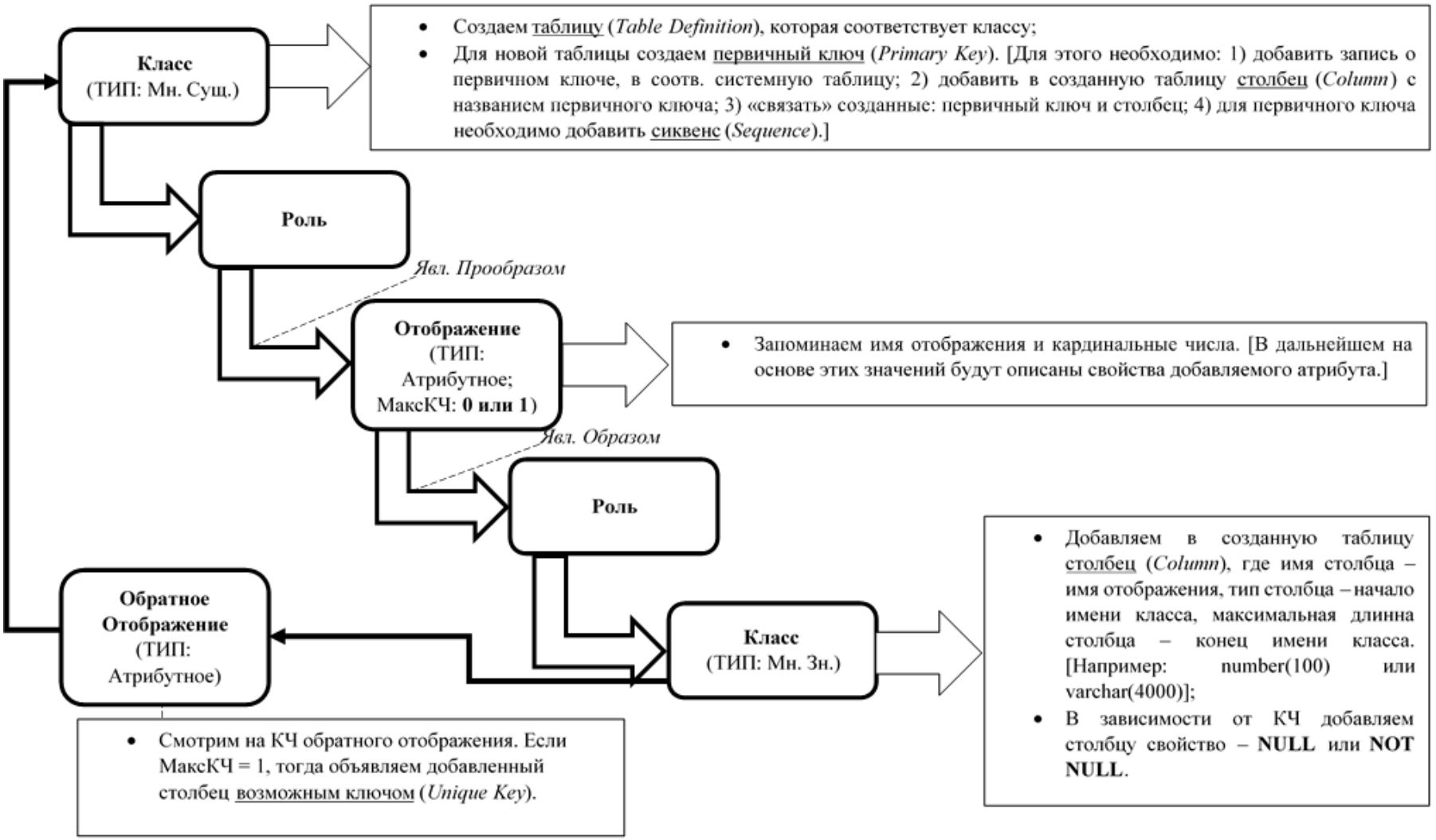
```
declare
 k number;
begin
   R_PKG_GENERATOR.R_PRC_START(pnFLAG => k,
                               psSCHEMA => 'TEST',
                               psTD_FLAG => 'DELETE',
                               psSEQ_FLAG => 'DELETE'
   );

   dbms_output.put_line(k);
end;
```

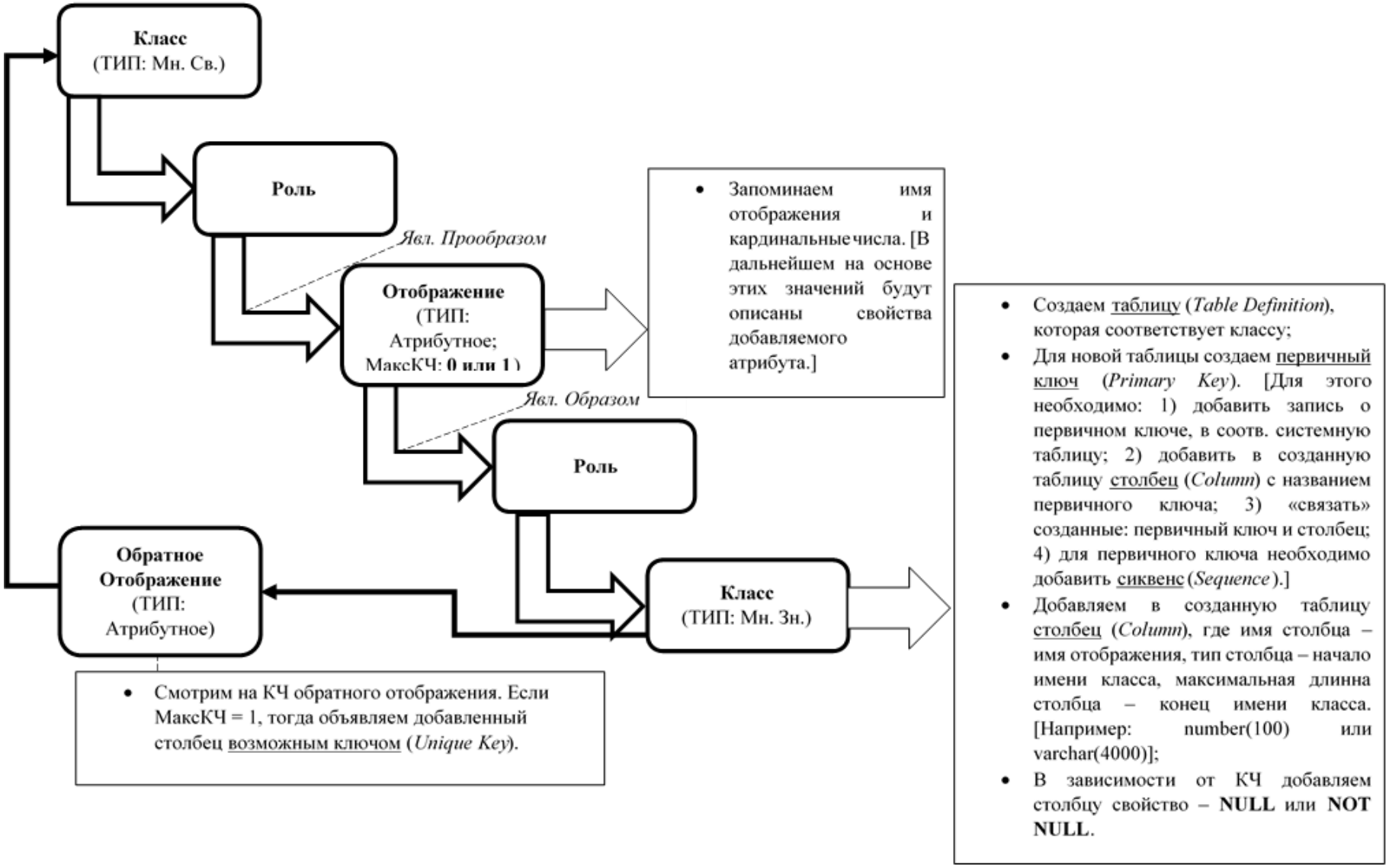
После окончания работы генератора, можно вернуться в RON и просмотреть полученную реляционную схему.

Приложение Б. Принцип работы генератора для правил 1-3

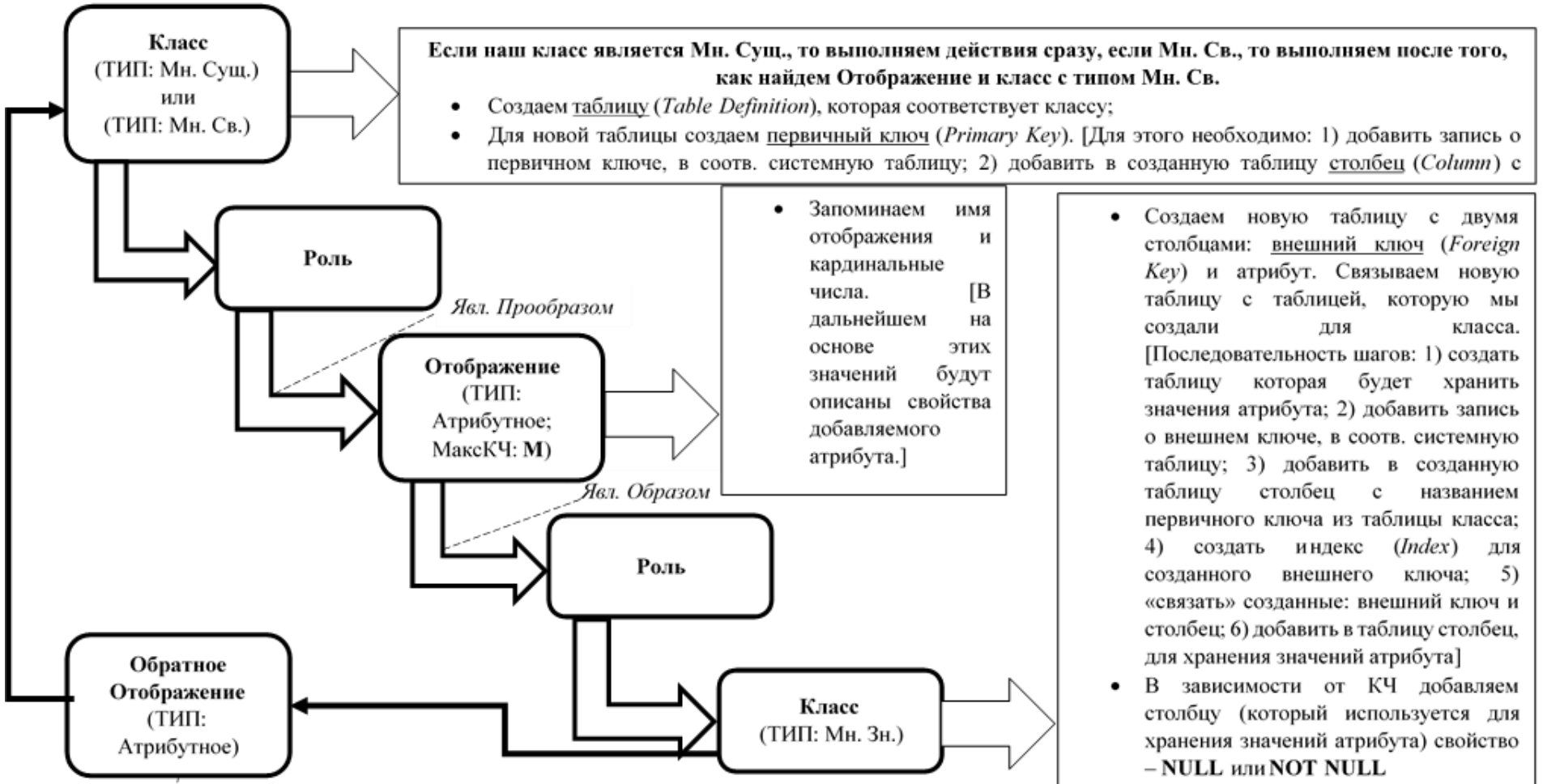
ПРАВИЛО 1



ПРАВИЛО 2


Класс
(ТИП: Мн. Св.)
Роль
Явл. Прообразом
Отображение
(ТИП:
Атрибутное;
МаксКЧ: 0 или 1)
Явл. Образом
Роль
Обратное
Отображение
(ТИП:
Атрибутное)
Класс
(ТИП: Мн. Зн.)
• Запоминаем имя отображения и кардинальные числа. [В дальнейшем на основе этих значений будут описаны свойства добавляемого атрибута.]
• Создаем таблицу (Table Definition), которая соответствует классу;
• Для новой таблицы создаем первичный ключ (Primary Key). [Для этого необходимо: 1) добавить запись о первичном ключе, в соотв. системную таблицу; 2) добавить в созданную таблицу столбец (Column) с названием первичного ключа; 3) «связать» созданные: первичный ключ и столбец; 4) для первичного ключа необходимо добавить сиквенс (Sequence).]
• Добавляем в созданную таблицу столбец (Column), где имя столбца – имя отображения, тип столбца – начало имени класса, максимальная длинна столбца – конец имени класса. [Например: number(100) или varchar(4000)];
• В зависимости от КЧ добавляем столбцу свойство – NULL или NOT NULL.
• Смотрим на КЧ обратного отображения. Если МаксКЧ = 1, тогда объявляем добавленный столбец возможным ключом (Unique Key).

ПРАВИЛО 3


Класс
(ТИП: Мн. Сущ.)
или
(ТИП: Мн. Св.)
Если наш класс является Мн. Сущ., то выполняем действия сразу, если Мн. Св., то выполняем после того, как найдем Отображение и класс с типом Мн. Св.
• Создаем таблицу (Table Definition), которая соответствует классу;
• Для новой таблицы создаем первичный ключ (Primary Key). [Для этого необходимо: 1) добавить запись о первичном ключе, в соотв. системную таблицу; 2) добавить в созданную таблицу столбец (Column) с
Роль
Явл. Прообразом
Отображение
(ТИП: Атрибутное; МаксКЧ: М)
• Запоминаем имя отображения и кардинальные числа. [В дальнейшем на основе этих значений будут описаны свойства добавляемого атрибута.]
Явл. Образом
Роль
Обратное Отображение
(ТИП: Атрибутное)
Класс
(ТИП: Мн. Зн.)
• Создаем новую таблицу с двумя столбцами: внешний ключ (Foreign Key) и атрибут. Связываем новую таблицу с таблицей, которую мы создали для класса. [Последовательность шагов: 1) создать таблицу которая будет хранить значения атрибута; 2) добавить запись о внешнем ключе, в соотв. системную таблицу; 3) добавить в созданную таблицу столбец с названием первичного ключа из таблицы класса; 4) создать индекс (Index) для созданного внешнего ключа; 5) «связать» созданные: внешний ключ и столбец; 6) добавить в таблицу столбец, для хранения значений атрибута]
• В зависимости от КЧ добавляем столбцу (который используется для хранения значений атрибута) свойство – NULL или NOT NULL
Смотрим на КЧ обратного отображения.
• Если МаксКЧ = 1, тогда объявляем столбец (используемый для хранения значений атрибута) первичным ключом.
• Если МаксКЧ = М, тогда объявляем оба столбца (внешний ключ и атрибут) первичным ключом.

Отчет о проверке № 1

ФИО: Petrov Evgeny
дата выгрузки: 20.05.2016 11:38:12
пользователь: chuvak.pw@yandex.ru / ID: 2211559

Информация о документе

№ документа: 29
Имя исходного файла: Дипломная работа_18.05.16_.docx
Размер текста: 1330 кБ
Тип документа: Не указано
Символов в тексте: 39409
Слов в тексте: 4869
Число предложений: 246

Информация об отчете

Дата: Отчет от 20.05.2016 11:38:12 - Последний готовый отчет
Комментарии: не указано
Оценка оригинальности: 76.39%
Заимствования: 23.61%
Цитирование: 0%

Оригинальность: 76.39%
Заимствования: 23.61%
Цитирование: 0%

Источники

Доля в тексте	Источник	Ссылка	Дата	Найдено в
12.02%	[1] http://www.inf.tsu.ru/library/DiplomaWorks/CompScience/2010/	http://inf.tsu.ru	раньше 2011 года	Модуль поиска Интернет
11.23%	[2] http://www.inf.tsu.ru/library/DiplomaWorks/CompScience/2012/	http://inf.tsu.ru	раньше 2011 года	Модуль поиска Интернет
2.79%	[3] Семантическая методика проектирования БД и ее перспективы, открывающиеся с применением ERM-модели данных	http://sun.tsu.ru	раньше 2011 года	Модуль поиска Интернет
2.67%	[4] Базы данных (Carolyn-Begg) (6/68)	https://scribd.com	08.01.2016	Модуль поиска Интернет
2.16%	[5] Синонимия в ERM-модели и проблемы обеспечения непротиворечивости и пополнения ERM-схем	http://sun.tsu.ru	16.11.2012	Модуль поиска Интернет
2.08%	[6] БАЗОВЫЕ И ПРОИЗВОДНЫЕ СТРУКТУРНЫЕ ПОНЯТИЯ ERM-МОДЕЛИ ДАННЫХ И ИЗОМОРФНОЕ ОТНОШЕНИЕ МЕЖДУ НИМИ - тема научной статьи по автоматике и вычислительной технике, читать бесплатно текст научно-исследовательской работы в электронной библиотеке КиберЛенинка	http://cyberleninka.ru	13.06.2013	Модуль поиска Интернет
1.98%	[7] http://sun.tsu.ru/mminfo/0194-17060/019417060.pdf	http://sun.tsu.ru	15.11.2012	Модуль поиска Интернет
1.75%	[8] Бабанов	http://inf.tsu.ru	раньше 2011 года	Модуль поиска Интернет
1.1%	[9] СИНОНИМИЯ ЭЛЕМЕНТОВ ERM-СХЕМ И ЕЕ ИСПОЛЬЗОВАНИЕ В МЕТОДИКЕ ERM-МОДЕЛИРОВАНИЯ ДЛЯ ГРАФИЧЕСКОЙ НОТАЦИИ - тема научной статьи по автоматике и вычислительной технике, читайте бесплатно текст научно-исследовательской работы в электронной библиотеке КиберЛени...	http://cyberleninka.ru	01.12.2014	Модуль поиска Интернет
0.88%	[10] 1.8. Инжиниринг и интегрирование бизнеса – часть 8 | Pandia.ru	http://pandia.ru	03.02.2014	Модуль поиска Интернет
0.87%	[11] http://www.inf.tsu.ru/library/DiplomaWorks/CompScience/2009/	http://inf.tsu.ru	раньше 2011 года	Модуль поиска Интернет
0.67%	[12] ДВА СОВРЕМЕННЫХ ПОДХОДА К СЕМАНТИЧЕСКОМУ МОДЕЛИРОВАНИЮ - ORM И ERMM - тема научной статьи по автоматике и вычислительной технике, читайте бесплатно текст научно-исследовательской работы в электронной библиотеке КиберЛенинка	http://cyberleninka.ru	01.12.2014	Модуль поиска Интернет
0.58%	[13] гарсиа-молина г. и др. системы баз данных. полный курс. 2003.djvu	http://inethub.olvi.net.ua	22.04.2014	Модуль поиска Интернет
0.51%	[14] дейт к.дж. - введение в системы баз данных - 2001.djvu	http://inethub.olvi.net.ua	раньше 2011 года	Модуль поиска Интернет
0.44%	[15] Домен (базы данных)	http://ru.wikipedia.org	30.11.2014	Модуль поиска Интернет
0.41%	[16] <![if !vml]><![endif]>	http://archive.nbuv.gov.ua	раньше 2011 года	Модуль поиска