ISSN 2413–9599 (Print)
ISSN 2541–870X (Online)

www.radioprom.org

DOI: 10.21778/2413-9599-2021-31-1-65-73
УДК 004.652.5

Построение базы данных для систем мониторинга сложных промышленных объектов

М. З. Бененсон[1], Е. А. Алексеева[1, 2]

[1] ФГАОУ ВО «МИРЭА — Российский технологический университет», Москва, Россия
[2] АО «Научно-исследовательский институт вычислительных комплексов им. М. А. Карцева», Москва, Россия

Постановка проблемы. При создании систем мониторинга промышленных объектов широкого назначения возникает необходимость решения задачи обработки и хранения объектов со сложной структурой данных. Пользователю должны быть предоставлены средства обработки и хранения определенных им типов данных и объектов.

Цель — разработка программной реализации интерфейса взаимодействия с базой данных, встроенной в систему мониторинга промышленных объектов.

Результаты. Разработан программный интерфейс взаимодействия с объектно-ориентированной базой данных. В соответствии с предполагаемой структурой промышленных объектов используются три программных класса для описания объектов различных типов. Разработаны методы классов, позволяющие задавать для различных типов объектов переменное число атрибутов. Предложен метод извлечения объекта с заданными значениями атрибутов, аналогичный запросу по образцу, и метод сложных (естественных) запросов, формируемых на языке разработки приложения.

Практическая значимость. Предлагаемая программная реализация интерфейса взаимодействия с встроенной базой данных может быть использована при создании широкого спектра систем индустриального мониторинга. Данный подход позволяет значительно сократить вычислительные ресурсы, необходимые для реализации подобных систем, снижает время и стоимость их разработки.

Ключевые слова: индустриальный мониторинг, объектно-ориентированная база данных, прикладной программный интерфейс, уникальный идентификатор объекта, запрос по образцу, естественный запрос

Для цитирования:
Бененсон М. З., Алексеева Е. А. Построение базы данных для систем мониторинга сложных промышленных объектов. 2021. Т. 31. № 1. С. 65–73. DOI: 10.21778/2413-9599-2021-31-1-65-73

© Бененсон М. З., Алексеева Е. А., 2021

Building a database for complex industrial monitoring systems

M. Z. Benenson[1], E. A. Alekseeva[2]

[1] MIREA — Russian Technological University, Moscow, Russia
[2] M. A. Kartsev Scientific and Research Institute of Computing Systems JSC, Moscow, Russia

Problem statement. When creating monitoring systems for industrial facilities for a range of purposes, it becomes necessary to solve processing and storing objects with a complex data structure. The user must be provided with tools for processing and storing the defined data and object types that they have defined.

Objective. Development of a software implementation of the interface for interaction with the database built into industrial facilities' monitoring system.

Results. A software interface for interacting with an object-oriented database has been developed. Three programming classes are used to describe various types of industrial system objects. Class methods have been developed that allow setting a variable number of attributes for different object types. The authors propose a method for extracting an object with specified attribute values, similar to the QBE method, and a method for complex (natural) queries written in the application development language.

Practical implications. The proposed software implementation of the interface for interaction with the built-in database can be used to create a wide range of industrial monitoring systems. This approach allows to significantly reduce the computing resources required for the implementation of such systems, reduces the time and cost of their development.

Keywords: industrial monitoring, object-oriented database, application programming interface, unique identifier for the object, query-by-example, native query

For citation:
Benenson M. Z., Alekseeva E. A. Building a database for complex industrial monitoring systems. Radio industry (Russia), 2021:31(1); pp. 65–73. (In Russian). DOI: 10.21778/2413-9599-2021-31-1-65-73

Введение

Целью предлагаемой работы является создание универсального программного обеспечения для систем мониторинга промышленных объектов различного назначения, которые относятся к классу систем диспетчерского управления и сбора данных (*SCADA*-системы).

SCADA-системы используются во всех отраслях хозяйства, где требуется обеспечивать операторский контроль за технологическими процессами в реальном времени. Одной из наиболее популярных *SCADA*-систем является *GENESIS32* [1], разработанная отечественной компанией *Iconics*.

В *SCADA*-системе *GENESIS32* для хранения и обработки данных используется синтаксис языка формирования запросов к реляционным СУБД — *ANSI SQL*, который позволяет реализовать поддержку таких СУБД, как *MySQL, Oracle* и ряда других.

Использование больших коммерческих СУБД при построении системы промышленного мониторинга значительно усложняет программное обеспечение в целом, затрудняет его переносимость и повышает стоимость программного продукта.

Соответственно, естественным решением поставленной задачи является использование встроенной СУБД.

Наиболее известной встроенной СУБД является *Sqlite*, которая используется при создании различных приложений. В отличие от большинства других баз данных, *Sqlite* представляет из себя как бы библиотеку и работает в том же процессе, что и само приложение. Однако, большим недостатком реляционных БД, к которым относится и *Sqlite*, является то, что они могут работать с весьма ограниченными по семантике наборами данных.

В предложенной Э. Кодом реляционной модели информация представляется в виде отношений (двумерных таблиц). При работе с таблицами не могут быть использованы такие возможности, предоставляемые объектно-ориентированными

языками программирования, как наследование классов и полиморфизм.

Соответственно, для такого сложного приложения, как система мониторинга промышленных объектов, более эффективным представляется применение объектно-ориентированных баз данных (ООБД).

В работе M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik [2] приведено определение объектной модели, в соответствии с которым были определены обязательные средства поддержки объектно-ориентированных баз данных (ООБД), в том числе: идентификаторы объектов; коллекции; типы, определяемые пользователем; классы; инкапсуляция; наследование, поддержка произвольных запросов.

В 1991 году был образован консорциум *ODMG*, основной целью которого было создание стандартов (общей модели) для ООБД, причем эти стандарты не являются официальными.

В последней базовой версии стандарта ООБД — *ODMG* 3.0 [3] предлагается использовать следующие основные компоненты:

- объектную модель *(Object Model — OM)*, которая определяет семантику объектов базы данных;
- язык определения объектов *(Object Definition Language — ODL),* предназначенный для определения спецификаций объектных типов;
- язык объектных запросов *(Object Query Language — OQL)*, предоставляющий средства доступа к объектной базе данных;
- язык обмена данными между объектами *(Object Interchange Format — OIF)*, применяемый для выгрузки и загрузки информации о текущем состоянии системы управления объектными данными в один или несколько файлов;
- средства связывания объектной модели с объектами языков программирования, устанавливающие соответствие между конструкциями языка *ODL* и конструкциями универсального языка программирования.

Основными элементами модели *ODMG* являются изменяемые объекты и литералы, представляющие собой значение или набор значений, которые не могут быть изменены. Каждый объект обладает уникальным идентификатором *(OID)*, литералы не имеют *OID*.

Все объекты заданного типа демонстрируют общее правило поведения и состояние. Объект иногда называют экземпляром его типа, правилом поведения называется набор операций, которые могут быть выполнены объектом или над ним, а состояние определяется значениями свойств, которыми обладает объект. Свойство может быть либо атрибутом объекта, либо связью с другими объектами.

Для создания нового экземпляра объекта используется метод *new()* соответствующего типа.

Язык определения объектов *(ODL)* предназначен для определения атрибутов и связей между типами. Он эквивалентен языку определения данных *DDL* (подмножество языка *SQL*) реляционных СУБД.

Язык объектных запросов *OQL* может применяться для построения двух типов запросов:

- ассоциативный запрос возвращает коллекцию объектов, причем решение задачи выбора способа поиска в базе данных объектов, соответствующих запросу, возлагается на системе управления объектными данными *(Object Data Management System — ODMS)*, а не на прикладную программу;
- навигационный запрос обеспечивает доступ к отдельным объектам, для перемещения от одного объекта к другому применяются связи между объектами.

Определение процедуры доступа к требуемым объектам выполняется в прикладной программе. Запрос на языке *OQL* формируется с помощью логических выражений, включающих выполняемые операторы запроса.

Языковые средства связывания с данными определяют способы установления соответствия между конструкциями языка определения объектов *(ODL)* и языка манипулирования объектами *(OML)* с конструкциями языка программирования, например *C++*.

Для связывания с данными предназначена библиотека соответствующего языка программирования, содержащая классы и функции, которые реализуют конструкции языков *ODL* и *OML*.

Приведенный в работе С. Д. Кузнецова [4] краткий обзор основных особенностей наиболее популярных коммерческих ООСУБД *(“GemStone”, “ITASCA”, “ObjectStore”, “Objectivity/DB”, “Versant”)* показывает очень большую техническую разнородность этих систем. В общем смысле все такие системы соответствуют базовой модели *ODMG*, но крайне редко в качестве языка запросов поддерживается *OQL* и ни в одной из этих систем не поддерживается язык определения объектов *ODL*.

В 1990 году М. Стоунбрейкером и другими авторами был опубликован «Манифест СУБД третьего поколения» [5], в котором предложена объектно-реляционная модель данных, основанная на сопоставлении объектов с таблицами базы данных. Эта модель существенно отличается от

реляционной модели в том виде, в котором ее предложил Э. Кодд [6].

Теоретическое обоснование идей построения объектно-реляционной модели было предложено в классической работе К. Дейта и Х. Дарвена [7].

Для объектно-реляционных баз данных (ОРБД) было предложено использовать уникальные идентификаторы объектов, поддержку механизма наследования, методы базы данных и ряд других свойств, характерных для ООБД. Например, в объектно-реляционной базе данных (ОРБД) *Postgresql* предусмотрено использование хранимых процедур, написанных на языке *PL/PgSQL*, и табличное наследование, что позволяет поддерживать полиморфизм при выполнении запросов к базе данных [8].

Однако объектно-реляционный подход не позволяет пользователю сохранять объекты приложения непосредственно в базе данных. Кроме того, разработанное для таких баз данных приложение должно взаимодействовать с полномасштабной СУБД, что ограничивает его переносимость.

В стандарт *SQL:1999* [9] для реляционной модели данных были введены определяемые пользователем типы данных *(UDT)* и типизированные таблицы.

Аналогом объекта в объектно-реляционной модели выступает строка типизированной таблицы, ассоциированной с *OID*.

В соответствии с моделью *ODMG* для работы с ООБД требуется создать схему базы данных привязку объектной модели к языку программирования, что необходимо для преобразования объектов приложения в объекты базы данных.

В данной работе для решения задачи мониторинга сложных промышленных объектов, имеющих конкретную область применения, предлагается использовать собственную объектно-ориентированную базу данных, непосредственно встроенную в систему управления.

Пользователю предоставляется возможность разрабатывать собственные типы объектов и собственные типы данных, что позволяет решить актуальную задачу создания универсального программного обеспечения для широкого спектра задач мониторинга промышленных объектов.

Программный интерфейс взаимодействия с базой данных системы индустриального мониторинга

В работе [10] предлагается иерархическая модель мониторинга станций московского метро, которая включает в себя четыре уровня детализации: карту моделей, модель станции, объекты модели и атрибуты модели.

Программная часть предлагаемого решения реализована на программной платформе *NET* компании *Microsoft*.

В качестве интерфейса взаимодействия с объектно-ориентированной базой данных предлагается использовать библиотеку *Db4objects.Db4o.dll*, предоставляемую СУБД *db4o* [11]. При этом не требуется выполнять привязку объектной модели к языку программирования, так как программное приложение *API db4o* непосредственно содержит все классы, необходимые для хранения и извлечения объектов.

Для доступа к данным требуется использовать меньше строк программного кода, причем схема базы данных значительно упрощается и существенно повышается производительность выполнения запросов.

Структура данных для системы мониторинга в реляционной модели может быть представлена в виде трех таблиц: таблицы типов *(табл. 1)* соответствует реализации классов в объектно-ориентированном программировании — ООП), таблицы устройств *(табл. 2)* и таблицы объектов *(табл. 3)* — в терминах ООП отвечает за создание экземпляров реализованных классов и заполнение их информационных полей.

Таблица 1. Пользовательские типы для описания 3D модели
Table 1. Custom types for describing a 3D model

Уникальное имя типа / Unique name of the type	Число атрибутов / Number of attributes	Имя атрибута / Attribute name	Тип атрибута / Attribute type	Имя атрибута / Attribute name	Тип атрибута / Attribute type
Базовый / Basic	0	–	–	–	–
Основной / Main	1	Описание / Description	Строка / String	–	–
…	…	…	…	…	…
Комплексный / Comprehensive	10	Описание / Description	Строка / String	Количество / Qty	Целый / Complete

Таблица 2. Устройства, привязанные к атрибутам объектов 3D модели
Table 2. Devices linked to attributes of 3D model objects

Уникальное имя объекта / Unique object name	Драйвер внешнего устройства / External device driver	Число связанных атрибутов / Number of associated attributes	Атрибут 1 / Attribute 1	Атрибут 2 / Attribute 2	Атрибут N / Attribute N
Часы № 1 / Clock 1	clock_driver_1	1	Время / Time	–	–
Часы № 2 / Clock 2	clock_driver_2	1	Время / Time	–	–
Рамка детектора № 1 / Detector frame 1	detector_drv_1	2	Статус / Status	Снимок / Snapshot	–

Таблица 3. Объекты 3D модели
Table 3. 3D model objects

Уникальное имя объекта / Unique object name	Тип объекта / Object type	Имя атрибута / Attribute name	Значение Атрибута / Attribute value	Имя атрибута / Attribute name	Значение Атрибута / Attribute value
Рамка детектора № 1 / Detector frame 1	Комплексный / Comprehensive	Описание Description	Металло-детектор / Metal detector	Количество / Qty	1
…	…	…	…	…	…
Часы № 1 / Clock 1	Часы / Clock	Время / Time	12:34:56	–	–
Часы № 2 / Clock 2	Часы / Clock	Время / Time	12:34:56	–	–

Представление иерархических данных в реляционном виде требует большого количества таблиц, которое соответствует разнообразию атрибутов описываемых объектов.

Для ведения базы данных (БД) реального времени с технологической информацией, описываемой в виде предлагаемой структуры, на языке *C#* было разработано программное обеспечение, использующее прикладной интерфейс взаимодействия с объектно-ориентированной БД *db4o*

Объектно-ориентированные базы данных — это базы данных, в которых информация, как и в объектно-ориентированных языках программирования, представлена в виде объектов.

Db4o является ООБД, которая предназначена для использования с прикладными программами, написанными на языках *Java* и *C#*.

При создании программного проекта приложения на языке *C#* в него необходимо добавить ссылку на библиотеку *Db4objects.Db4o.dll*, которая содержит все классы, необходимые для хранения и извлечения объектов базы данных.

База данных в *db4o* размещается в отдельном локальном файле, для доступа к которому служит метод *Db4o.OpenFile*, позволяющий открыть экземпляр объекта *IObjectContainer*. Например:

```
db = Db4oFactory.OpenFile("devices.data").
```

При открытии *IObjectContainer* неявным образом создается контекст транзакции, которая заканчивается при закрытии *IObjectContainer*.

Для завершения транзакции требуется закрыть контейнер объектов в соответствующем блоке *finally*:

```
finally{
  if (db != null)
    db.Close();
}
```

В базе данных *db4o* хранятся непосредственно сами объекты, причем за идентификацию объектов отвечает неявный или скрытый указатель *this*, который в терминологии ООБД называется *OID* (идентификатор объекта) и представляет собой указатель на динамическую переменную в памяти.

В качестве аналога объекта в реляционной модели выступает строка реляционной таблицы.

В соответствии с предлагаемой структурой данных было создано три класса:

- *Types3D* — типы пользователей;
- *ObjectsDevice* — устройства *3D*-модели;
- *Objects3D* — объекты *3D*.

Класс *ObjectsDevice* имеет четыре элемента данных: *id* — идентификационный номер устройства,

quant — количество атрибутов, *namedevice* — название устройства, *nametype* — пользовательский тип этого устройства.

В *dbo4* объект создается конструктором некоторого объектного типа. Объект обладает свойством индивидуальности, которое выражается в том, что при создании объекта ему присваивается уникальный идентификатор *(OID)*, отличающий его от любого другого объекта.

Конструктор и функции: *set()* и *get()* для установки и получения параметров объекта являются методами класса *ObjectsDevice*.

Например, для создания объекта *ob* класса *Objects3D* вызывается конструктор этого класса:

```
Objects3D ob = new Objects3D(id, lquant, namedevice, nametype);
```

В конструкторе элементам класса присваивается значение соответствующих параметров:

```
public Objects3D(int id, int quant, String name, String nametype){
        this.id = id;
        this.unicalname = name;
        this.typesname = nametype;
        this.quant = lquant;
 }
```

Поскольку различным объектам класса *Objects3D* может соответствовать различное число атрибутов, то и число аргументов, которые передаются методам данного класса заранее неизвестно. Однако, язык *C#* позволяет задавать переменное число параметров метода с помощью указания в списке его аргументов ключевого слова *"params"* и указателя массива [12].

С этой целью в конструкторе класса используется метод установки значений атрибутов:

```
class Objects3D{
  //название объекта
   public String name;
  //тип объекта
  public String typesname;
  //количество атрибутов
   public int quant;
   // определение массива названий атрибутов
   private String[] nameatr = new string[4];
   //определение массива значений атрибутов
   private String[] valatr = new string[4];
   //метод установки значений атрибутов
   public void setvalAtrib (params Atr[] p {
   for (int i = 0; i < quant; i++)
      valatr[i] = p[i].y;
   }
}
```

В методе *setvalAtrib() p* определяет массив с переменным числом параметров, а тип массива *Atr* (Атрибуты) определяется в виде класса со своим конструктором:

```
class Atr{
  public String y;
//Конструктор класса
  public Atr(String x) {
    this.y= x;
   }
}
```

В приведенном коде элемент у класса *Atr* определен как строка произвольной длинны, что позволяет задавать в качестве параметра любые символьные строки.

Приведем в качестве примера код передачи значений атрибутов объекта «Рамка детектора № 1»:

```
// задание значений атрибутов
Atr val1 = new Atr("Металлодетектор");
Atr val2 =new Atr("1");
//вызов метода установки значений атрибутов
ob.setvalAtrib(val1, val2);
```

Для хранения экземпляра объекта в баз данных вызывается метод Store контейнера объектов, например:

```
db.Store(ob);
```

Для получения значений атрибутов используется метод get:

```
public String getAttribute(int j) {
return valatr[j];
}
```

Для удаления объект необходимо вызвать метод *Delete* контейнера, например:

```
db.Delete(ob);
```

Механизм формирования запросов к объектно-ориентированной базе данных

Ключевые возможности любой СУБД определяются механизмом создания запросов.

Для создания запросов к реляционным базам данных служат язык структурированных запросов *SQL* и язык запросов по образцу *QBE*.

Предложенный группой *ODMG* язык объектных запросов *OQL* очень близок к *SQL/92*, его расширения относятся к таким объектно-ориентированным понятиям, как «сложные объекты», «объектные идентификаторы», «путевые выражения», «полиморфизм», «вызов операций» и «отложенное связывание».

Хотя язык объектных запросов *OQL* и является простым с концептуальной точки зрения, но он оказался очень сложным для использования непрограммистами и по этой причине не получил широкого распространения.

В ООСУБД *db4o* для создания запросов предлагается использовать программные методы,

написанные непосредственно на языке разработки приложения.

В представленной работе для создания запросов используются два программных метода: метод извлечения объекта с заданными значениями атрибутов и метод построения сложных запросов на основе логических выражений.

Метод извлечения объекта с заданными значениями конкретных атрибутов по аналогии с реляционными базами данных называется запросом по образцу — *QBE*.

При создании запроса по образцу в качестве параметра запроса указывается шаблон поиска. Поля, указанные в шаблоне поиска в виде *null, 0* или *"" (пустая строка)*, рассматриваются как не имеющие ограничений.

Например, сформируем запрос по образцу для поиска объектов с типом «Часы»:

```
Objects3D qbe = new Objects3D(0, 0, null, "Часы");
```

Запрос возвращает коллекцию типа *IObjectSet*, который является классом *db4o*. Набор *qbe* содержит все объекты, соответствующие параметрам шаблона, имеющим не нулевое значение:

```
IObjectSet result = db.QueryByExample(qbe);
```

Здесь db.*QueryByExample* — ссылка на контейнер объектов базы данных.

Распечатка результата выполнения запроса *qbe* приведен на *рис. 1*.

```
Результат QBE запроса :
_____________________________
| Ид.№ объекта | Имя объекта |
_____________________________
|      2       | Часы №1     |
_____________________________
|      3       | Часы №2     |
_____________________________
```

Рисунок 1. Результат выполнения QBE запроса
Figure 1. Result of QBE query execution

В реляционных базах данных выборка записей из одной или нескольких таблиц выполняется с помощью оператора *SELECT* в соответствии с заданным логическим выражением (предикатом).

В *db4o* не применяются специальные языки запросов, подобные *SQL* или *OQL*, вместо конструкций этих языков при создании сложных запросов к базе данных используются логические выражения, написанные непосредственно на языке разработки приложения. Такой тип запросов носит название естественных запросов *(native queries — NQ)*.

В *C#* для написания естественных запросов могут использоваться объектно-ориентированные указатели — делегаты *(delegate)*, которые позволяют расширять описание классов за счет вновь создаваемых методов [10].

В качестве примера рассмотрим метод, представленный в виде анонимного делегата, который возвращает список объектов *(objects)* с идентификационным номером меньше 3:

```
IList<Objects3D> objects = db.Query<Objects3D>
(delegate(Objects3D example) {return example.
getid() < 3; });
```

В приведенном коде db.*Query* это ссылка на контейнер объектов базы данных, класс *IList* языка *C#* позволяет получить коллекцию объектов *(objects)* класса *Objects3D*.

Для итеративной обработки параметров массивов и коллекций в языке *C#* используется специальный оператор *foreach*.

Приведем код программы для получения параметров объектов из коллекции *objects*:

```
foreach (Objects3D example in objects){
  id = example.getid();
  namedev = example.getUnicalname();
  typesname = example.getTypesname();
  quant = example.getquant();
}
```

В результате выполнения приведенного запроса получим два объекта: «Рамка детектора № 1» и «Часы № 1». Параметры этих объектов, полученные при выполнении *NQ* запроса, приведены на *рис. 2*.

```
Результат NQ запроса :
_____________________________
| Имя объекта | Тип объекта |
_____________________________
|Рамка детектора №1 | Комплексный |
_____________________________________________________________
| Имя атрибута | Зн.атр.      | Имя атрибута | Зн.атр.      |
_____________________________________________________________
| Описание     | Металлодетектор | Количество | 1          |
_____________________________________________________________


_____________________________
| Имя объекта | Тип объекта |
_____________________________
|Часы №1      | Часы        |
_____________________________

| Имя атрибута | Зн.атр.      |
_____________________________
```

Рисунок 2. Результат выполнения NQ запроса
Figure 2. Result of NQ query execution

Использование объектно-ориентированной базы данных *db4o* для построения системы промышленного мониторинга позволяет встраивать функции обращения к базе данных непосредственно в программное обеспечение. Предлагаемый подход позволяет отказаться от установки внешней СУБД, что значительно облегчает переносимость программного обеспечения.

Программное обеспечение, необходимое для работы с *db4o* под *Windows*, включает в себя единственный файл динамически подключаемой библиотеки *Db4objects.Db4o.dll*, занимающий всего лишь 740 КБ.

Использование встроенной в программное обеспечение базы данных обеспечивает компактность программного кода, поскольку не требует использования хранимых процедур и триггерных функций.

Рассматриваемая система мониторинга станций московского метро была реализована на программной платформе *NET* компании *Microsoft*.

Разработанная на языке *C#* версия программы взаимодействия с объектно-ориентированной базой данных может быть достаточно просто переписана на *Java* версию, которая будет работать на любой платформе с поддержкой языка *Java*, включая ОС *Windows* и *Linux*.

В демонстрационной версии программы создавалось 100 объектов с переменным числом атрибутов от 1 до 10, которые записывались в последовательный файл. Для приведенных параметров объем базы данных составил 32 кб.

При использовании структуры данных, приведенной в *табл. 1–3*, для ОРСУБД *Postgres* объем базы данных составил 15 кб. Таким образом, на демонстрационном примере была достигнута пятикратная экономия памяти. Очевидно, что при увеличении количества объектов и количества атрибутов выигрыш в требуемой памяти будет возрастать.

Использование при построении запросов к базе данных методов, являющихся элементами программных классов, обеспечивает высокую производительность всей системы.

В настоящее время проводятся работы по апробации предложенной концепции построения системы мониторинга промышленных объектов на реальных данных.

Выводы

Для широкого спектра задач предложена методика хранения информации в объектно-ориентированной базе данных, непосредственно встроенной в систему мониторинга промышленных объектов.

На языке *C#* разработан прикладной интерфейс взаимодействия с базой данных системы индустриального мониторинга. Предложены методы хранения информации и построения запросов к объектно-ориентированной базе данных.

Предлагаемый подход позволяет значительно сократить вычислительные ресурсы, необходимые для реализации систем индустриального мониторинга, снижает время разработки и стоимость подобных систем.

ПРИСТАТЕЙНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Швецов Д.* SCADA-система GENESIS32 в сквозной автоматизации производства [Электронный ресурс] // ИСУП. 2007 № 4 (16). URL: http://isup.ru/articles/2/243 (дата обращения: 12.01.2021).
2. *Atkinson M., Bancilhon F., DeWitt D., Dittrich K., Maier D., Zdonik S*. The Object-Oriented Database System Manifesto. Proc. 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, Japan (1989). New York, Elsevier Science, 1990. 17 p.
3. *The Object* Data Standard: ODMG 3.0 / ed. by R.G.G. Cattell and Douglas K. Barry, with contributions by Mark Berler, Jeff Eastman, David Jordan, Craig L. Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez. Morgan Kaufmann Publishers, Inc., 2000. 288 p. ISBN 1558606475.
4. *Кузнецов С. Д.* Три манифеста баз данных: ретроспектива и перспективы. Доклад на международной научной конференции «Базы данных и информационные технологии XXI века» [Электронный ресурс] // CIT forum : сайт. URL: http://citforum.ru/database/articles/manifests/ (дата обращения: 12.01.2021).
5. S*tonebraker M., Moore D.* Object-Relational DBMSs: The Next Great Wave. Morgan Kaufmann Publishers, 1996. 216 p. ISBN 978-1558603974.
6. *Codd E. F.* A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. 1970. Vol. 13. No. 6. P. 377–387.
7. *Date C. J., Darwen H.* Foundation for Object-Relational Databases: The Third Manifesto. Addison-Wesley Professional, 1998. 496 p. ISBN-13 : 978-0201309782.
8. *Уорсли Д., Дрейк Дж.* PostgreSQL для профессионалов. СПб. : Питер, 2003. 496 с. ISBN 5-94723-337-1, 1-56592-846-6.
9. *Melton J.* Advanced SQL:1999. Understanding Object-Relational and Other Advanced Features. Morgan Kaufmann Publishers, 2003. 562 p. ISBN-13 : 978-1558606777.
10. *Бондаренко М. А., Бондаренко А. В., Бененсон М. З.* Аппаратно-программная платформа индустриального мониторинга // Вопросы радиоэлектроники. 2019. № 5. С. 20–27. DOI: 10.21778/2218-5453-2019-5-20-27.
11. *Paterson J., Edlich S., Horning H., Hörning R.* The Definitive Guide to db4o. Apress, 2006. 486 p. ISBN 978-1-59059-656-2.
12. *Шарп Д.* Microsoft Visual C#. Подробное руководство. СПб. : Питер, 2017. 848 с. ISBN 978-5-496-02372-6.

REFERENCES

1. Shvetsov D. [ProSoft SCADA GENESIS32 through automation]. *Zhurnal* ISUP, 2007;4(16):Available at: http://isup.ru/articles/2/243 (accessed: 12.01.2021). (In Russian).
2. Atkinson M., Bancilhon F., DeWitt D., Dittrich K., Maier D., Zdonik S. The Object-Oriented Database System Manifesto. Proc. 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, Japan (1989). New York, Elsevier Science, 1990, 17 p.
3. The Object Data Standard: ODMG 3.0 / ed. by R.G.G. Cattell and Douglas K. Barry, with contributions by Mark Berler, Jeff Eastman, David Jordan, Craig L. Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez. Morgan Kaufmann Publishers, Inc., 2000, 288 p. ISBN 1558606475.
4. Kuznezov S. D. [Three database manifestos: retrospect and perspectives. Report at the international scientific conference "Databases and information technologies of the XXI century"]. CIT forum. (In Russian). Available at: http://citforum.ru/database/articles/manifests/ (accessed: 12.01.2021).
5. Stonebraker M., Moore D. Object-Relational DBMSs: The Next Great Wave. Morgan Kaufmann Publishers, 1996, 216 p. ISBN 978-1558603974.
6. Codd E. F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 1970;13(6):377–387.
7. Date C. J., Darwen H. Foundation for Object-Relational Databases: The Third Manifesto. Addison-Wesley Professional, 1998, 496 p. ISBN-13 : 978-0201309782.
8. Worsley D., Drake J. Practical PostgreSQL. St. Petersburg, Piter Publ., 2003, 496 p. (In Russian). ISBN 5-94723-337-1, 1-56592-846-6.
9. Melton J. Advanced SQL:1999. Understanding Object-Relational and Other Advanced Features. Morgan Kaufmann Publishers, 2003, 562 p. ISBN-13 : 978-1558606777.
10. Bondarenko M. A., Bondarenko A. V., Benenson M. Z. Hardware-software platform of industrial monitoring. *Voprosy radioelectroniki*, 2019;5:20–27 (In Russian). DOI: 10.21778/2218-5453-2019-5-20-27.
11. Paterson J., Edlich S., Horning H., Hörning R. The Definitive Guide to db4o. Apress, 2006, 486 p. ISBN 978-1-59059-656-2.
12. Sharp D. Microsoft Visual C. Detailed guide. St. Petersburg, Piter Publ., 2017, 848 p. (In Russian). ISBN 978-5-496-02372-6.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Бененсон Михаил Залманович, к. т. н., доцент базовой кафедры № 254 – вычислительных комплексов ИИТ, ФГАОУ ВО «МИРЭА – Российский технологический университет», 119454, Москва, просп. Вернадского, д. 78, e-mail: mz_ben@mail.ru.

Алексеева Елена Александровна, к. т. н., доцент базовой кафедры ИИТ, ФГАОУ ВО «МИРЭА – Российский технологический университет»; консультант по научной работе, АО «Научно-исследовательский институт вычислительных комплексов им. М. А. Карцева», 117437, Москва, ул. Профсоюзная, д. 108, e-mail: ealeks@niivk.ru.

AUTHORS

Mihail Z. Benenson, PhD (Engineering), docent, department of computer systems of MIREA – Russian technological university, 78, prospekt Vernadskogo, Moscow, 119454, Russia, e-mail: mz_ben@mail.ru.

Elena A. Alekseeva, PhD (Engineering), docent, department of computer systems of MIREA – Russian technological university; scientific consultant, M. A. Kartsev Scientific and Research Institute of Computing Systems JSC, 108, ylitsa Profsoyuznaya, Moscow, 117437, Russia, e-mail: ealeks@niivk.ru.

Поступила 18.03.2020; принята к публикации 23.12.2020; опубликована онлайн 26.03.2021.
Submitted 18.03.2020; revised 23.12.2020; published online 26.03.2021.