**Серия «Математика»**
2014. Т. 10. С. 27—43

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 510.62 : 004.82

# Погружение реляционных баз данных в объектные онтологии: реализационные аспекты

И. А. Казаков
*Иркутский государственный университет*

А. А. Малых
*Иркутский государственный университет*

А. В. Манцивода
*Иркутский государственный университет*

**Аннотация.** Исследуется проблема погружения информации из баз данных в логическую среду обработки знаний. Представлены подходы к реализации метода погружения баз данных в онтологии через их моделирование с помощью объектных теорий. Данный метод ориентирован на единообразную работу с базами данных в больших распределенных информационных системах, включая облачные вычисления.

Во введении приводится актуальность проблемы. В первой части строится объектная теория базы данных [1]: дается базовый механизм и правила построения объектной модели по произвольной реляционной базе данных. Отдельным пунктом рассматривается моделирование внешних ключей базы данных при помощи средств объектных теорий.

Затем приводится реализация интерпретатора языка запросов Libretto на реляционных базах данных. Описывается общая архитектура системы Libretto. Подробно рассматриваются два подхода к реализации механизмов работы с базами данных на Libretto: первый способ заключается в реализации Libretto API в рамках СУБД; суть второго подхода заключается в подмене транслятора языка Libretto. Для реализации метода подмены транслятора Libretto, формально определяется подмножество языка Libretto, отвечающее за формирование запросной части языка $\mathcal{L}_{\text{sql}}$. Подробно рассматривается трансляция элементарных запросов, трансляция путей, трансляция инверсных свойств, трансляция вложенных запросов и трансляция предикатов.

Затем приводится сравнение эффективности реализаций двух данных подходов. Даются выводы и дальнейшие пути развития подхода.

**Ключевые слова:** онтологии, базы данных, объектные теории, дескриптивные логики, Libretto, большие распределенные системы.

## 1. Введение

Одной из актуальных проблем является задача вовлечения информации, накопленной в базах данных, в системы логической обработки. Поскольку базы данных и логические системы основаны на очень разных формализмах, эта задача не имеет простого решения [7]. Нами развивается подход, основанный на погружении баз данных в специальные дескриптивные логики. Эта работа проводится в рамках проекта Libretto [4; 5] — языка программирования, ориентированного на разработку больших распределенных систем, функционирующих в неоднородных пространствах, например, в рамках облачных вычислений. Этот проект основан на идее единого информационного пространства, когда разнородные источники и форматы информации погружаются в единую логико-информационную среду, позволяющую разрабатывать большие распределенные системы.

В статьях [1; 2] нами было аксиоматизировано понятие реляционной базы данных в терминах объектных дескриптивных логик на основе понятия объектной теории [3]. В данной работе исследуются два метода реализации этих логико-информационных конструкций в рамках системы Libretto.

## 2. Объектные теории баз данных

Объектные теории баз данных были введены в [1]. Их основная идея заключается в описании реляционных БД средствами объектно-ориентированной дескриптивной логики [3]. Для построения объектной теории для некоторой базы данных $\mathbb{DB}$ необходимо отобразить ее компоненты в структуры онтологии. Для этого вводится оператор $E \overset{\mathtt{tr}}{\to} \mathtt{e}$, переводящий сущность $E$ базы данных в соответствующую ей сущность $\mathtt{e}$ онтологии: $\mathtt{tr}(E) = \mathtt{e}$.

### 2.1. Моделирование сущностей баз данных

Каждому табличному отношению

$$\mathbf{r}^i = \langle R^i, body_{\mathbf{r}^i}, header_{\mathbf{r}^i} \rangle,$$

где $R^i$ — имя отношения, $body_{\mathbf{r}^i}$ — его тело (множество кортежей), а $header_{\mathbf{r}^i}$ — множество заголовков, сопоставляется имя концепта $C^i$:

$$\mathtt{tr}(\mathbf{r}^i) = C^i$$

Атрибуты (столбцы) таблиц моделируются с помощью т-свойств. С каждым атрибутом базы данных $A^i_j$ отношения $\mathbf{r}^i$ сопоставляем имя

концепта $B^i_j$ и имя т-свойства $P^i_j$, $i \in \overline{1,n}$; $j \in \overline{1,k}$ (совпадающим именам атрибутов из разных табличных отношений должны соответствовать разные имена концептов и т-свойств):

$$\mathtt{tr}(A^i_j) = \{B^i_j, P^i_j\}$$

## 2.2. Объектная теория базы данных

Семантика новых сущностей описывается с помощью аксиом объектных теорий (см. [3]). Семантика концепта $B^i_j$ и т-свойства $P^i_j$ определена с помощью аксиомы домена:

$$\exists P^i_j \sqsubseteq B^i_j, j \in \overline{1,k} \tag{2.1}$$

и ранга

$$\exists P^{i-}_j \sqsubseteq D^i_j, j \in \overline{1,k} \tag{2.2}$$

где $D^i_j$ — область значений для атрибута $A^i_j$. Поскольку в ячейке таблицы не может быть более одного значения, то $P^i_j$ может принимать либо ни одного, либо одно значение (отсутствие значений моделирует NULL):

$$B^i_j \sqsubseteq \ \leq 1.P^i_j, j \in \overline{1,k} \tag{2.3}$$

Семантика табличного концепта $C^i$ описывается через наследование от концептов-столбцов:

$$\begin{array}{c} C^i \sqsubseteq B^i_1 \\ \cdots \\ C^i \sqsubseteq B^i_k \end{array} \tag{2.4}$$

Объекты, которые моделируют конкретные данные, представленные в $\mathbb{DB}$ в виде кортежей (строк таблиц), описываются следующим образом. Пусть

$$body_{\mathbf{r}^i} = \{t^i_1, \ldots, t^i_l\}$$

– набор кортежей отношения $\mathbf{r}^i$. Для каждого $t^i_j$ водим новую константу $o^i_j$, определив

$$o^i_j = \mathtt{tr}(t^i_j).$$

Вводим для $o^i_j$ аксиому принадлежности табличному концепту

$$C^i(o^i_j), \tag{2.5}$$

где $C^i = \mathtt{tr}(\mathbf{r}^i)$, и аксиомы значений т-свойств

$$P^i_s(o^i_j, t^i_j(A^i_s)) \tag{2.6}$$

для всех атрибутов $A_s^i \in sign(\mathbf{r}^i)$ таких, что $t_j^i(A_s^i) \neq \texttt{NULL}$ и $P_s^i \in \texttt{tr}(A_s^i)$. Этот момент является существенным: в объектных теориях отсутствие значения свойства моделируется не с помощью явной константы NULL, а через отсутствие соответствующей аксиомы.

Определим для отношения $\mathbf{r}^i$ множество аксиом $\mathbb{O}^i$, состоящее из $k$ аксиом вида (2.1)–(2.4), а также наборов аксиом вида (2.5) и (2.6) для каждого кортежа $t_j^i$ и каждого непустого значения $t_j^i(A_t^i)$, соответственно.

**Определение 1.** *Множество аксиом*

$$\mathbb{O}_{\mathbb{DB}} = \bigcup_{i=1}^{n} \mathbb{O}^i$$

*называется* объектной теорией *базы данных* $\mathbb{DB}$. *Будем говорить, что сущности объектной теории (концепты* $C_i$, *объекты* $o_i$ *и т. д.)* моделируют *в* $\mathbb{O}_{\mathbb{DB}}$ *соответствующие сущности базы* $\mathbb{DB}$ *(отношения* $\mathbf{r}_i$, *строки* $t_i$ *и т.д.).*

В [1] доказывается, что $\mathbb{O}_{\mathbb{DB}}$ является объектной теорией [3] в словаре $\mathcal{W} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$, где

$$\mathcal{C} = \texttt{tr}(\{\mathbf{r}^1, \ldots, \mathbf{r}^n\}) \cup \texttt{tr}(sign(\mathbb{DB}))_{\mathcal{C}} = \{C^1, \ldots, C^n\} \cup \bigcup_{i=1}^{n} \{B_1^i, \ldots, B_{k^i}^i\}$$

$$\mathcal{R} = \emptyset$$

$$\mathcal{P} = \texttt{tr}(sign(\mathbb{DB}))_{\mathcal{P}} = \bigcup_{i=1}^{n} \{P_1^i, \ldots, P_{k^i}^i\}$$

$$\mathcal{O} = \bigcup_{i=1}^{n} \texttt{tr}(body_{\mathbf{r}^i}) = \bigcup_{i=1}^{n} \{o_1^i, ..., o_{l^i}^i\}$$

Этот результат позволяет использовать объектные теории баз данных для моделирования работы с БД в рамках логической среды, не прибегая к сложным гибридным формализмам (см. [7]). Можно сказать, что данный механизм превращает базы данных в теории некоторой простой дескриптивной логики, обеспечивая тем самым единое логическое пространство. В следующем пункте показывается, каким образом можно применять данные теоретические выкладки в информационных системах.

### 2.3. Внешние ключи и о-свойства

Прежде чем перейти к реализационным аспектам, нам необходимо расширить определение $\mathbb{O}_{\mathbb{DB}}$. Речь идет о моделировании связей между кортежами (строками таблиц), которые реализуются в базах данных с помощью внешних ключей — т. е. групп значений, в совокупности уникально идентифицирующих, неформально говоря, кортежи базы данных. В объектных теориях внешний ключ соответствует о-свойству, которое связывает объекты с другими объектами. Для простоты сформулируем определение для случая, когда первичный ключ кортежа, на который ссылается данный кортеж, состоит из одного атрибута (столбца). Общее определение формулируется аналогично. Обозначим первичный ключ таблицы $\mathbf{r}^i$ через $Id^i$. Пусть кортежи таблицы $\mathbf{r}^j$ содержат внешний ключ $FK^i$, ссылающийся на элементы таблицы $\mathbf{r}^i$ через первичный ключ $Id^i$. Тогда для каждого кортежа $t^i_k \in body_{\mathbf{r}^i}$ и $t^j_l \in body_{\mathbf{r}^j}$ добавляем аксиому

$$Q(\mathtt{tr}(t^j_l), \mathtt{tr}(t^i_k)),$$

где $Q$ – имя о-свойства такого, что $Q = \mathtt{tr}(FK^i)$, и кроме того,

$$t^j_l(FK^i) = t^i_k(Id^i).$$

Полагаем также $\mathtt{tr}(t^j_l(FK^i)) = \mathtt{tr}(t^i_l)$. Для самого о-свойства $Q$ добавляются аксиомы домена:

$$\exists Q \sqsubseteq \mathtt{tr}(\mathbf{r}^j) \tag{2.7}$$

и ранга

$$\exists Q^- \sqsubseteq \mathtt{tr}(\mathbf{r}^i) \tag{2.8}$$

С этого момента считаем, что для каждого внешнего ключа из базы данных в теории $\mathbb{O}_{\mathbb{DB}}$ содержатся аксиомы для соответствующего ему о-свойства $Q$. Легко проверяется, что в расширенном варианте, включающем о-свойства, $\mathbb{O}_{\mathbb{DB}}$ остается объектной теорией в смысле [1].

## 3. Реализация интерпретатора языка Libretto на базах данных

В данном разделе демонстрируется, как формализм, представленный выше, может быть реализован с помощью языка Libretto.

### 3.1. Общая архитектура системы Libretto

Архитектура системы Libretto позволяет реализовать различные подходы к работе с БД. Ее особенностью является то, что система типов данных языка изоморфна логике $\mathcal{OODL}$. С точки зрения реализации, Libretto-интерпретатор работает с объектными структурами

данных не напрямую, а через абстрактный программный интерфейс — Libretto API, являющийся информационным аналогом логики $\mathcal{OODL}$. Транслятор Libretto переводит Libretto-код в код абстрактной Libretto-машины, работающей на основе Libretto API. Разные реализации API позволяют использовать Libretto на разных платформах и для разных целей в рамках разработки больших распределенных систем.

Основные принципы взаимодействия с базами данных на Libretto будут проиллюстрированы на примере небольшой, но реально используемой базы, служащей для накопления новостей из муниципальных образований ряда регионов. Эта база состоит из 20 таблиц, содержащих суммарно 16 768 записей, в том числе около 14 000 новостей (см. рис 1).

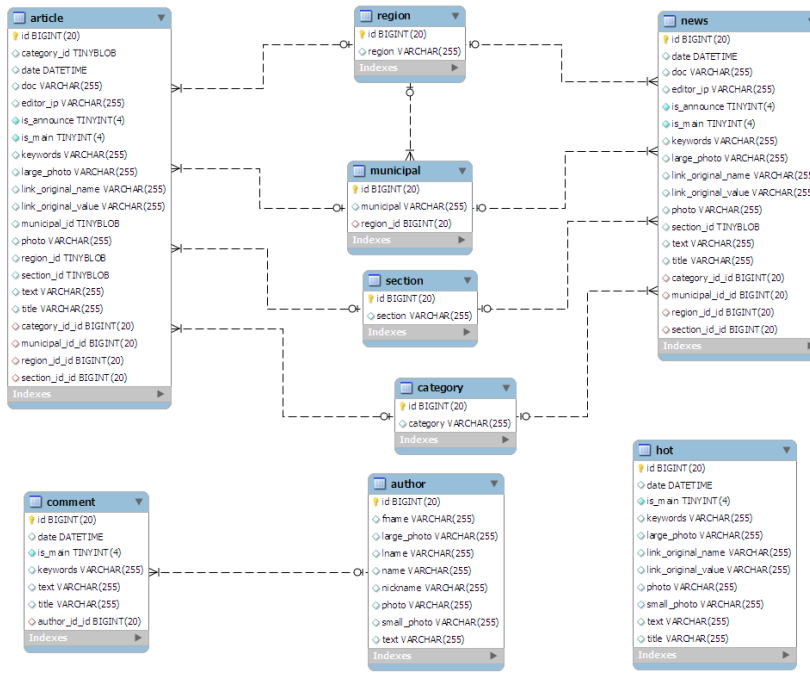


*Рис. 1.* Структура базы муниципальных новостей

## 3.2. Методы работы с базами данных на Libretto

Нами были разработаны и исследованы два подхода к реализации механизмов работы с базами данных на Libretto. Первый способ заключается в реализации Libretto API в рамках СУБД, что превращает Libretto напрямую в язык запросов к базам. Отображение из Libretto в базу данных через интерфейс Libretto API позволяет как получать информацию из базы данных (в объектном виде), так и корректировать

саму БД, продолжая работать в объектном стиле, поддерживаемым Libretto. Назовем такой способ подменой хранилища.

Основная идея второго способа — это подмена транслятора языка Libretto. Вместо абстрактной Libretto-машины новый транслятор переводит Libretto-код непосредственно в SQL. В связи с ограничениями, обусловленными различием архитектур Libretto и SQL, таким образом может быть оттранслирован код, написанный на ограниченном подмножестве Libretto. Если запрос выходит за рамки этого подмножества, то нужно использовать первый способ — подмены хранилища.

## 3.3. Отображение структуры БД в Libretto

Оба эти способа базируются на отображении базы данных $\mathbb{DB}$ в соответствующую ей объектную теорию $\mathbb{O}_{\mathbb{DB}}$, поскольку прежде чем начать работу с базой данных, необходимо предоставить Libretto ее объектную модель. Нами рассматривались три способа генерирования такой модели по базе данных — автоматический, ручной, смешанный.

При автоматическом построении структурной онтологии она генерируется как результат анализа базы данных специализированным модулем системы. Использование служебной информации о структуре базы данных, о ее первичных и внешних ключах играет в построении модели большую роль. Метод автоматического построения дешев с точки зрения затраты ресурсов, однако качество сгенерированной модели зачастую получается недостаточным и сильно зависящим от качества первичной БД.

При ручном построении структурной онтологии вся работа проводится человеком. Он может настроить структуру онтологии под предметную область базы данных. Качество моделей, построенных человеком, в общем случае выше, чем при первом способе. Недостатком этого подхода являются большие трудовые затраты.

Смешанное построение — построение структурной онтологии в автоматическом режиме с последующим ручным редактированием. Этот гибридный подход, видимо, является оптимальным, поскольку позволяет регулировать соотношение качества модели и трудовых затрат.

На рис. 2 представлена упрощенная структура объектной теории $\mathbb{O}_{\mathbb{DB}}$ базы муниципальных новостей, построенной смешанным способом. В эту структуру кроме основных понятий базы данных введен ряд обобщающих понятий, например, класс Info (обобщение понятий `Article` и `News`).

Имея такую модель, можно формировать запросы к ней. В общем случае запрос на Libretto имеет форму пути, состоящего из нескольких шагов:

```
start/step1/step2/.../stepN
```

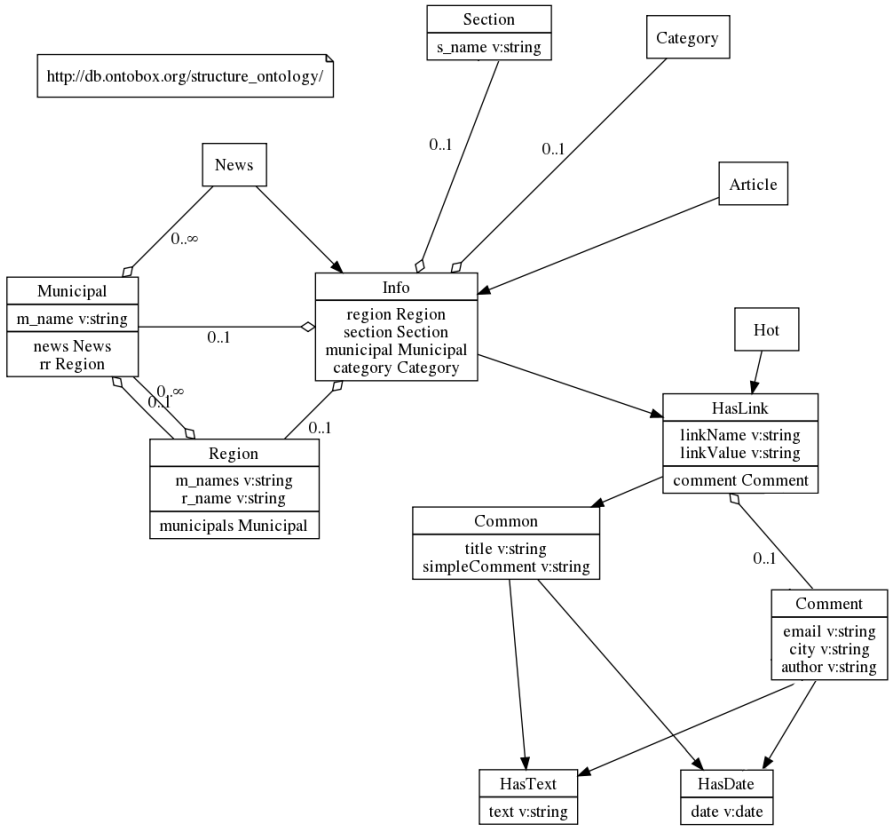


*Рис. 2.* Структура объектной теории базы муниципальных новостей

Нулевой шаг — `start` — задает начальный контекст, от которого потом производятся все дальнейшие переходы. Промежуточные шаги `step1`, `step2`, ... задают последовательность переходов. Последний шаг `stepN` определяет значение всего запроса. Если интерпретировать данный путь как запрос к базе данных, то на первом шаге мы получаем один или несколько объектов, которые интерпретируются как строки таблицы. Переход от шага к шагу интерпретируется либо как переход между таблицами через внешние ключи, либо как фильтр, проверяющий контекстные значения. Последний шаг обозначает получаемое значение, которое в SQL-запросе находится в блоке `SELECT`. Внутри каждого шага может присутствовать предикат в квадратных скобках, фильтрующий контекстные значения. Например, следующий запрос на Libretto

$$\texttt{News}[\texttt{municipal/m\_name} == \text{"Город Иркутск"}]/\texttt{text}$$

выберет тексты всех новостей, относящихся к муниципальному образованию "город Иркутск".

Рассмотрим теперь более подробно каждый из двух подходов работы с базами данных на Libretto.

### 3.4. Метод подмены хранилища

Для реализации этого метода необходимо реализовать Libretto API на базах данных. Функции API можно разделить на два типа:

1) функции, относящиеся к структуре онтологии, такие как получение информации о классах, т- или о-ойствах;

2) функции, относящиеся к объектам, такие как получение объектов, значений их т- или о-свойств.

Для реализации функций первого типа доступ к базе данных не нужен, поскольку вся необходимая для них информация содержится в $\mathbb{O}_{\mathbb{DB}}$. Необходимо реализовать только функции второго типа, манипулирующие конкретными объектами. Они включают в себя:
- получение списка всех объектов класса;
- получение объекта по уникальному имени (URI);
- получение значение т-свойства у объекта;
- получение значения о-свойства у объекта;
- получение всех объектов, ссылающихся на данный объект через определенное о-свойство.

Эти функции простые, низкоуровневые, поэтому их реализация состоит в трансляции в небольшие сегменты SQL.

Рассмотрим, например, реализацию взятия значений свойств объекта (операцию «слэш» '/'). Существенной особенности Libretto является единообразная работа со скалярными и векторными значениями. Например, получение как единственного значения муниципалитета, описываемого новостью, так и получение множества муниципалитетов, входящих в регион, будет выглядеть очень похоже:

```
&news444/municipal    (муниципалитет новости № 444)
&region2/municipals   (муниципалитеты региона № 2)
```

(в Libretto имена объектов начинаются с амперсанда '&'). Однако в SQL эти два запроса будут транслироваться по-разному. Это трансляция первого запроса:

Libretto: `&news444/municipal`

SQL:

```
SELECT MUNICIPAL FROM NEWS WHERE ID = "444"
```

Здесь `MUNICIPAL`, `tr(MUNICIPAL)` = `municipal`, — столбец таблицы, соответствующий т-свойству `municipal`, `NEWS` — это таблица новостей, а

&news444 – объект конкретной новости с первичным ключом ID = 444, т. е. tr(444) = &news444.

Во втором случае нам необходимо реализовать отношение «один-ко-многим»:

Libretto: &region2/municipals

SQL:

```
SELECT MUICIPAL.ID
FROM MUNICIPAL, REGION
WHERE REGION.ID=MUNICIPAL.REGION_ID
    AND REGION.ID="2"
```

Здесь MUNICIPAL — это таблица муниципалитетов, содержащая привязку к регионам, REGION — таблица регионов. Ключ "2" такой, что tr(2) = &region2. Условие REGION.ID=MUNICIPAL.REGION_ID задает связь между таблицами.

Выбор варианта трансляции проводится на основе информации о кардинальности соответствующих свойств в модели. На рис. 2 видно, что свойство municipal имеет кардинальность 0..1, а municipals – $0..\infty$.

Приведенные примеры касались работы с объектами по ключам, что соответствует в Libretto о-свойствам. Получение конкретных значений в таблицах (что соответствует т-свойствам в Libretto) проводится аналогично. Например, так можно получить текст новости:

Libretto: &news444/text

SQL:

```
SELECT TEXT FROM NEWS WHERE ID = "444"
```

Здесь tr(TEXT) = text.

Подобным образом реализуются запросы для других базовых операций, таких как получение объекта по имени, получение владельца объекта по о-свойству.

Реализованный таким образом Libretto API обеспечивает работу полнофункционального интерпретатора языка на базах данных. К сожалению, в этом подходе имеются серьезные проблемы с эффективностью, поскольку для выполнения даже самого простого запроса на Libretto происходит много обращений к базе данных. Кроме того, низкоуровневые операции Libretto API не позволяют применять глобальные оптимизации.

### 3.5. Метод подмены транслятора

Рассмотрим второй вариант интерпретации Libretto-запросов в базах данных — через их явную трансляцию в SQL-запрос. Определим подмножество Libretto, на котором будет работать транслятор:

**Определение 2.** *Определим множества Libretto-запросов* $\mathcal{L}_{sql}$ *и предикатов* $\mathcal{P}_{sql}$ *как минимальные множества, удовлетворяющие следующим условиям:*

1) *если* $N$ *— имя класса или свойства, то* $N \in \mathcal{L}_{sql}$ *и* $N$ *является шагом.*

2) *если* $N$ *— имя свойства, то инверсное свойство* $\sim N \in \mathcal{L}_{sql}$ *и* $\sim N$ *является шагом.*

3) *если* $d \in D$, *где* $D$ *— тип данных (целочисленный, строковый и т. д.), то* $d \in \mathcal{L}_{sql}$ *и* $d$ *является шагом.*

4) *если* $S \in \mathcal{L}_{sql}$ *является шагом и* $R \in \mathcal{P}_{sql}$, *то* $S[R] \in \mathcal{L}_{sql}$ *и также является шагом.*

5) *если* $P, S \in \mathcal{L}_{sql}$, *причем* $S$ *является шагом, то* $P/S \in \mathcal{L}_{sql}$.

6) *если* $P_1, P_2 \in \mathcal{L}_{sql}$, *то* $P_1\, rel\, P_2 \in \mathcal{P}_{sql}$, $rel \in \{==, !=, <, >, <=, >=\}$.

7) *если* $R_1, R_2 \in \mathcal{P}_{sql}$, *то*

$$not\, R_1,\ R_1\, and\, R_2,\ R_1\, or\, R_2 \in \mathcal{P}_{sql}$$

Легко проверить, что запрос

$$\texttt{News}[\texttt{municipal/m\_name} == \text{"Город Иркутск"}]/\texttt{text}$$

принадлежит $\mathcal{L}_{\text{sql}}$, а

$$\texttt{municipal/m\_name} == \text{"Город Иркутск"} \in \mathcal{P}_{\text{sql}}$$

Трансляция в SQL производится на основе набора правил и шаблонов. Проиллюстрируем некоторые из них.

3.5.1. *Трансляция элементарных запросов*

Элементарные запросы (например, получение значения свойства объекта) транслируются так же, как и в методе, представленном в предыдущем параграфе, поскольку элементарные запросы соответствуют элементарным операциям Libretto API.

3.5.2. *Трансляция путей*

Рассмотрим случай, когда транслируемый запрос является путем, состоящим из цепочки шагов. Пусть, например, необходимо получить тексты всех муниципальных новостей региона № 2:

Libretto: &region2/municipals/news/text

SQL:

```
SELECT NEWS.TEXT
FROM NEWS, MUNICIPAL, REGION
WHERE MUNICIPAL.REGION_ID = REGION.ID
    AND NEWS.MUNICIPAL_ID = MUNICIPAL.ID
    AND REGION.ID = 2
```

В этом случае цепочка шагов в Libretto–запросе соответствует цепочке переходов по ключам в таблицах.

3.5.3. *Трансляция инверсных свойств*

В Libretto допустимо инверсное использование свойств, которое позволяет по сущности (объекту или значению) получить объекты, имеющие ссылки на данную сущность. Например, используя инверсию, можно запросить названия всех информационных материалов, относящихся к муниципалитету номер 11:

Libretto: &municipal11/ $\sim$ municipal/title

SQL:

```
SELECT ARTICLE.TITLE
FROM ARTICLE, MUNICIPAL
WHERE
    ARTICLE.MUNICIPAL_ID = MUNICIPAL.ID
    AND MUNICIPAL.ID = 11
UNION ALL
SELECT NEWS.TITLE
FROM NEWS, MUNICIPAL
WHERE
    NEWS.MUNICIPAL_ID = MUNICIPAL.ID
    AND MUNICIPAL.ID = 11
```

Областями определения и значения для `municipal` являются классы `Info` и `Municipal`, соответственно. В $\sim$ `municipal` они меняются местами. Тонкость здесь заключается в том, что поскольку область определения `municipal` — абстрактный класс `Info`, от которого наследуют сразу два базовых класса `Article` и `News`, а структура базы данных плоская и не содержит объединяющего понятия `Info`, то на уровне SQL приходится объединять два отдельных запроса — для `Article` и для `News`. Этот пример иллюстрирует явное использование объектной иерархии предметной области, а также то, насколько компактнее благодаря этому можно формулировать запросы.

3.5.4. *Вложенные запросы*

Рассмотрим трансляцию вложенных запросов — когда в Libretto-запросе одна и та же таблица, ассоциированная с какой-либо сущностью онтологии, встречается более одного раза. В этом случае необходимо сформировать SQL-подзапрос, и этот подзапрос должен быть вложен в SQL-запрос в блоке условий при связывании таблиц.

Рассмотрим Libretto-запрос, который можно использовать для проверки корректности базы (результатом этого запроса должно быть только имя региона 2):

Libretto: &region2/municipals/news/region/r_name

SQL:

```
SELECT REGION.REGION
FROM REGION, NEWS
WHERE NEWS.REGION_ID = REGION.ID
    AND NEWS.ID IN (
      SELECT NEWS.ID
      FROM NEWS, MUNICIPAL, REGION
      WHERE MUNICIPAL.REGION_ID = REGION.ID
        AND NEWS.MUNICIPALl_ID = MUNICIPAL.ID
        AND REGION.ID = 2)
```

По структуре Libretto-запроса нам необходимо от региона номер 2 перейти ко всем его муниципалитетам, затем для каждого муниципалитета выбрать его новости, и затем от новостей вновь перейти к регионам. Объект `&region2` и о-свойство `region` ассоциированы с одной и той же таблицей — `REGION`. Поэтому, начиная с о-свойства `REGION` строим новый SQL-запрос, а результат уже имеющегося используем как контекст, в рамках которого формируется новый.

3.5.5. *Предикаты*

При трансляции предикатов (конструкций языка Libretto, фильтрующих контекстные значения) используются те же правила, что и при трансляции путей:

Libretto: `News[municipal/m_name ==" Город Иркутск"]/text`

SQL:

```
SELECT TEXT
FROM NEWS, MUNICIPAL
WHERE NEWS.MUNICIPAL_ID = MINICIPAL.ID
      AND MUNICIPAL.MUNICIPAL = 'город Иркутск'
```

Здесь `MUNICIPAL`, `tr(MUNCIPAL_ID) = municipal`, — столбец таблицы, соответствующий о-свойству `municipal`.

### 3.6. Сравнение реализаций

Таблица 1

Сравнение реализаций (миллисекунды)

| Libretto-запрос | API | TRN |
|---|---|---|
| `&news444/municipal/m_name` | 0 | 0 |
| `&news444/municipal/rr/r_name` | 16 | 0 |
| `&region2/municipals/news/text` | 515 | 32 |
| `&news444/region/municipals/m_name` | 16 | 15 |
| `&region2/ rr/m_name` | 0 | 0 |
| `&municipal22/ municipals/r_name` | 15 | 0 |
| `&news444/municipal/ municipals/r_name` | 0 | 0 |
| `News/text` | 3500 | 1422 |
| `News/municipal/m_name` | 5047 | 31 |
| `News/municipal/rr/r_name` | 5875 | 0 |
| `Region/municipals/news/text` | 1203 | 203 |
| `Municipal/news/section/s_name` | 4250 | 172 |
| `News/region/municipals/m_name` | 58813 | 750 |
| `Region/∼rr/m_name` | 31 | 0 |
| `Municipal/∼municipals/r_name` | 31 | 0 |
| `News/municipal/ммunicipals/r_name` | 5485 | 0 |
| `News/∼news/rr/r_name` | 5297 | 0 |
| `&region/municipals/news/ region/r_name` | 359 | 312 |
| `&municipal22/news/∼news/m_name` | 0 | 0 |
| `&municipal11/∼municipal/title` | 234 | 16 |
| `News[municipal/m_name == "Город Иркутск"]/text` | 5063 | 47 |

Нами проводились эксперименты по сравнению эффективности и выразительности обоих подходов. Реализация Libretto API — более универсальный способ для выполнения Libretto на базах данных, реализующий полный функционал Libretto. Второй метод, основанный на подмене транслятора, теоретически менее выразителен, хотя с практической точки зрения является достаточно полным – в процессе проведенных экспериментов все необходимые запросы оказались выразимыми на языке $\mathcal{L}_{\mathrm{sql}}$. Таблица 1, в которой представлены измерения для метода подмены хранилища (API) и метода подмены транслятора (TRN), показывает, что второй метод трансляции принципиально эффективнее первого. При трансляции в Libretto API происходит мно-

жество вызовов простых SQL-запросов, а во втором случае выполняется один сложный SQL-запрос. Кроме того, второй метод намного более восприимчив к возможности применения разнообразных оптимизаций.

## Заключение

Результаты представленных в работе исследований являются основой для построения компоненты Libretto-системы, ответственной за вовлечение информации из баз данных в логическое пространство языка. Кроме того, описанная в данной работе технология является одной из основ для построения на базе Libretto-системы среды уровня PaaS (Platform as a Service) для разработки больших распределенных систем в рамках технологий облачных вычислений.

Дальнейшая работа в данном направлении будет заключаться в последовательном расширении языка $\mathcal{L}_{\text{sql}}$, в частности, средствами модификации баз данных, а также разработкой методов оптимизации генерируемого кода.

## Список литературы

1. Казаков И. А. Базы данных как онтологии / И. А. Казаков, А. В. Манцивода // Изв. Иркут. гос. ун-та. Сер. Математика. – 2011. – Т. 4, № 1. – С. 20–30.
2. Казаков И. А. Алгебры Кодда и дескриптивные логики / И. А. Казаков // Изв. Иркут. гос. ун-та. Сер. Математика. – 2011. – Т. 4, № 3. – С. 68–73.
3. Малых А. А. Объектно-ориентированная дескриптивная логика / А. А. Малых, А. В. Манцивода // Изв. Иркут. гос. ун-та. – Сер. Математика. – 2011. – Т. 4, № 1. – С. 57–72.
4. Libretto: объектно-итерационный язык и система управления объектными базами знаний [Электронный ресурс]. – URL: http://ontobox.org.
5. Malykh A. A Query Language for Logic Architectures / A. Malykh, A. Mantsivoda // Proceedings of 7th International Conference "Perspectives of System Informatics". – Springer-Verlag Berlin Heidelberg, Lecture Notes in Computer Science 5947. – 2010. – P. 294-–305.
6. Abiteboul S. Foundations of Databases / S. Abiteboul, R. B. Hull, V. Vianu. – Addison-Wesley, 1995. – 685 p.
7. Rosati R. On Combining Description Logic Ontologies and Nonrecursive Datalog Rules // Lecture Notes in Computer Science. – 2008. – Vol. 341. – P. 13–27. – (Web Reasoning and Rule Systems).

**Казаков Илья Анатольевич**, кандидат физико-математических наук, ведущий инженер, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1, тел.: (3952)242210
(e-mail: `kazakov@baikal.ru`)

**Малых Антон Александрович**, кандидат физико-математических наук, старший научный сотрудник, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1, тел.: (3952)242210 (e-mail: malykh@gmail.com)

**Манцивода Андрей Валерьевич**, доктор физико-математических наук, профессор, Институт математики экономики и информатики, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1, тел.: (3952)242210 (e-mail: andrei@baikal.ru)

---

**I. A. Kazakov, A. A. Malykh, A. V. Mantsivoda**

## Embedding Relational Databases in Object Ontologies: Implementation Issues

**Abstract**. In this paper the problem of database embedding in logical knowledge-management environment is considered. The approaches to the implementation of database embedding in ontologies are established through their modeling based on object theories. This method focuses on the unified database management within big distributed information systems.

In introduction we give the relevance of the investigated problem. At the first part, we point attention on objective theory of databases. According to it, we determine the basic mechanism and the rules of objective model building on a random relational database system. Then we define modeling of external keys of a database through objective theory features.

Further we consider the implementation of a query language interpreter «Libretto» on relational databases. Then the general architecture of the «Libretto» system is given. In addition to this, we mention on two ways of implementation mechanisms of the databases which are based on «Libretto». The first one is about implementation Libretto API in the context of DBMS; the latest could be described as a substitution of the translator «Libretto».

Later the databases sublanguage «Libretto» is determined on order for implementation method of substitution of the translator «Libretto». This databases sublanguage replies for request generating. Then we analyze translation of elementary requests, transmission ways, transmission of inverse features, transmission of inserted requests, predicates transmission.

Finally, we give the comparison of the efficiency of 2 above-mentioned ways, summarize undertaken work and see the further development of these ways.

**Keywords:** ontology, database, object theory, description logic, Libretto, big distributed systems.

## References

1. Kazakov I.A., Mantsivoda A.V. Databases as Ontologies. *The Bulletin of Irkutsk State University. Mathematics*, 2011, vol. 4, no.1, pp. 20–30.
2. Kazakov I.A. Kodds Algebra and Description Logics. *The Bulletin of Irkutsk State University. Mathematics*, 2011, vol. 4, no.3, pp. 68–73.
3. Malykh A.A., Mantsivoda A.V. Object-oriented description logics. *The Bulletin of Irkutsk State University. Mathematics*, 2011, vol. 4, no. 1, pp. 57–72.

4. Malykh A.A., Mantsivoda A.V. Query Language for Logic Architectures. *Proceedings of 7th International Conference "Perspectives of System Informatics". Springer-Verlag Berlin Heidelberg, Lecture Notes in Computer Science 5947*, 2010, pp. 294-–305.
5. Libretto: object–iterative programming language and object databases manage system. URL: http://ontobox.org.
6. Rosati R. On Combining Description Logic Ontologies and Nonrecursive Datalog Rules. *Lecture Notes in Computer Science. Web Reasoning and Rule Systems*, 2008, vol. 341, pp. 13–27.
7. Abiteboul S., Hull R., Vianu V. Foundations of Databases. *Addison-Wesley*, 1995. 685 p.

**Kazakov Ilya Anatol'evich**, Candidate of Sciences (Physics and Mathematics), Irkutsk State University, 1, K. Marx st., Irkutsk, 664003, tel.: (3952)242210 (e-mail: kazakov@baikal.ru)

**Malykh Anton Aleksandrovich** , Candidate of Sciences (Physics and Mathematics), Senior Research Scientist, Irkutsk State University, 1, K. Marx st., Irkutsk, 664003, tel.: (3952)242210 (e-mail: malykh@gmail.com)

**Mantsivoda Andrei Valer'evich**, Doctor of Sciences (Physics and Mathematics), Professor, Irkutsk State University, 1, K. Marx st., Irkutsk, 664003, tel.: (3952)242210 (e-mail: andrei@baikal.ru)