УНИВЕРСАЛЬНАЯ МОДЕЛЬ ДАННЫХ ДЛЯ РЕШЕНИЯ ИССЛЕДОВАТЕЛЬСКИХ ЗАДАЧ

В.А. Салеев, Д.Е. Яблоков

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

В статье рассматривается методология построения универсального хранилища данных, предназначенного для приложений ориентированных на специалистов занимающихся научными исследованиями. Предполагается, что структура хранилища не будет жёстко привязана к какой-либо научной предметной области, а будет востребована во многих проектах связанных с накоплением, анализом и обработкой данных. Предлагаемый подход к организации системы хранения опирается на базовые принципы, концепции и технологии, используемые в процессе проектирования программного обеспечения, основанном на анализе предметной области и обнаружении критериев общности и изменчивости, являющихся гарантией хорошей техники абстрагирования одного из основных инструментов для работы с данными.

Ключевые слова: универсальная модель данных, структура хранения, уровень абстракции, типизация данных, внутренняя классификация, категорирование данных, внешняя классификация.

Создание компьютерных систем для научных исследований – дело далеко не простое. По мере того как возрастает их сложность, процессы конструирования соответствующих программных продуктов становятся всё более трудоёмкими, причём затраты на разработку растут экспоненциально. Это относится и к процессу проектирования и реализации подсистем хранения, необходимых для функционирования приложений, использующихся в научной работе, а также для обеспечения специалистов-исследователей разнородной, но в тоже время достоверной и полной информацией. Многие научные лаборатории или исследовательские центры могли бы работать с приложениями на основе предлагаемой модели данных из-за её универсальности и высокого уровня абстракции при описании свойств объектов и проведении их классификации. Далее приводятся фрагменты ER-модели и примеры кода из DDL-скрипта для создания базы данных использующей универсальную модель хранения информации.

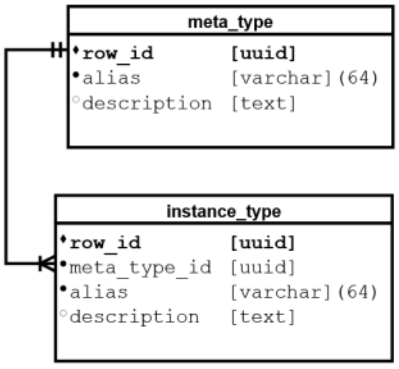

Рис. 1. Типизация данных

Пример кода создания таблицы мета-типов, т.е. набора элементарных примитивов, относящихся к некоторым внутренним понятиям предметной области, перечень которых априори установлен:

```
CREATE TABLE IF NOT EXISTS meta_type
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	alias varchar(128) NOT NULL,
	description text,
	CONSTRAINT meta_type_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT meta_type_uindex
		UNIQUE(alias)
);
```

Пример кода создания таблицы типов экземпляров, т.е. перечня концепций определяющих некоторую совокупность свойств, отличающих их владельца от других сущностей, не содержащих идентичный набор свойств:

```
CREATE TABLE IF NOT EXISTS instance_type
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	meta_type_id uuid NOT NULL,
	alias varchar(128) NOT NULL,
	description text,
	CONSTRAINT instance_type_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT instance_type_fkey
		FOREIGN KEY(meta_type_id)
		REFERENCES meta_type(row_id),
	CONSTRAINT instance_type_uindex
		UNIQUE(alias)
);
```

Хранящуюся в данном отношении информацию можно использовать как инструмент внутренней классификации. Организация такой классификации возможна по каким-либо существенным признакам, характеризующим общность предметов, понятий или явлений. Такой способ классификации очень важен при проведении научных исследований, т.к. является средством для закрепления результатов изучения закономерностей классифицируемых данных.

Как известно, классификацией является любое системное распределение данных по каким-либо существенным признакам, выбранным для удобства их представления и обработки. Кроме того, инструменты классификации разделяются по способу их воздействия на классифицируемые элементы информации.

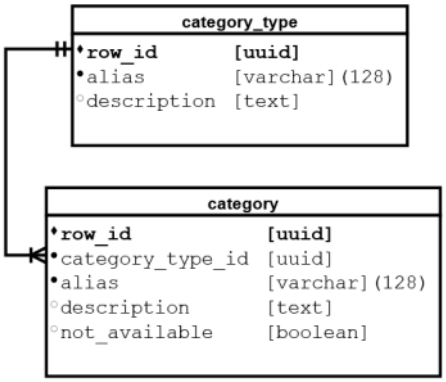

Рис. 2. Категорирование данных

Пример кода создания таблицы типов категорий, т.е. набора элементарных примитивов, относящихся к некоторым внешним понятиям предметной области, перечень которых априори установлен:

```
CREATE TABLE IF NOT EXISTS category_type
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	alias varchar(128) NOT NULL,
	description text,
	CONSTRAINT category_type_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT category_type_uindex
		UNIQUE(alias)
);
```

Пример кода создания таблицы категорий, т.е. инструмента обеспечивающего необходимую гибкость через дополнительный уровень косвенности при определении критериев общности для сущностей вне зависимости от их типизации:

```
CREATE TABLE IF NOT EXISTS category
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	category_type_id uuid NOT NULL,
	alias varchar(128) NOT NULL,
	description text,
	not_available boolean,
	CONSTRAINT category_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT category_fkey
		FOREIGN KEY(category_type_id)
		REFERENCES category_type(row_id),
	CONSTRAINT category_uindex
		UNIQUE(alias)
);
```

Сущность может быть связана с одной или несколькими категориями. Это делает возможным применение дополнительного инструмента классификации при анализе или декомпозиции конкретной сущности или семейства, к которому она принадлежит. Иными словами, категорирование позволяет получить представление о сущности в терминах множества связанных с ней категорий. В частности, механизм категорирования можно использовать для указания соответствия конкретного элемента данных определённому семейству или классу, который по замыслу исследователя, должен однозначно идентифицировать какое-либо измерение, свойство или качество информации.

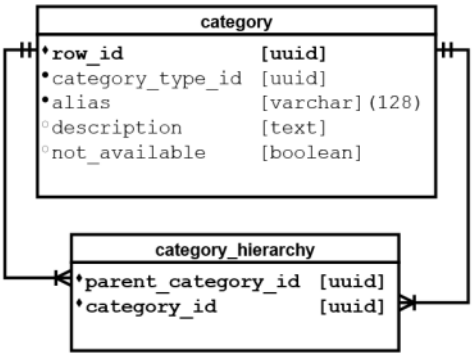

Рис. 3. Взаимосвязь категорий по принципу «подобное поведение»

Пример кода создания таблицы иерархии категорий, определяющей внешние критерии общности по принципу подобного поведения:

```
CREATE TABLE IF NOT EXISTS category_hierarchy
(
	parent_category_id uuid NOT NULL,
	category_id uuid NOT NULL,
	CONSTRAINT category_hierarchy_pkey
		PRIMARY KEY(category_id,
			parent_category_id),
	CONSTRAINT category_hierarchy_fkey_1
		FOREIGN KEY(parent_category_id)
		REFERENCES category(row_id),
	CONSTRAINT category_hierarchy_fkey_2
		FOREIGN KEY(category_id)
		REFERENCES category(row_id)
);
```

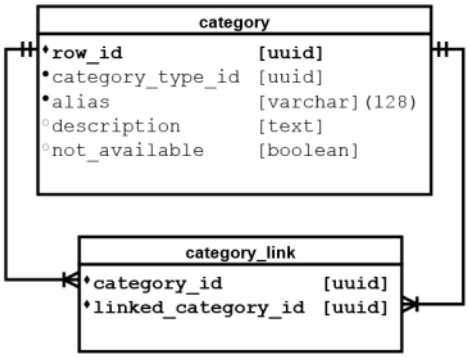

Рис. 4. Взаимосвязь категорий по принципу «является частью»

Пример кода создания таблицы присоединённых категорий, содержащей данные о внешних критериях общности по принципу отношений часть-целое:

```
CREATE TABLE IF NOT EXISTS category_link
(
	category_id uuid NOT NULL,
	linked_category_id uuid NOT NULL,
	CONSTRAINT category_link_pkey
		PRIMARY KEY(category_id,
			linked_category_id),
	CONSTRAINT category_link_fkey_1
		FOREIGN KEY(category_id)
		REFERENCES category(row_id),
	CONSTRAINT category_link_fkey_2
		FOREIGN KEY(linked_category_id)
		REFERENCES category(row_id)
);
```

Пример кода создания таблицы свойств, определяющей дескрипторы глобальной коллекции свойств:

```
CREATE TABLE IF NOT EXISTS property
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	structural_code smallint NOT NULL,
	CONSTRAINT property_pkey
		PRIMARY KEY(row_id)
);
```

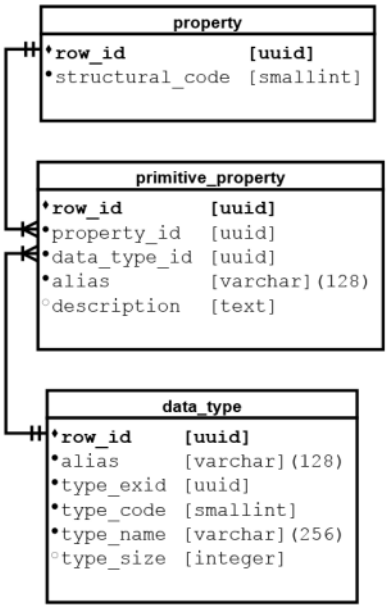

Рис. 5. Примитивные свойства

Пример кода создания таблицы типов данных:

```
CREATE TABLE IF NOT EXISTS data_type
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	alias varchar(128) NOT NULL,
	type_exid uuid NOT NULL,
	type_code smallint NOT NULL,
	type_name varchar(256),
	type_size integer,
	CONSTRAINT data_type_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT data_type_uindex1
		UNIQUE(alias),
	CONSTRAINT data_type_uindex2
		UNIQUE(type_exid),
	CONSTRAINT data_type_uindex3
		UNIQUE(type_code)
);
```

Пример кода создания таблицы примитивных свойств:

```
CREATE TABLE IF NOT EXISTS primitive_property
(
	row_id uuid NOT NULL
		DEFAULT uuid_generate_v4(),
	property_id uuid NOT NULL,
	data_type_id uuid NOT NULL,
	alias varchar(128) NOT NULL,
	description text,
	CONSTRAINT primitive_property_pkey
		PRIMARY KEY(row_id),
	CONSTRAINT primitive_property_fkey_1
		FOREIGN KEY(property_id)
		REFERENCES property(row_id),
	CONSTRAINT primitive_property_fkey_2
		FOREIGN KEY(data_type_id)
```

```
            REFERENCES data_type(row_id),
      CONSTRAINT primitive_property_uindex
            UNIQUE(alias)
);
```

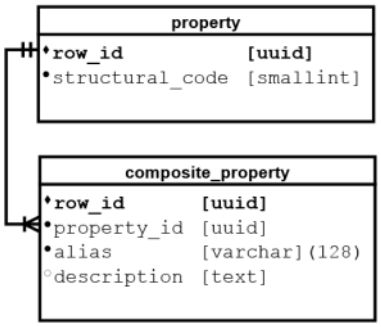

Рис. 6. Составные свойства

Пример кода создания таблицы составных свойств:

```
CREATE TABLE IF NOT EXISTS composite_property
(
      row_id uuid NOT NULL
            DEFAULT uuid_generate_v4(),
      property_id uuid NOT NULL,
      alias varchar(128) NOT NULL,
      description text,
      CONSTRAINT composite_property_pkey
            PRIMARY KEY(row_id),
      CONSTRAINT composite_property_fkey
            FOREIGN KEY(property_id)
            REFERENCES property(row_id),
      CONSTRAINT composite_property_uindex
            UNIQUE(alias)
);
```

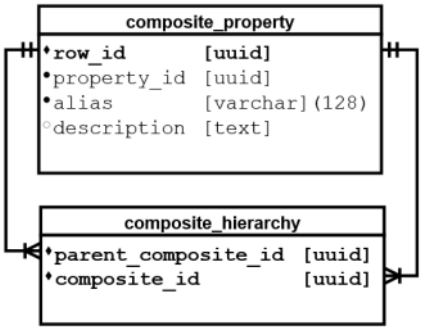

Рис. 7. Взаимосвязь составных свойств по принципу «является частью»

Пример кода создания таблицы иерархии составных свойств:

```
CREATE TABLE IF NOT EXISTS composite_hierarchy
(
      parent_composite_id uuid NOT NULL,
      composite_id uuid NOT NULL,
      CONSTRAINT composite_hierarchy_pkey
            PRIMARY KEY(parent_composite_id,
                  composite_id),
      CONSTRAINT composite_hierarchy_fkey_1
            FOREIGN KEY(parent_composite_id)
            REFERENCES
```

```
                composite_property(row_id),
        CONSTRAINT composite_hierarchy_fkey_2
                FOREIGN KEY(composite_id)
                REFERENCES
                        composite_property(row_id)
);
```

Основными преимуществами предлагаемого подхода можно считать возможность применения рассматриваемой универсальной модели к любому виду информации, а также возможность определения системы понятий дающих основу для создания предметно-ориентированного языка. Наиболее приближенный к контексту конкретной области научных знаний, такой язык может детально представлять взаимосвязь между структурой предметной области и тем, как в нём выражены её общность и вариация.

Литература

1. Фаулер, М. Архитектура корпоративных программных приложений / Мартин Фаулер – М.: Вильямс, 2007. – 544 с.
2. Эмблер, С.В. Рефакторинг баз данных: Эволюционное проектирование / С.В. Эмблер, П.Дж. Садаладж – М.: Вильямс, 2007. – 368 с.
3. Фаулер, М. Предметно-ориентированные языки программирования / Мартин Фаулер – М.: Вильямс, 2011. – 577 с.
4. Яблоков, Д.Е. Парадигмы программирования /Д.Е. Яблоков // XIV МНПК Научное обозрение физико-технических наук в XXI веке. – 2015. – №2(14). – С. 94-98.