Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра системного программирования

004.6(07)
Р462

ОБЪЕКТНЫЕ БАЗЫ ДАННЫХ

Задания для практических занятий
и методические указания по их выполнению

Челябинск
Издательский центр ЮУрГУ
2013

УДК 004.652.5(075.8)
Р462

Одобрено учебно-методической комиссией
факультета вычислительной математики и информатики

Методическое пособие подготовлено в соответствии с ФГОС ВПО 3-го поколения по образовательному направлению 010300.62 «Фундаментальная информатика и информационные технологии».

Рецензент:
П.С. Костенецкий

Р462 **Объектные базы данных:** задания для практических занятий и методические указания по их выполнению / сост.: Т.В. Речкалов, М.Л. Цымблер. – Челябинск: Издательский центр ЮУрГУ, 2013. – 28 с.

Методическое пособие предназначено для студентов, обучающихся по направлению «Фундаментальная информатика и информационные технологии» при изучении дисциплины «Объектные базы данных». Пособие содержит тексты заданий и указания по их выполнению по следующим темам: Объектно-ориентированные базы данных, Объектно-реляционные возможности СУБД Oracle, XML СУБД Sedna, документ-ориентированная СУБД MongoDB, графовая СУБД Neo4j, геоСУБД PostGIS. Также в пособии приведены темы докладов для самостоятельной работы и список рекомендуемой литературы.

УДК 004.652.5(075.8)

1. ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ РАБОТ

1.1. Основные понятия объектных баз данных

Задание 1. Подготовка эссе (2 часа)

Подготовьте эссе на тему «Основные понятия объектных баз данных».

См. *Указания к заданию 1. Подготовка эссе* на с. 8.

Задание 2. Подготовка доклада (самостоятельная работа)

Подготовьте 10-минутное сообщение об одной объектной СУБД из следующего списка:

1. Документ-ориентированная СУБД
   a) CouchDB
   b) IBM Lotus Notes
   c) RavenDB
   d) BaseX
2. Графовая СУБД
   a) DEX
   b) OrientDB
   c) Neo4j
   d) Google Pregel
3. Хранилище «ключ-значение»
   a) Redis
   b) Chordless
   c) LevelDB
   d) Azure Table Storage
4. Колоночная СУБД
   a) Cassandra
   b) Cloudera
5. Объектная СУБД
   a) Versant
   b) db4o
6. XML СУБД
   a) BaseX
   b) Berkeley DB XML
   c) eXist
7. СУБД Oracle NoSQL Database.
8. СУБД для обработки научных данных SciDB.

См. Указания к заданию 2. Подготовка доклада на с. 8.

1.2. Объектно-ориентированные базы данных

Задание 3. Разработка описания классов на языке ODL (2 часа)

Разработайте описание классов предметной области, изображенной на рис. 1, на языке ODL (Object Definition Language).

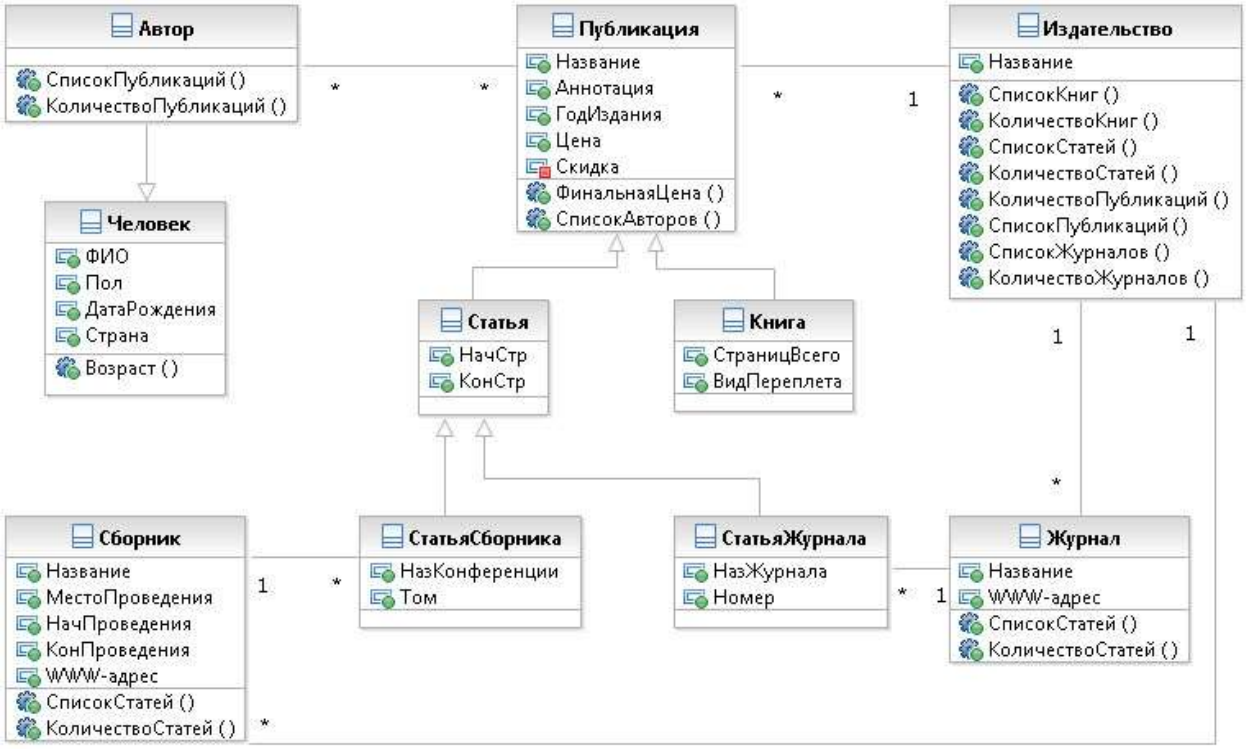

Рис. 1. Диаграмма классов предметной области «Интернет магазин научных публикаций»

См. Указания к заданию 3. Разработка описания классов на языке ODL на с. 8.

1.3. Объектно-реляционная СУБД Oracle

Разработайте приложение, предназначенное для ведения базы данных о клиентах, товарах и заказах, используя объектно-реляционные возможности СУБД Oracle. Диаграмма классов предметной области представлена на рис. 2.

Задание 4. Разработка классов (2 часа)

1. Разработайте классы (объектные типы СУБД Oracle), реализующие классы предметной области и связи между ними.
2. Создайте объектные таблицы для хранения экземпляров классов.

См. *Указания к заданию 4. Разработка классов* на с. 9.

Задание 5. Разработка объектно-реляционных запросов (6 часов)

1. Разработайте запросы на добавление данных в созданные объектные таблицы.

2. Разработайте и протестируйте запросы на выборку данных из созданных таблиц.

См. *Указания к заданию 5. Разработка объектно-реляционных запросов* на с. 11.

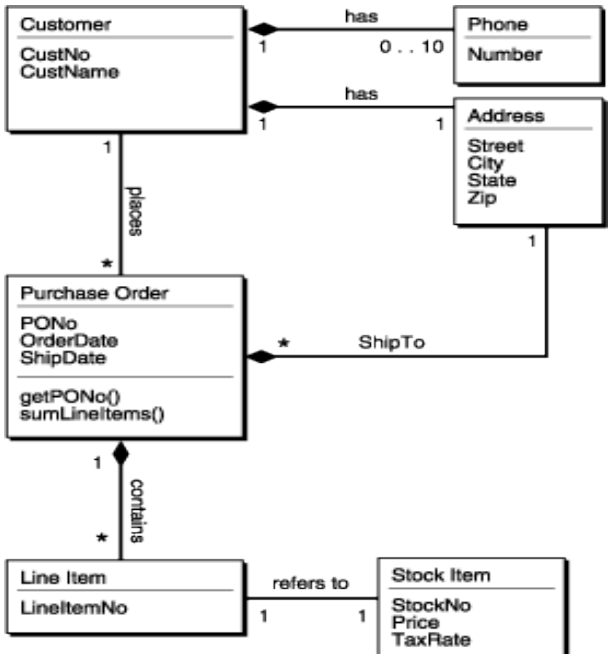

Рис. 2. Диаграмма классов предметной области «Заказы»

1.4. XML СУБД Sedna

Задание 6. Разработка приложения XML СУБД Sedna (6 часов)

Разработайте приложение для XML СУБД Sedna, которое предназначено для ведения XML базы данных о книгах домашней библиотеки. Пример базы данных представлен на рис. 3.

```
<?xml version="1.0" encoding="koi8-r" ?>
<all_books>
<!--  данные об одной или более книгах   -->
    <book id="00001">
      <genre>фантастика</genre>
      <author>Стругацкий А.Н.</author> <author>Стругацкий Б.Н.</author>
      <title>Трудно быть богом</title>
      <year>2003</year> <city>Москва</city>
      <publisher>Мода-Пресс</publisher>
      <annotation>Классика отечественной фантастики.</annotation>
      <pages>321</pages> <cover>твердый</cover> <price>100.00</price>
    </book>
    <book id="00002">
      <genre>компьютеры</genre>
      <author>Брукс Ф.</author>
      <title>Мифический человеко-месяц или как создаются программные систе-
    мы</title>
      <year>2001</year> <city>Москва</city>
      <publisher>Символ-Плюс</publisher>
      <annotation>Библия для руководителя программного проекта.</annotation>
      <pages>304</pages> <cover>твердый</cover> <price>248.00</price>
    </book>
  </all_books>
```

Рис. 3. Пример XML-базы данных о книгах домашней библиотеки

См. *Указания к заданию 6. Разработка приложения XML СУБД Sedna* на с. 12.

1.5. Документ-ориентированная СУБД MongoDB

Задание 7. Разработка базы данных с использованием СУБД MongoDB (2 часа)

Используя СУБД MongoDB, разработайте базу данных, предназначенную для хранения логов веб-сервера. Лог включает в себя следующие поля: адрес ресурса (URL), IP-адрес пользовательского компьютера, отметка времени начала просмотра ресурса, длительность просмотра ресурса.

См. *Указания к заданию 7. Разработка базы данных с использованием СУБД* MongoDB на с. 15.

Задание 8. Разработка запросов в СУБД MongoDB (4 часа)

3. Разработайте и протестируйте запросы на выборку данных из созданных коллекций.
4. Разработайте и протестируйте функции MapReduce для анализа посещаемости ресурсов web-сервера.

См. *Указания к заданию 8. Разработка запросов в СУБД MongoDB* на с. 15.

1.6. Графовая СУБД Neo4j

Задание 9. Разработка базы данных социальной сети (2 часа)

Используя графовую СУБД Neo4j, разработайте базу данных модельной социальной сети. Социальная сеть имеет два типа узлов: человек и группа. Узел человек имеет следующие атрибуты: ФИО, пол, возраст, город. Узел группа имеет следующие атрибуты: название (например: спорт, игры, компьютеры). Кроме того узлы имеют атрибут "тип". Семантика связи между узлами – дружеские отношения соответствующих персон («А является другом Б»), участие в группе («А является участником группы Б»).

См. *Указания к заданию 9. Разработка базы данных социальной сети* на с. 21.

Задание 10. Разработка запросов к графам (2 часа)

Разработайте и протестируйте запросы на выборку данных из созданной графовой базы данных.

См. *Указания к заданию 10. Разработка запросов к графам* на с. 21.

Задание 11. Визуализация графа (2 часа)

Разработайте и протестируйте запросы на обновление свойств узлов графа и выполните с их помощью визуализацию графа.

1.7. ГеоСУБД PostGIS

Используя СУБД PostGIS, разработайте картографическое приложение для получения статистических сведений о субъектах Российской Федерации.

Для выполнения данного задания в СУБД Postgres установлена картографическая база данных России с комплектом карт 4 регионов.

Задание 12. Разработка пространственных запросов (4 часа)

Разработайте и протестируйте запросы на выборку данных с использованием функций PostGIS.

См. *Указания к заданию 12. Разработка пространственных запросов* на с. 25.

Задание 13. Визуализация геоданных с помощью MapServer (2 часа)

Выполните настройку визуализации геопространственных данных при помощи MapServer и PostGIS.

См. *Указания к заданию 13. Визуализация геоданных с помощью MapServer* на с. 25.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

2.1. Основные понятия объектных баз данных

Указания к заданию 1. Подготовка эссе

1. При подготовке эссе используйте следующие материалы [1–4].
2. Подготовьте презентацию в формате PowerPoint или PDF. Примерный план:
   1) Причины появления систем баз данных, основанных не на реляционной модели данных.
   2) 1-й, 2-й и 3-й манифесты систем баз данных: авторы, основные идеи, влияние на технологии баз данных.
   3) Объектно-ориентированные и объектно-реляционные системы баз данных: основные особенности, сходства и различия, примеры.
   4) Движение NoSQL: основные идеи, классификация СУБД.

Указания к заданию 2. Подготовка доклада

1. При подготовке доклада используйте материалы сайта [4].
2. Подготовьте презентацию доклада в формате PowerPoint или PDF (не более 10 слайдов, включая титульный). Текст доклада впишите в заметки к слайдам. Примерный план доклада:
   1) Краткая история создания продукта. Примеры реальных задач, для решения которых используется продукт.
   2) Системная архитектура (краткое описание).
   3) Язык СУБД (краткое описание синтаксиса и примеры запросов).
   4) Особенности использования СУБД.
   5) Литература и web-ресурсы о СУБД, использованные при подготовке доклада.

2.2. Объектно-ориентированные базы данных

Указания к заданию 3. Разработка описания классов на языке ODL

1. Изучите синтаксис и семантику языка ODL, используя источники [5, 6]:
2. Разработайте для каждого класса описание атрибутов, методов и связей в соответствии с рис. 1. Для каждого класса определите из имеющихся атрибутов ключ (возможно, составной), дополняющий ОИД.
3. При описании связей для каждой связи убедитесь в наличии инверсной ей связи.
4. В описании методов классов предусмотрите исключительные ситуации.
5. Определите экстенты для каждого класса.

2.3. Объектно-реляционная СУБД Oracle

В качестве основного источника справочной информации по СУБД Oracle и языку PL/SQL используйте ресурс [7].

Указания к заданию 4. Разработка классов

1. Разработайте интерфейсы классов, изображенных на рис. 2, с помощью команды `CREATE OR REPLACE TYPE … AS OBJECT`, учитывая следующие замечания:
   1) Для именования классов, которые по своей природе не являются массивами, используйте суффикс `_objtyp` (например: `StockItem_objtyp`, `LineItem_objtyp` и др.).
   2) Для именования классов, которые по своей природе являются массивами, используйте суффикс `_vartyp` и встроенный в PL/SQL тип `varray` (класс `PhoneList_vartyp`).
   3) Связи между классами реализуйте с помощью ссылочных типов `REF` (сравните ссылочные типы в объектно-реляционной модели с внешними ключами в реляционной модели данных).
   4) В определение класса `Customer_objtyp` включите ORDER-метод вида
      `compareCustOrders(x IN Customer_objtyp) RETURN INTEGER`
      для сравнения двух экземпляров класса.
   5) Разработайте тип `LineItemList_ntabtyp`, реализующий вложенную таблицу экземпляров класса `LineItem_objtyp`, с помощью команды `CREATE OR REPLACE TYPE … AS TABLE OF …`.
   6) В определение класса `PurchaseOrder_objtyp` включите атрибут, имеющий тип `LineItemList_ntabtyp`. Метод `getPONo` определите как MAP-метод для идентификации экземпляра при сравнении экземпляров класса.
2. Разработайте реализацию методов классов с помощью команды `CREATE OR REPLACE TYPE BODY … AS …`.
3. Разработайте объектную таблицу `Customer_objtab` в соответствии со схемой на рис. , используя команду `CREATE TABLE … OF …`. С помощью ключевых слов `OBJECT IDENTIFIER IS PRIMARY KEY` укажите, что первичный ключ будет использоваться в качестве ОИД объектов-строк.
4. Разработайте объектную таблицу `PurchaseOrder_objtab` в соответствии со схемой на рис. , используя команду `CREATE TABLE … OF …`. С помощью ключевых слов `OBJECT IDENTIFIER IS PRIMARY KEY` укажите, что первичный ключ будет использоваться в качестве ОИД объектов-строк. Определите поле `LineItemList_ntab` как хранимую вложенную таблицу `PoLine_ntab` с помощью ключевых слов `NESTED TABLE … STORE AS …` и определите первичный ключ вложенной таблицы как составной, включающий в себя, в том числе скрытое системное поле `NESTED_TABLE_ID`.

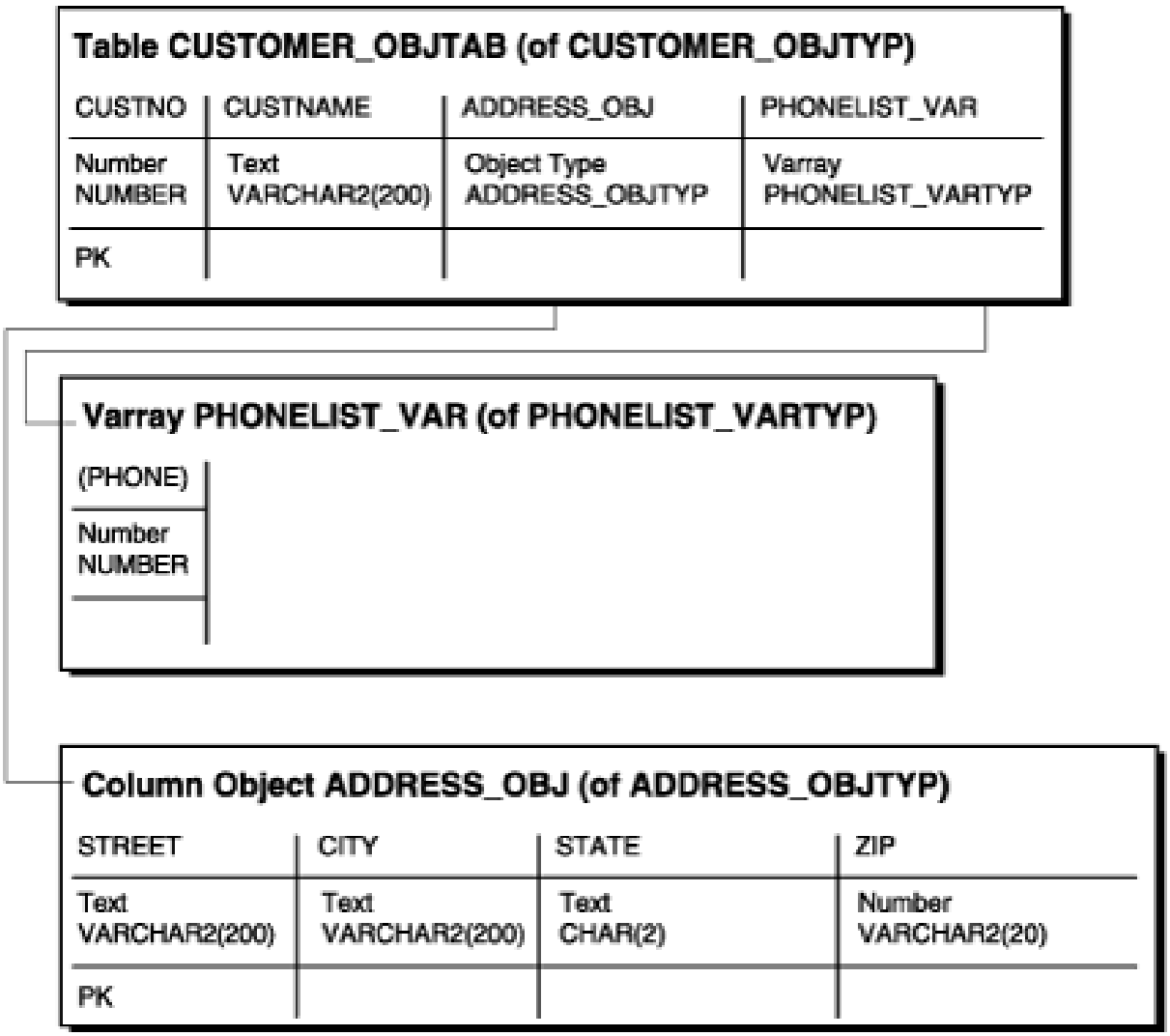

Рис. 4. Объектно-реляционное представление таблицы Customer_objtab

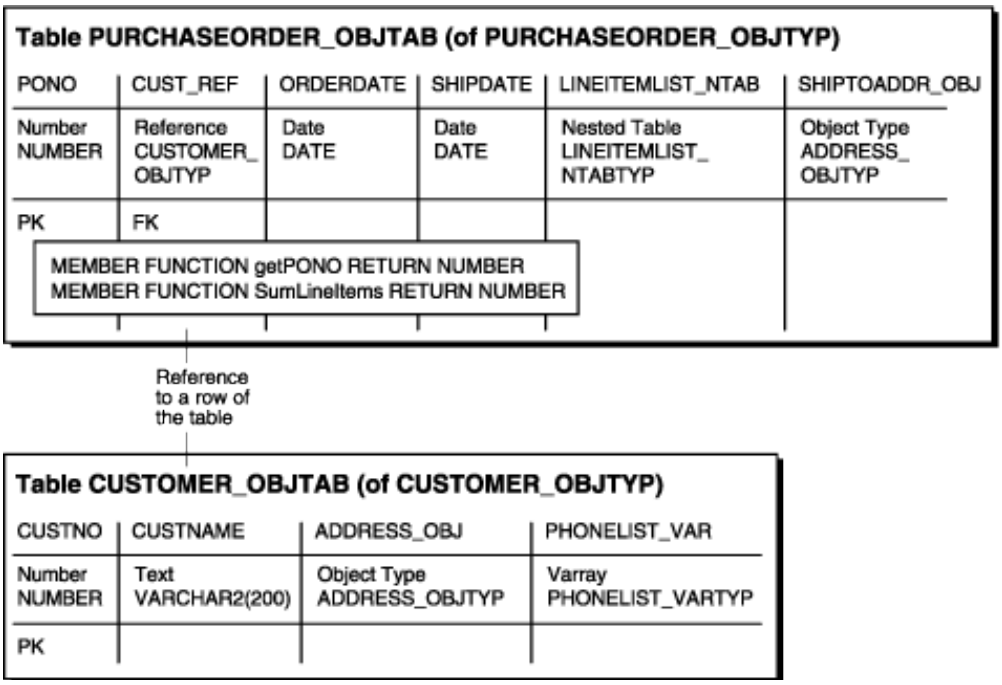

Рис. 5. Объектно-реляционное представление таблицы PurchaseOrder_objtab

5. Добавьте ограничение целостности для вложенной таблицы `PoLinen_ntab` в соответствии со схемой на рис. 6, используя ключевые слова `SCOPE FOR (…) IS … .`

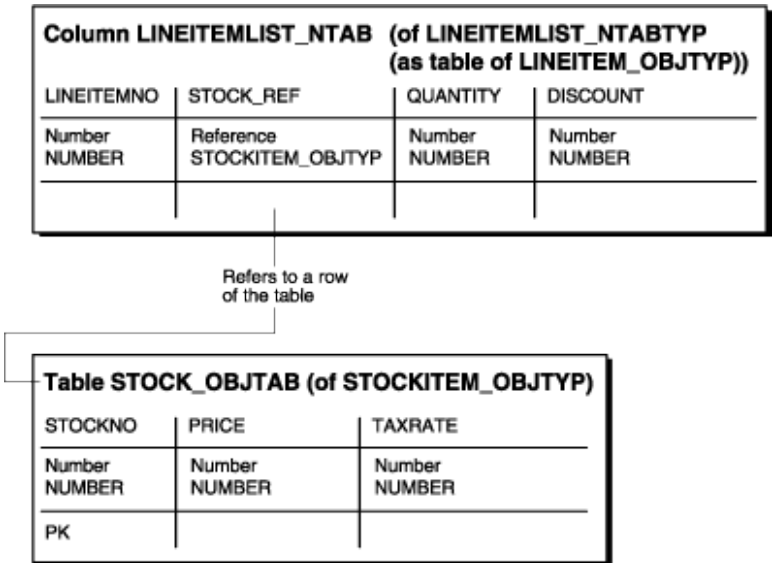

Рис. 6. Объектно-реляционное представление таблицы LineItemList_ntab

Указания к заданию 5. Разработка объектно-реляционных запросов

1. Разработайте и протестируйте запросы, добавляющие следующие данные в таблицы:

   1) `Stock_objtab`:

StockNo	Price	TaxRate
1004	6750.00	2
1011	4500.23	2
1534	2234.00	2
1535	3456.23	2
1734	7843.28	3

   2) `Customer_objtab`:

CustNo	CustName	Address	PhoneList
1	Bruce Wayne	('2 Avocet Drive', 'Redwood Shores', 'CA', '95054')	('415-555-0102')
2	Tony Stark	('323 College Drive', 'Edison', 'NJ', '08820')	('609-555-0190', '201-555-0140')
3	Peter Parker	('177 Sunset Blvd', 'LA', 'CA', '90210')	NULL
4	Clark Kent	('85 Infinite Loop', 'Cupertino', 'CA', '95014')	('123-456-7890', '987-654-3210')

   3) `PurchaseOrder_objtab`:

PONo	Cust_ref	OrderDate	ShipDate	LineItemList_ntab	ShipToAddr_obj
1	1	SYSDATE	SYSDATE+7	NULL	NULL
2	2	SYSDATE	SYSDATE+7	NULL	NULL
3	3	SYSDATE	SYSDATE+7	NULL	NULL
4	4	SYSDATE	SYSDATE+7	NULL	NULL

2. Разработайте и протестируйте запросы, добавляющие данные во вложенную таблицу `LineItemList_ntab`.
3. Разработайте и протестируйте запросы, выполняющие выборку данных из таблиц:
   1) Выдать упорядоченный список всех клиентов.
   2) Выдать упорядоченный список всех заказов с указанием суммы каждого заказа.
   3) Выдать для каждого заказа его данные, включая данные о клиенте.
   4) Выдать данные о заказе и входящих в него позициях для товара с заданным кодом.
   5) Выдать максимальную скидку среди всех позиций по всем заказам.

2.4. XML СУБД Sedna

В качестве основного источника справочной информации по XML СУБД Sedna и языку XQuery используйте ресурс [8].

Указания к заданию 6. Разработка приложения XML СУБД Sedna

1. Разработайте спецификацию DTD или XML-схему (на выбор) для описания данных о книгах домашней библиотеки.
2. Разработайте тестовую XML базу данных, содержащую не менее 15 записей (данные должны быть осмысленными).
3. Разработайте следующие запросы на языке XQuery. Параметры запроса необходимо реализовать как переменные (например, `$minYear` и `$maxYear` для указания диапазона лет):
   1) Выдать список книг (ИД книги, первый автор, название, город, издательство, год, вид обложки, цена, скидка).
   2) Выдать список авторов (без повторений).
   3) Выдать список жанров (без повторений).
   4) Выдать список городов и издательств (без повторений).
   5) Выдать список книг указанного автора (первый автор, название, город, издательство, год, вид переплета, цена).
   6) Выдать список книг указанного издательства (первый автор, название, город, издательство, год, вид переплета, цена).
   7) Выдать список книг указанного жанра (первый автор, название, город, издательство, год, вид переплета, цена).
   8) Выдать список книг, изданных в указанном диапазоне лет (первый автор, название, город, издательство, год, вид переплета, цена).
   9) Выдать список книг, имеющих цену в указанном диапазоне (первый автор, название, город, издательство, год, вид переплета, цена).
   10) Выдать общее количество и общую стоимость книг указанного автора.
   11) Выдать количество книг в твердом и мягком переплетах.

2.5. Документ-ориентированная СУБД MongoDB

В качестве основного источника справочной информации по документ-ориентированной СУБД MongoDB используйте приведенную ниже краткую справку и ресурсы [9-10].

Краткая справка по СУБД MongoDB

СУБД MongoDB реализует документ-ориентированную модель данных. *Документ* (см. пример на рис. 7) – это набор, состоящий из имен и значений свойств. Значение может быть простым типом, массивом или другим документом.

Ключевое отличие реляционной модели от документ-ориентированной в том, что при использовании реляционной модели данные раскладываются по нескольким таблицам с фиксированной структурой, что приводит к необходимости выполнения соединений для работы с взаимосвязанными данными из разных таблиц. Документ-ориентированная модель представляет данные в агрегированной форме, то есть работает с объектом как с единым целым: все данные, относящиеся к сущности, хранятся в одном объекте базы данных.

Второе важное отличие – структура объектов в MongoDB не декларируются схемой данных. Наборы объектов хранятся в виде коллекций и каждый объект может иметь собственную структуру, но данный подход редко встречается на практике. Обычно объекты в коллекции имеют схожую структуру с небольшими отличиями. Преимуществом бессхемного подхода является высокая гибкость при изменении и уточнении предметной области.

```
{
   _id: ObjectID('4bd9e8el7cefd644108961bb',
   title: 'Adventures in Databases',
   url: 'http://example.com/databases.txt'
   author: 'msmith',
   vote.count: 20,
   tags: ['databases', 'mongodb', 'indexing'],
   image:
   {
      url: 'http://example.com/db.jpg',
      caption: '',
      type: 'jpg',
      size: 75381,
      data: "Binary"
   },

   comments:
   [
      {
         user: 'bjones',
         text: 'Interesting article!'
      },
      {
         user: 'blogger',
         text: 'related article at http://example.com/db/db.txt'
      }
   ]
}
```

Рис. 7. Документ, представляющий статью на социальном новостном сайте

Следующим свойством MongoDB является возможность работы с произвольными запросами на уровне гибкости языка реляционных баз данных SQL. Поддержка произвольных запросов означает, что нет принципиальных ограничений, требуемых для выполнения запроса. Например, данным свойством не обладают NoSQL СУБД организованные по принципу key-value (ключ-значение). В таких СУБД запросы могут выполняться только по ключу. Следствием поддержки произвольных запросов является необходимость в развитой системе вторичных индексов. С их помощью достигается высокая скорость исполнения запросов. Индексы MongoDB реализованы на основе B-деревьев и поддерживают те же способы организации, что и реляционные СУБД.

MongoDB содержит механизм журналирования, при использовании которого пользователь получает более высокую гарантию сохранения записей ценой снижения производительности. При необходимости данную подсистему можно отключить.

MongoDB поддерживает технологию горизонтального масштабирования с использованием автосегментирования и дублирования реплик данных. Данный механизм гарантирует автоматическое восстановление без точки общего отказа.

Основные *программные артефакты MongoDB*:

1) `mongod` – исполняемый файл сервера баз данных;
2) `mongo` – командная оболочка MongoDB. Язык оболочки JavaScript.
3) `mongodump` и `mongorestore` – стандартные утилиты резервного копирования и восстановления базы данных
4) `mongoexport` и `mongoimport` – утилиты экспорта и импорта файлов в форматах JSON, CSV и TSV
5) `mongosniff` – анализатор протокола для просмотра команд, посылаемых серверу базы данных.
6) `mongostat` – аналог `iostat`; постоянно опрашивает MongoDB и операционную систему для выдачи полезной статистики, в том числе количества операций (вставки, выборки, обновления, удаления и др.) в секунду, объема выделенной виртуальной памяти и числа подключений к серверу.

Пример	Семантика
`use <db name>`	Переключение контекста на указанную базу данных (users)
`db.users.insert({username: "Smith"})`	Вставка документа в коллекцию users
`db.users.find()`	Получение содержимого коллекции users
`db.users.count()`	Получение размера коллекции users
`db.users.find({username: "Jones"})`	Поиск с использованием селектора запроса (документ, с которым сравниваются все документы в коллекции)
`db.users.update({username: "Smith"}, {$set: {country: "Canada"}})`	Во все документы коллекции users, содержащие имя пользователя Смит, будет добавлена информация о стране.

Пример	Семантика
`db.users.update({username: "Smith"}, {$unset: {country: "Canada"}})`	Из всех документов коллекции users, содержащих имя пользователя Смит, будет удалена информация о стране.
`db.foo.remove()`	Удаление всех документов из коллекции foo (аналог SQL команды TRUNCATE)
`db.users.drop()`	Удаление коллекции вместе со всеми индексами (аналог SQL команды DROP)

Указания к заданию 7. Разработка базы данных с использованием СУБД MongoDB

1. Разработайте консольную утилиту для преобразования лога веб-сервера в формате CSV (Comma Separated Values), в формат JSON. Лог должен содержать поля со следующими названиями: URL, IP, timeStamp, timeSpent.
2. Разработайте запросы для загрузки полученных данных в формате JSON в СУБД MongoDB.

Указания к заданию 8. Разработка запросов в СУБД MongoDB

1. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средствами выборки:
   1) Выдать упорядоченный список URL ресурсов.
   2) Выдать упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL.
   3) Выдать упорядоченный список URL ресурсов, посещенных в заданный временной период.
   4) Выдать упорядоченный список URL ресурсов, посещенных пользователем с заданным IP-адресом.
2. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средства программирования на основе парадигмы MapReduce:
   1) Выдать список URL ресурсов с указанием суммарной длительности посещения каждого ресурса, упорядоченный по убыванию.
   2) Выдать список URL ресурсов с указанием суммарного количества посещений каждого ресурса, упорядоченный по убыванию.
   3) Выдать список URL ресурсов с указанием количества посещений каждого ресурса в день за заданный период, упорядоченный URL ресурса и убыванию количества посещений.
   4) Выдать список IP-адресов с указанием суммарного количества и суммарной длительности посещений ресурсов, упорядоченный по адресу, убыванию количества и убыванию длительности.

2.6. Графовая СУБД Neo4j

В качестве основного источника справочной информации по графовой СУБД Neo4j и языку Cypher используйте приведенную ниже краткую справку и ресурс [11].

Краткая справка по языку Cypher

Cypher представляет собой декларативный язык запросов СУБД Neo4j. С помощью Cypher возможно сопоставление паттерны узлов и отношений в графе для извлечения и модификации информации, поддержка системы идентификаторов для обозначения именованных узлов, граничных узлов и их параметров, а так же создание, обновление и удаление узлов, связей и их свойств.

При указании на части паттернов используются условные имена (называемые *идентификаторами*).

Пример: `START n=node(1) MATCH n-->b RETURN b`

Здесь *n* и *b* являются идентификаторами. Идентификаторы регистрозависимы, могут содержать буквы, числа и символы подчеркивания, но должны начинаться с буквы. Если идентификатор должен содержать другие символы, то его необходимо заключить в кавычки.

Комментарии добавляются с помощью двойного слэша.

Допустимые *выражения* в Cypher:

1) *Численный литерал* (целый или вещественный): `13, 40000, 3.14`.
2) *Строковый литерал*: `"Hello", 'World'`.
3) *Булевский литерал*: `true, false, TRUE, FALSE`.
4) *Идентификатор*: `n, x, rel, myFancyIdentifier, `A name with weird stuff in it[]!``.
5) *Свойство*: `n.prop, x.prop, rel.thisProperty, myFancyIdentifier.`(weird property name)``.
6) *Nullable-свойство* (свойство со знаком вопроса или восклицания): `n.prop?, rel.thisProperty!`.
7) *Параметр*: `{param}, {0}`
8) *Коллекция* выражений:
   `["a", "b"], [1,2,3], ["a", 2, n.property, {param}], [ ]`.
9) *Вызов функции*: length(p), nodes(p).
10) *Агрегирующая функция*: avg(x.prop), count(*).
11) *Типы отношений*: `:REL_TYPE, :`REL TYPE``, `:REL1|REL2`.
12) *Паттерн пути*: `a-->()<--b`.
13) *Предикат*: `a.prop = "Hello", length(p) > 10, has(a.name)`

В Cypher поддерживаются следующие *форматы запросов*:

Вид запроса	Синтаксис запроса
Чтение данных	START [MATCH] [WHERE] RETURN [ORDER BY] [SKIP] [LIMIT]
Запись данных	CREATE [UNIQUE]* [SET|DELETE|FOREACH]* [RETURN [ORDER BY] [SKIP] [LIMIT]]
Чтение и запись данных	START [MATCH] [WHERE] [CREATE [UNIQUE]]* [SET|DELETE|FOREACH]* [RETURN [ORDER BY] [SKIP] [LIMIT]]

Язык запросов Cypher состоит из следующих *предложений*:

1) `START`: начальные точки запроса в графе, полученные с помощью индексов или идентификаторов узлов (ID).

Пример	Семантика
`START n=node(*)`	Стартовая точка будет установлена на все узлы.
`START n=node({ids})`	Стартовая точка будет установлена на один или более узлов, указанных в списке идентификаторов.
`START n=node({id1}), m=node({id2})`	Несколько стартовых точек.

2) `MATCH`: графовый паттерн, удовлетворяющий запросу, ограниченный начальными точками из оператора `START`. Любой паттерн может использоваться в предложении `MATCH`, за исключением паттернов, содержащих карты свойств.
   Пример: `MATCH (n)-->(m)`
3) `WHERE`: критерий фильтрации.
   Пример: `WHERE n.property <> {value}`
4) `RETURN`: возвращаемое значение.

Пример	Семантика
`RETURN *`	Вернуть все идентификаторы.
`RETURN n AS columnName`	Использовать псевдоним в качестве имени колонки.
`RETURN DISTINCT n`	Вернуть уникальные строки.
`ORDER BY n.property`	Отсортировать по свойству.

Пример	Семантика
`ORDER BY n.property DESC`	Отсортировать по свойству в убывающем порядке.
`SKIP {skip_number}`	Пропустить заданное количество строк в выдаче.
`LIMIT {limit_number}`	Ограничить выдачу заданным количеством строк.
`SKIP {skip_number} LIMIT {limit_number}`	Пропустить сколько-то строк, а потом ограничить выдачу.

5) `CREATE`: создание узлов и связей.

Пример	Семантика
`CREATE (n {name: {value}})`	Создать узел с заданными свойствами.
`CREATE n = {map}`	Создать узел с заданными свойствами.
`CREATE n = {collectionOfMaps}`	Создать узел с заданными свойствами.
`CREATE (n)-[r:KNOWS]->(m)`	Создать связь заданного типа и направления; присвоить ему идентификатор.
`CREATE (n)-[:LOVES {since: {value}}]->(m)`	Создать отношение из заданного типа, направления и набора свойств.

6) `DELETE`: удаление узлов, связей и свойств.
   Примеры: `DELETE n, r` - удалить узел или отношение. `DELETE n.property` - удалить свойство.
7) `SET`: установка значений свойств.
   Примеры: `SET n.property={value}` - обновить или создать свойство.
   `SET n={map}` - задать все свойства (удалит все существующие свойства).
8) `FOREACH`: выполнение операций обновления для каждого элемента из списка.
9) `WITH`: разбивает запрос на несколько независимых частей.
   При выполнении запросов на языке Cypher необходимо задавать искомые паттерны отношений между узлами. Паттерны составляются по следующим правилам:

Правило	Семантика
`(n)-->(m)`	Проверка на существование отношения от n к m.
`(n)--(m)`	Проверка на существование отношения между n и m (направление игнорируется).
`(m)<-[:KNOWS]-(n)`	Проверка на существование отношения от n к m типа KNOWS.
`(n)-[:KNOWS|LOVES]->(m)`	Проверка на существование отношения от n к m типа KNOWS или LOVES.
`(n)-[r]->(m)`	Присвоить идентификатор отношению.
`(n)-[r?]->(m)`	Опциональное отношение.
`(n)-[*1..5]->(m)`	Переменная для пути отношений между n и m.
`(n)-[*]->(m)`	Нет ограничений на глубину отношений.
`(n)-[:KNOWS]->(m {property: {value}})`	Сопоставить набор свойств в предложениях CREATE или CREATE UNIQUE.

В языке Cypher предусмотрены *функции для работы с коллекциями и массивами*:

Функция	Семантика
`NODES(path)`	Список узлов в указанном пути.
`RELATIONSHIPS(path)`	Отношения в указанном пути.
`EXTRACT(x IN collection: x.prop)`	Коллекция значений выражения для каждого элемента коллекции.
`FILTER(x IN coll: x.prop <> {value})`	Возвращает коллекцию элементов, для которых предикат истинен.
`TAIL(collection)`	Все элементы коллекции, кроме первого.
`RANGE({begin}, {end}, {step})` `REDUCE(str = "", n IN coll : str + n.prop)`	Создать набор чисел с указанным шагом. Вычислить выражение для каждого элемента коллекции и аккумулировать результат.
`FOREACH (n IN coll : SET n.marked = true)`	Выполнить операцию обновления для каждого элемента в коллекции.

Вспомогательные функции:

Функция	Семантика
`LENGTH(collection)`	Возвращает длину коллекции.
`TYPE(a_relationship)`	Строковое представление типа отношения.
`COALESCE(n.property?, {defaultValue})`	Первое ненулевое выражение.
`HEAD(collection)`	Первый элемент коллекции.
`LAST(collection)`	Последний элемент коллекции.
`TIMESTAMP()`	Количество прошедших миллисекунд от полуночи 1 января 1970 года по … .
`ID(node_or_relationship)`	Внутренний идентификатор отношения или узла.

При работе с примитивами языка Cypher используется развитая система *предикатов*:

Предикат	Семантика
`n.property <> {value}`	Использование операторов сравнения.
`HAS(n.property) AND n.property = {value}`	Использование булевых операторов для комбинирования предикатов.
`HAS(n.property)`	Использование функций.
`identifier IS NULL`	Проверка на null.
`n.property? = {value}`	Возвращает истину, если свойство не существует.
`n.property! = {value}`	Возвращает ложь, если свойство не существует.
`n.property =~ {regex}`	Регулярные выражения.
`(n)-[:KNOWS]->(m)`	Убедиться, что паттерн имеет хотя бы одно совпадение.
`n.property IN [{val1}, {val2}]`	Проверка, что элемент существует в коллекции.

В языке Cypher также имеются *функции для вычисления предикатов над коллекциями*:

Предикат	Семантика
`ALL(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется для всех элементов коллекции.
`ANY(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется хотя бы для одного элемента коллекции.
`NONE(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат не выполняется для всех элементов коллекции.
`SINGLE(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется строго для одного элемента коллекции.

Операторы: математические +, -, *, /, %; сравнения =, <>, <, >, <=, >=, булевы AND, OR, NOT; строковые +; операторы коллекций: +, IN; регулярные выражения: =~; операторы свойств ?, !.

Агрегирующие функции: COUNT, SUM, AVG, MAX, MIN, COLLECT, PERCENTILE_DISC, PERCENTILE_CONT.

Математические функции: ABS, ROUND, SQRT, SIGN.

Строковые функции: STR, REPLACE, SUBSTRING, LEFT, RIGHT, LTRIM, RTRIM, TRIM, LOWER, UPPER.

Указания к заданию 9. Разработка базы данных социальной сети

1. Разработайте скрипт с запросами на языке Cypher, которые загружают базу данных социальной сети в СУБД Neo4j. База должна содержать не менее 10 узлов и не менее 15 связей между ними.
2. Загрузите базу данных в СУБД Neo4j при помощи консольного клиента (Neo4jShell.bat).

Указания к заданию 10. Разработка запросов к графам

1. Изучите справочную информацию о языке Cypher по теме Reading Clauses и разработайте следующие запросы:
   1) Выдать упорядоченный список ФИО персон.
   2) Выдать список ФИО мужчин с указанием возраста, упорядоченный по убыванию возраста.
   3) Выдать упорядоченный список ФИО друзей персоны заданными ФИО.
   4) Выдать упорядоченный список ФИО друзей друзей персоны заданными ФИО.
   5) Выдать упорядоченный по алфавиту список ФИО персон, в котором для каждой персоны указано количество друзей.
2. Изучите справочную информацию о языке Cypher по теме Functions и разработайте следующие запросы:
   1) Выдать упорядоченный список групп социальной сети.
   2) Выдать упорядоченный список групп персоны с заданными ФИО.
   3) Выдать список групп социальной сети с указанием количества членов каждой группы, упорядоченный по убыванию количества членов группы.
   4) Выдать список ФИО персон, в котором для каждой персоны указано количество групп, в которые она входит, упорядоченный по убыванию количества групп.
   5) Выдать общее количество групп, в которых состоят друзья друзей персоны с заданными ФИО.
3. Дополнительное. Дополните узлы пользователей социальной сети массивом опубликованных записей этих пользователей (аналог истории твитов или статусов) и разработайте следующие запросы:
   1) Вывести список записей персоны с заданными ФИО.

2) Вывести список средних длин записи для каждого пользователя, отсортированный по убыванию средней длины.
   Указание: вычислить сумму длин всех записей с использованием функции REDUCE.
3) Вывести все записи в социальной сети, длиннее некоторой величины. Указание: используйте функцию Filter.
4) Выдать список ФИО персон, в котором для каждой персоны указано количество записей данной персоны, упорядоченный по убыванию количества записей.
5) Вывести все записи друзей друзей персоны с заданными ФИО.

Указания к заданию 11. Визуализация графа

1. Изучите справочную информацию по средствам визуализации Neo4j. Отобразите граф.
2. Добавьте свойство *friend* всем друзьям одной персоны и настройте профиль визуализации таким образом, чтобы узлы со свойством *friend* отображались красным цветом.
3. Добавьте свойство *twoHandFriend* всем друзьям друзей одной персоны и настройте профиль визуализации таким образом, чтобы узлы со свойством *twoHandFriend* отображались желтым цветом.
4. Добавьте свойство *manyFriends* персонам, имеющим больше 5 друзей, свойство *fewFriends* – персонам, у которых меньше 3 друзей. Настройте профиль визуализации таким образом, чтобы узлы со свойством *manyFriends* отображались зеленым цветом, узлы со свойством *fewFriends* отображались красным цветом, а все остальные узлы – желтым цветом.
5. Добавьте свойство *group1* персонам, состоящим в некоторой группе, *group2* – персонам, состоящим в другой группе, а свойство *bothGroups* -- персонам, состоящим в обеих группах. Настройте профиль визуализации таким образом, чтобы узлы со свойством *group1* отображались синим цветом, узлы со свойством *group2* отображались красным цветом, а узлы со свойством *bothGroups* – фиолетовым цветом.

2.7. ГеоСУБД PostGIS

При выполнении заданий будет использоваться учебная географическая база данных http://gis-lab.info/qa/geosample.html, адаптированная для актуальной версии PostGIS. База данных содержит сведения по Новосибирской и Кемеровской областям, республике Алтай и Алтайском крае.

В качестве основного источника справочной информации по СУБД PostGIS используйте приведенную ниже краткую справку и ресурсы [12, 13].

Краткая справка по СУБД PostGIS

PostGIS представляет собой расширение объектно-реляционной СУБД PostgreSQL, предназначенное для хранения географических данных. PostGIS включает поддержку пространственных индексов R-Tree/GiST и функции обработки геоданных.

Объекты ГИС, поддерживаемые PostGIS, являются надмножествами "Simple Features", определенных Консорциумом OpenGIS (OGC). При установке по умолчанию PostGIS устанавливается в папку СУБД PostgreSQL `share\contrib\postgis`.

Для работы с географической информацией в базу данных PostgreSQL необходимо поставить расширение PostGIS с помощью следующих команд:

```
psql -d yourdatabase -c "CREATE EXTENSION postgis;"
psql -d yourdatabase -c "CREATE EXTENSION postgis_topology;"
```

Некоторые сборки PostGIS во время установки создают в PostgreSQL шаблон географической базы данных `template_postgis`. В этом случае можно инициализировать новую БД следующей командой:

```
CREATE DATABASE my_spatial_db TEMPLATE=template_postgis
```

PostGIS определяет три специальных вида данных: *геометрия (geometry)*, *география (geography)* и *растр (raster)*. Геометрии представляют собой геометрические объекты следующих видов: *point*, *line*, *polygon*, *multipoint*, *multiline*, *multipolygon*, *geometrycollections*.

Начиная с версии PostGIS 2.0, появилась поддержка геометрий *TINS* и *Polyhedral Surfaces*. Они определены в формате *Well Known Text Open GIS* (с расширениями *XYZ*, *XYM*, *XYZM*). При работе с геометриями используются *планарные системы координат*. Географии предназначены для работы с объектами планетарного масштаба в геодезических системах координат. Растры используются для хранения и анализа растровой информации. Назначение и работа с географиями и геометриями оставлена на самостоятельное изучение.

Объекты ГИС, поддерживаемые PostGIS, являются надмножествами "Simple Features", определенных Консорциумом OpenGIS (OGC). Начиная с версии PostGIS 0.9, поддерживаются все объекты и функции, определенные OGC в спецификации "Simple Features SQL". PostGIS расширяет стандарт поддержкой координат 3DZ, 3DM и 4D.

Спецификация OpenGIS определяет два стандартных способа определения пространственных объектов: в форме *Well-Known Text (WKT)* и в форме *Well-Known Binary (WKB)*. WKT и WKB включают информацию о типе объекта и координаты, составляющие объект.

Примеры текстового представления (WKT) пространственных объектов:

```
POINT(0 0)
LINESTRING(0 0, 1 1, 1 2)
POLYGON((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2,1 1))
MULTIPOINT(0 0, 1 2)
MULTILINESTRING((0 0, 1 1, 1 2),(2 3, 3 2, 5 4))
```

```
MULTIPOLYGON(((0 0, 4 0, 4 4, 0 4, 0 0),
     (1 1, 2 1, 2 2, 1 2, 1 1)),
     ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
GEOMETRYCOLLECTION(POINT(2 3), LINESTRING((2 3, 3 4)))
```

Помимо этого, спецификация OpenGIS требует, чтобы внутренний формат хранения пространственных объектов включал идентификатор системы координат (spatial referencing system identifier - SRID). SRID необходим для добавления объекта в базу данных.

Ввод/вывод в этих форматах доступен с использованием следующих интерфейсов:

```
bytea WKB = asBinary(geometry);
text WKT = asText(geometry);
geometry = GeomFromWKB(bytea WKB, SRID);
geometry = GeometryFromText(text WKT, SRID);
```

Например, правильный запрос для создания и вставки пространственного объекта OGC может быть таким:

```
INSERT INTO geotable ( the_geom, the_name )
VALUES (GeomFromText('POINT(-126.4 45.32)', 312), 'A Place');
```

Существуют *два способа создания таблицы с колонкой пространственного типа данных*. В первом способе таблица создается с помощью команды CREATE TABLE, а в колонке с пространственными данными указывается тип вида geometry(<geometryType>, <srid>). geometryType определяет тип геометрий, которые будут храниться в этом столбце. Srid определяет систему координат для данных геометрий. Пример:

```
CREATE TABLE ROADS (
     ID int4,
     ROAD_NAME varchar(25),
     geom geometry(LINESTRING,4326) );
```

Дополнительный пространственные колонки могут быть добавлены с помощью ALTER TABLE. Пример:

```
ALTER TABLE roads ADD COLUMN geom2 geometry(LINESTRINGZ, 4326);
```

В предыдущих версия PostGIS добавление пространственных колонок происходило в два этапа с использованием специальной функции AddGeomtryColumn.

Спецификация функции и пример ее вызова:

```
AddGeometryColumn(<schema_name>, <table_name>, <column_name>,
     <srid>, <type>, <dimension> )
SELECT AddGeometryColumn('public', 'roads', 'geom', 423,
'LINESTRING', 2)
```

Пример пространственного запроса, который возвращает список всех геометрий, содержащих пространственно указанный полигон:

```
SELECT id, the_geom
FROM thetable
WHERE
ST_Contains(the_geom,'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))');
```

Указания к заданию 12. Разработка пространственных запросов

Изучите справочную информацию по функциям PostGIS и разработайте следующие запросы:

1) Выдать расстояние между двумя заданными населенными пунктами (например, между Новосибирском и Кемерово).
2) Выдать площадь заданного субъекта Российской Федерации.
3) Выдать площади всех особо охраняемых природных территорий в республике Алтай.
4) Выдать без повторений пары субъектов Российской Федерации, имеющие общие границы.
5) Выдать суммарную длину дорог на территории Новосибирской области.
6) Выдать общее количество населенных пунктов в каждом из субъектов Российской Федерации.
7) Выдать общее количество озер в радиусе 100 километров от Барнаула.

Указания к заданию 13. Визуализация геоданных с помощью MapServer

Разработайте и протестируйте запросы на получение данных для визуализации с использованием функций PostGIS. Примените разработанные запросы в MapServer и выполните следующие визуализации:

1) Отобразить карту субъектов Российской Федерации со всеми населенными пунктами.
2) Отобразить карту автомобильных и железных дорог.
3) Отобразить карту водоемов.
4) Отобразить при мелком масштабе (1:1000000 и более) только магистрали и города, а при крупном масштабе – полную карту дорог и все населенные пункты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кузнецов, С.Д. Три манифеста баз данных: ретроспектива и перспективы. / С.Д. Кузнецов. URL: http://citforum.ru/database/articles/manifests/ (дата обращения: 11.08.2013).
2. Кузнецов, С.Д. Транзакционные параллельные СУБД: новая волна. / С.Д. Кузнецов. URL: http://citforum.ru/database/articles/kuz_oltp_2010/ (дата обращения: 11.08.2013).
3. Клеменков, П.А. Большие данные: современные подходы к хранению и обработке / П.А. Клеменков, С.Д. Кузнецов // Труды ИСП РАН. 2012. Т. 23. С. 143-158.
4. NoSQL Databases. / URL: http://www.nosql-database.org (дата обращения: 11.08.2013)
5. Кузнецов, С.Д. Базы данных. Модели и языки. / С.Д. Кузнецов. – М.: БИНОМ, 2008. – 720 с.
6. Oracle Documentation. / URL: http://docs.oracle.com (дата обращения: 11.08.2013)
7. Кайт, Т. Oracle для профессионалов. / Т. Кайт. Пер. с англ. – СПб.: DiaSoft, 2004. – 1427 с.
8. Sedna XML Database. / URL: http://www.sedna.org (дата обращения: 11.08.2013).
9. The MongoDB manual. / URL: http://docs.mongodb.org/manual/ (дата обращения: 11.08.2013)
10. Бэнкер, К. MongoDB в действии. / К. Бэнкер. Пер. с англ. А. Слинкин. – М.: ДМК Пресс, 2012. – 394 с.
11. The Neo4j Manual. / URL: http://docs.neo4j.org/ (дата обращения: 11.08.2013).
12. PostGIS 2.0 Manual. / URL: http://postgis.net/documentation (дата обращения: 11.08.2013).
13. Скоробогатов, Д. Руководство по PostGIS. / Д. Скоробогатов, М. Дубинин. URL: http://gis-lab.info/docs/postgis/manual/ (дата обращения: 11.08.2013).

ОГЛАВЛЕНИЕ