



ПОСТ-РЕЛЯЦИОННЫЕ СИСТЕМЫ БАЗ ДАННЫХ

Я не согласен с тем, что вы говорите, но буду до последней капли крови защищать ваше право высказать вашу собственную точку зрения.
Ф.М. Вольтер

Технологии баз данных

Содержание

2

- Мотивация пост-реляционных систем баз данных
- Объектно-ориентированные системы баз данных
- Объектно-реляционные системы баз данных
- "Истинно" реляционные системы баз данных

Технологии баз данных © М.Л. Цымблер

Несоответствие импеданса

3

- Отсутствие неатомарных типов данных.
 - 1НФ запрещает "вложенные" таблицы.
- Отсутствие пользовательских типов данных.
 - Нельзя определить новый тип данных и его операции на базе имеющихся типов и операций.
- Отсутствие типов данных со сложной структурой
 - Структуры (записи), коллекции (множество, массив, список и др.), данные САПР и ГИС, документ-данные.
- Отсутствие поддержки концепций ООП
 - Объект, класс, инкапсуляция, наследование, полиморфизм.

Технологии баз данных © М.Л. Цымблер

Манифесты систем баз данных

4

- **1989, 1-й манифест**
 - *Аткинсон М. и др.* Манифест систем объектно-ориентированных баз данных // СУБД. 1995. № 4. С. 142-155.
 - Попытка дать научное определение системы ОО баз данных.
 - Новые системы необходимо строить на базе новой ООМД (имеющуюся РМД нужно отбросить).
- **1990, 2-й манифест**
 - *Стоунбрейкер М. и др.* Системы баз данных третьего поколения: Манифест // СУБД. 1996. № 2.
 - Инженерно-публицистический документ – ответ сообщества SQL-разработчиков авторам 1-го манифеста.
 - Новые системы необходимо строить на базе имеющейся SQL-модели.
- **1995, 3-й манифест**
 - *Дарвен Х., Дейт К.* Третий манифест // СУБД. -1996. -№ 1. -С. 110-123.
 - Строго научное определение новых систем баз данных.
 - Новые системы необходимо строить на базе истинно реляционной модели (имеющиеся реляционную и SQL модели нужно отбросить).

Технологии баз данных © М.Л. Цымбалер

ОО системы баз данных

5

- Консорциум ODMG (*Object Database Management Group*, впоследствии *Object Data Management Group*) образован в 1991 для выработки промышленного стандарта ОО баз данных.
- Компоненты стандарта ODMG
 - Язык определения данных (Object Definition Language, ODL).
 - Язык объектных запросов (Object Query Language, OQL).
 - Языки манипулирования объектами (Object Manipulation Languages, OML).

Технологии баз данных © М.Л. Цымбалер

ODL, Object Definition Language

6

- ODL позволяет описать схему данных в виде набора интерфейсов классов (атрибуты и методы классов, связи между классами).
- ODL не является языком программирования; реализация методов классов выполняется на одном из языков категории OML.

Технологии баз данных © М.Л. Цымбалер

Типы в ODL

7

- Атомарные
 - integer, float, character, string, boolean, enum
- Составные (Объектные и литеральные)
 - Struct и struct { тип имя, тип имя, ... } – структура
 - Коллекции
 - Set и set <тип> – множество
 - Bag и bag <тип> – мультимножество
 - List и list <тип> – список
 - Array и array <тип, количество> – массив
 - Dictionary и dictionary <типКлюча, типЗначения> – словарь
- Ограничения на типы связей
 - не разрешается использовать атомарный тип: Set <integer>
 - не разрешается использовать структуры: Struct { Supplier s, Part p }
 - не разрешается применять конструктор типа-коллекции более одного раза: Set <Array<Part, 100>>

Технологии баз данных © М.Л. Цымбалер

Описания в ODL: атрибуты

8

```
class Supplier {
  attribute string name;
  attribute struct addrType {
    string street, string city, string zip } address;
  attribute float rating; };
class Part {
  attribute string name;
  attribute enum colorType { red, white, green } color;
  attribute Supplier::addrType address;
  attribute float price; };
```

Технологии баз данных © М.Л. Цымбалер

Описания в ODL: связи

9

```
class Supplier {
  attribute string name;
  attribute struct addrType {
    string street, string city, string zip } address;
  attribute integer rating;
  relationship Set <Part> supplierOf; };
class Part {
  attribute string name;
  attribute enum colorType { red, white, green } color;
  attribute Supplier::addrType address;
  attribute float price;
  relationship Set <Supplier> suppliedBy; };
```

Технологии баз данных © М.Л. Цымбалер

Описания в ODL: обратные связи

10

```
class Supplier {
  attribute string name;
  attribute struct addrType {
    string street, string city, string zip } address;
  attribute integer rating;
  relationship Set <Part> supplierOf inverse Part::suppliedBy; };
class Part {
  attribute string name;
  attribute enum colorType { red, white, green } color;
  attribute Supplier::addrType address;
  attribute float price;
  relationship Set <Supplier> suppliedBy inverse
Supplier::supplierOf; };
```

Связи в ODL

11

E/R связь	ODL связь	
	C1	C2
1:1	relationship <C2> relToC2	relationship <C1> relToC1
1:M	relationship <i>collectType</i> <C2> relToC2	relationship <C1> relToC1
M:1	relationship <C2> relToC2	relationship <i>collectType</i> <C1> relToC1
M:N	relationship <i>collectType</i> <C2> relToC2	relationship <i>collectType</i> <C1> relToC1

collectType – Set, List, Bag, Array

Описания в ODL: многосторонние связи

12

```
class Supplement {
  attribute integer Qty;
  relationship Supplier theSupplier inverse Supplier::supplierFor;
  relationship Part thePart inverse Part::partFor; };
class Supplier {
  ...
  relationship Set <Supplement> supplierFor
  inverse Supplement::theSupplier ; };
class Part {
  ...
  relationship Set <Supplement> partFor
  inverse Supplement::thePart ; };
```

Описания в ODL: операции

13

```
class Supplier {
  attribute string name;
  attribute struct addrType {
    string street, string city, string zip } address;
  attribute integer rating;
  relationship Set <Part> supplierOf
    inverse Part::suppliedBy;
  integer totalPartsQty() raises (noSuppliedParts);
  integer suppliedPartsQty(in Set <string>)
    raises (noSuppliedParts);
  void suppliedPartNames(out Set <string>)
    raises (noSuppliedParts); };
```

Описания в ODL: наследование

14

- Обычное наследование
 - ▣ class Ancestor { атрибуты, связи, методы };
 - ▣ class Descendant extends Ancestor { новые атрибуты, связи, методы };
- Множественное наследование
 - ▣ class Ancestor1 { ... };
 - ▣ class Ancestor2 { ... };
 - ▣ class Descendant extends Ancestor1 : Ancestor2 { новые атрибуты, связи, методы };
 - ▣ Возможны конфликты имен атрибутов, связей, методов, наследуемых от разных классов.

OQL, Object Query Language

15

- Язык запросов, близкий к SQL (не язык программирования).
- Результат OQL-запроса имеет тип из перечня типов ODMG.
- Не имеет явных операций обновления (используются вызовы соответствующих методов классов).
- Операторы OQL могут вызываться из любого языка программирования. В OQL-запросах могут присутствовать вызовы методов, реализованных на других языках программирования.

OQL: примеры запросов

16

- Найти даты рождения служащих, зарплата которых превышает 20000 руб.
 - ▣ `select distinct E.birthDate`
`from employees E`
`where E.salary > 20000`
 - ▣ Результатом запроса будет литеральное значение типа **set** <date>, т.е. литеральное значение-множество, элементами которого являются значения типа date.

OQL: примеры запросов

17

- Найти имена и даты рождения служащих, зарплата которых превышает 20000 руб.
 - ▣ `select distinct struct (name: E.name, birth: E.birthDate)`
`from employees E`
`where E.salary > 20000`
 - ▣ Результатом запроса является литеральное значение-множество, элементами которого являются значения типа **struct** { string <20> name; date birth }.

OQL: примеры запросов

18

- Найти имена и даты рождения служащих, зарплата которых превышает 20000 руб.
 - ▣ в классе EMP определена операция age (возраст соответствующего служащего)
 - ▣ `select distinct struct (name: E.name, age: E.age)`
`from employees E`
`where E.salary > 20000.00`
 - ▣ Результатом запроса является литеральное значение-множество, элементами которого являются значения типа **struct** { string <20> name; interval age }

OQL: примеры запросов

19

- Получить номера начальников отделов и тех сотрудников их отделов, зарплата которых превышает 20000 руб.
 - `select distinct struct (mgr: D.mgr, dhs: (select E from d.consistsOf as E where E.salary > 20000)) from departments D`
- Запрос похож на SQL-запрос с вложенным подзапросом, но это только внешнее сходство!
 - В разделе FROM "подзапроса" имеется переход по связи consistsOf от экземпляра объектного типа DEPT ко множеству экземпляров объектного типа EMP.
 - Результат запроса имеет тип `set < struct { integer mgr ; bag < EMP > DHS } >`.

OQL: примеры запросов

20

- Хранимые в базе данных объекты могут иметь индивидуальные имена.
 - Пусть в базе данных сохраняется объект типа DEPT с именем mainDepartment
 - `mainDepartment`
 - Результатом запроса будет соответствующий объект
 - `mainDepartment.consistsOf`
 - Результатом запроса будет литеральное значение-множество, состоящее из объектов типа EMP, ассоциированных через связь consistsOf с объектом mainDepartment .
- Имя экстенета можно трактовать как имя объекта-множества.
 - `EMPLOYEES`
 - Результатом будет литеральное множество, состоящее из всех объектов, которые содержатся в указанном экстенете.

OQL: примеры запросов

21

- Пусть имеются определения объектных типов:
 - `typedef Set <interval> ages;`
 - `class persInfo { attribute string <20> n; attribute date b; };`
 - `typedef Bag <persInfo > info;`
- `ages (select distinct E.age from employees E where E.salary > 20000)`
 - Результатом запроса является объект-множество, включающий литеральные значения типа interval.
- `info (select persinfo (n: E.name, b: E.birthDate) from employees E where E.salary > 20000)`
 - По литеральным значениям имени и даты рождения каждого служащего, размер зарплаты которого превышает 20000 руб., конструируется объект типа persInfo, а на основе литерального мультимножества этих объектов конструируется объект-мультимножество типа info .

ОQL: примеры запросов

23

- Пусть имеются определения объектных классов
 - class DEPT {
...
relationship EMP managedBy
inverse EMP :: managerOf
}
 - class EMP {
...
relationship DEPT managerOf
inverse DEPT :: managedBy
}
- mainDepartment.managedBy
 - Результатом запроса будет объект типа EMP.
- mainDepartment.managedBy.name
 - Результатом запроса будет значение типа string.
- Получить имена всех сотрудников отдела
 - mainDepartment.consistsOf.name – неверно (для связи "один-ко-многим" такие путевые выражения в OQL не допускаются)
 - select E.name from mainDepartment.consistsOf as E

Технологии баз данных © М.Л. Цымбалер

ОQL: примеры запросов

23

- Получить имена служащих и номера их отделов для служащих, работающих в отделах с фондом заработной платы, размер которого превышает 1000000 руб.
 - select **struct** (name: E.name, dept: D.deptNo)
from employees E, E.worksAt D
where E.worksAt <>NIL andthen
D.deptTotalSalary > 1000000

Технологии баз данных © М.Л. Цымбалер

ОQL: примеры запросов

24

- Выбрать всех служащих, у которых имеются однофамильцы.
 - select distinct E
from employees E, employees E1
where E <> E1 and E.name = E1.name

Технологии баз данных © М.Л. Цымбалер

OQL: примеры запросов

25

- Выбрать всех служащих, работающих в отделе.
 - `select distinct E`
`from employees E`
`where is_defined(E.worksAt.deptNo)`
 - `select distinct E`
`from employees E`
`where E.worksAt <> NIL`

OQL: примеры запросов

26

- Найти название отдела, в котором работают служащие, средняя зарплата которых меньше средней зарплаты служащих любого другого отдела.
 1. Построить множество объектов-служащих, для которых известен номер отдела
 - `define employeesD () as`
`select e from employees E`
`where E.works is not NIL`
 2. Сгруппировать служащих с известными номерами отделов по номерам отделов и вычислить размер средней зарплаты для каждого отдела
 - `define deptAvgSal () as`
`select deptNo, avgSal : avg (select X.E.salary from partition X)`
`from employeesD() E`
`group by deptNo : E.deptNo`
 3. Отсортировать полученное мультимножество по значениям атрибута avg Sal
 - `define sortedDeptAvgSal() as`
`select S from deptAvgSal () as S order by S.avgSal`
 4. Выбрать значение атрибута deptNo из элемента списка sortedDeptAvgSal с наименьшим значением avgSal (этот элемент стоит в списке первым)
 - `first (sortedDeptAvgSal ().deptNo`

OML, Object Manipulation Language

27

- OML – это ОО язык для реализации методов классов и приложений баз данных.
- В одной ООСУБД могут поддерживаться несколько OML.
 - В стандарте ODMG-2 специфицированы правила связывания для языков C ++, Smalltalk и Java.

Критика стандарта ODMG

28

- Эkleктичен: в стандарт стремились поместить свойства всех коммерческих ООСУБД, существовавших к моменту написания документа.
- Разделение языков ODL, OCL и OML затрудняет целостное понимание модели.
- OQL сложен для использования непрограммистами.

Технологии баз данных © М.Л. Цымбалер

ОР системы баз данных

29

- Основаны на расширении SQL (стандарт SQL:1999) ОО свойствами:
 - ▣ *определяемые пользователями типы данных*, атрибуты и методы.
 - ▣ *типизированные таблицы*, строки которых являются экземплярами (или значениями) пользовательских типов.

Технологии баз данных © М.Л. Цымбалер

Пример: UDT

30

- **create type** EmpNo **as integer final**;
- **create type** DeptNo **as integer final**;
- **create type** ProjNo **as integer final**;
- **create table** EMP (
 - empID EmpNo,
 - empName varchar(20),
 - deptID DeptNo,
 - projID projNo);
- **select** empName **from** EMP
 - where** empID > deptID; -- Ошибка
- **select** empName **from** EMP
 - where** **cast**(empID **to integer**) > **cast**(deptID **to integer**);

Технологии баз данных © М.Л. Цымбалер

Пример: типизированные таблицы

31

```
❑ create type emp_t as (  
    empName varchar(20),  
    empBdate date,  
    empSal salary,  
    dept ref (dept));  
instantiable -- могут быть созданы экземпляры  
not final -- могут быть созданы подтипы  
ref is system generated  
instance method age () returns decimal (3,1);
```

Пример: типизированные таблицы

32

```
❑ create type programmer_t under emp_t as (  
    progLang varchar (10))  
instantiable  
not final;  
  
❑ create type dept_t as (  
    deptNo integer,  
    deptName varchar(200),  
    deptMgr ref (emp_t))  
instantiable  
not final;
```

Пример: типизированные таблицы

33

```
❑ create table EMP of emp_t (  
    ref is deptID system generated,  
    dept with options scope DEPT)  
  
❑ create table PROGRAMMER of programmer_t under  
    EMP;  
  
❑ create table DEPT of dept_t (  
    ref is empID system generated,  
    deptMgr with options scope EMP);
```

Пример: типизированные таблицы

34

- Найти имена всех служащих, размер заработной платы которых меньше 20000
 - ▣ `select empName
from EMP
where empSal < 20000;`
- Найти имена всех служащих, не являющихся программистами, размер заработной платы которых меньше 20000
 - ▣ `select empName
from only (EMP)
where empSal < 20000;`

Технологии баз данных © М.Л. Цымбалер

Пример: типизированные таблицы

35

- Найти имена и названия отделов всех служащих, размер заработной платы которых меньше 20000
 - ▣ `select empName, DEPT->deptName
from EMP
where empSal < 20000;`
- Найти имена служащих и имена руководителей их отделов для служащих, получающих зарплату, меньшую 20000
 - ▣ `select empName, DEPT->deptMgr->empName
from EMP
where empSal < 20000;`

Технологии баз данных © М.Л. Цымбалер

Пример: типизированные таблицы

36

- Найти имя и возраст руководителя отдела 605
 - ▣ `select deptMgr->empName, deptMgr->age()
from DEPT
where deptNo = 605;`
- Получить полные данные о руководителе отдела 605
 - ▣ `select deref(deptMgr)
from DEPT
where deptNo = 605;`

Технологии баз данных © М.Л. Цымбалер

ИР системы баз данных

37

- Третий манифест (в отличие от первых двух манифестов)
 - ▣ не содержит критики предыдущих манифестов
 - ▣ не опирается на существующие реализации СУБД
 - ▣ обновляется авторами (как технический отчет).
- Отвергает ОО и SQL подходы первых двух манифестов.
- Представляет собой набор конструктивных предложений, следование которым может (по мнению авторов) привести к созданию СУБД следующего поколения, удовлетворяющим современным потребностям и базирующимся на чистой реляционной модели.



Крис Дейт
(р. 1941)



Хью Дарвен
(р. 19??)

Технологии баз данных © М.Л. Цымбалер

Третий манифест систем баз данных

38

- Поддержка ОО возможностей желательна, но эти возможности ортогональны реляционной модели.
 - ▣ Реляционная модель не нуждается в расширении или коррекции, чтобы можно было связать эти возможности с некоторым языком баз данных, способным представлять искомые основы.
 - ▣ Пусть такой язык существует и называется **D** (Date and Darwen's Database Dream :-).
- Язык **D** является предметом *предписаний, запретов и очень строгих суждений* двух типов:
 - ▣ RM (Relational Model) – следующих из сущности РМД, которые абсолютны и не могут быть предметом компромисса
 - ▣ ОО (Other Orthogonal :-) – не относящихся непосредственно к РМД.
- В настоящее время отсутствуют СУБД на основе идей Третьего манифеста, и ситуация не имеет тенденции к изменению.

Пост-реляционные системы баз данных © М.Л. Цымбалер

Заключение

39

- Проблема "несоответствия импеданса" – основная причина появления пост-реляционных систем.
- ОО подход: РМД нужно отбросить. Стандарт ODMG: ODL+OQL+OML.
- ОР подход: дополнение SQL ОО возможностями.
- ИР подход: SQL и РМД нужно отбросить как не реляционные.

Технологии баз данных © М.Л. Цымбалер