ВТОРОЙ МАНИФЕСТ И ОБЪЕКТНО-РЕЛЯЦИОННЫЕ СИСТЕМЫ БАЗ ДАННЫХ

Все можно наладить, если вертеть это в руках достаточно долго.

А. Блох

Содержание

- Основные положения Второго манифеста
- Объектные расширения в стандарте SQL:1999
- Объектно-реляционные черты СУБД Oracle

Второй манифест

- Сетевые и иерархические СУБД (1970-е) – *системы баз данных первого поколения* (CODASYL, IMS).
- Реляционные СУБД (1980-е) – *системы баз данных второго поколения* (DB2, INGRES, NON-STOP SQL, Oracle, Rdb/VMS). В силу проблемы несоответствия импеданса нуждаются в пересмотре.
- Второй манифест – попытка определить *системы баз данных третьего поколения*.

Принципы СУБД 3 поколения

- Обеспечение поддержки более богатых структур объектов и правил.
  - Средства манипулирования нетрадиционными элементами данных (тексты, пространственные данные, видеоданные и др.).
  - Возможность задавать группу правил, касающихся элементов данных, записей и наборов (впоследствии триггеры).
- Включение СУБД второго поколения.
  - непроцедурный доступ и независимость данных
- Открытость для других подсистем.
  - допускать реализацию доступа из дополнительных инструментов, функционирующих в различных средах
  - каждая система третьего поколения должна легко объединяться с другими СУБД для создания распределенных систем баз данных.

Предложения 2 манифеста

- Предложения по управлению объектами и правилами
  - уточняют требования к управлению объектами и правилами.
- Предложения по увеличению функциональных возможностей СУБД
  - набор предложений, являющихся следствием того, что системы третьего поколения должны включать в себя системы второго поколения.
- Предложения, исходящие из требования открытости систем третьего поколения.

Система типов

- Являются желательными механизмы :
  - система абстрактных типов данных для создания новых базовых типов;
  - конструкторы типов "массив", "последовательность", "запись", "множество", "объединение";
  - функции как тип (методы);
  - рекурсивная композиция всех перечисленных выше конструкторов.

Инкапсуляция, методы, наследование

- Необходимо поддерживать инкапсуляцию и методы в базах данных.
- Должно поддерживаться как единичное, так и множественное наследование.

УИД записей

- Уникальные идентификаторы записей должны задаваться СУБД только в том случае, когда недоступен определенный пользователем первичный ключ.

Правила в базах данных

- Правила (триггеры, ограничения) станут одной из ключевых характеристик будущих систем.

Увеличение функциональных возможностей СУБД

- Все виды программируемого доступа к базам данных должны осуществляться через непроцедурный язык доступа высокого уровня.
- Должно быть по крайней мере два способа спецификации наборов (множества, массивы, последовательности и др.): посредством перечисления членов и путем использования языка запросов для задания членов.
- Существенно наличие обновляемых представлений.

Открытость СУБД 3 поколения

- СУБД третьего поколения должны быть доступны из различных языков программирования высокого уровня.
- SQL – интергалактический язык баз данных.
- Запросы и ответы на них должны образовывать нижний уровень коммуникаций между клиентом и сервером.

Первый и второй манифесты

- Общие темы: выгодность использования богатой системы типов, функций, наследования и инкапсуляции.
- Отличия:
  - Второй манифест обращается к более широкому кругу вопросов (не только управление объектами): поддержка управления данными, правилами и объектами на основе полного набора инструментов при наличии интеграции СУБД и языка запросов в многоязычную среду.
  - Доступ к СУБД должен осуществляться при помощи языка запросов (без физической навигации).
  - Использование автоматических наборов данных.
  - Добавление свойства стабильности (персистентности) в языки программирования.
  - УИД должны задаваться либо пользователем, либо системой.
  - Современные реляционные системы могут эволюционировать к СУБД 3 поколения.

Объектная модель SQL

- *Определяемые пользователями типы данных*, атрибуты и методы.
- *Типизированные таблицы*, строки которых являются *экземплярами* (или *значениями*) пользовательских типов.

Пример: UDT

- 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);

Пример: типизированные таблицы

- 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

- 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

- 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

- 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;

Пример: типизированные таблицы

- 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;

Пример: типизированные таблицы

- 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;

Пример: определение объектного типа


```
create type TAddress as object (
    Country char(3),
    City char(20),
    ZipCode char(8), …);
create type TPerson as object (
    Name char(50),
    Birth date,
    Address TAddress,
    member function Age return number);
create type body TPerson is
member function Age return number is
begin
    return round(months_between(sysdate, Birth)/12);
end;
end;
```

Методы-конструкторы

- *Конструктор* – предопределенный метод объектного типа для создания экземпляров данного типа.

```
declare
    JohnDoe TPerson;
    JohnAddress TAddress;
    JohnAge number;
begin
    JohnDoe := TPerson('John Doe', 3-Jul-24, TAddress('USA', 'NY', '12345'));
     JohnAddress := JohnDoe.Address;
     JohnAge := JohnDoe.Age();
end;
```

Методы сравнения экземпляров

- Пользователь может определить один из (но не оба одновременно) следующих *методов сравнения* экземпляров объектного типа :
  - *map-метод* – сопоставляет одному экземпляру число;
  - *order-метод* – выдает результат сравнения двух экземпляров (0, >0 или <0).

```
create type TPerson as object (
   … ,
   map member function MapPerson
            return number);
create type body TPerson is
map member function MapPerson
            return number is
begin
   return Age();
end;
end;
```

```
create type TPerson as object (
   … ,
   order member function MapPerson
            return number);
create type body TPerson is
order member function OrderPerson
      (x in TPerson) return number is
begin
   return Age() – x.Age();
end;
end;
```

Объектные типы в таблицах

- Поле реляционной таблицы может иметь объектный тип (поле – *объект-столбец*).

```
declare
   JohnDoe TPerson;
create table emp (
   empno number primary key,
   person TPerson);
JohnDoe := TPerson('John Doe', 3-Jul-24, TAddress('USA', 'NY', '12345'));
insert into emp values (emp_seq.nextval, JohnDoe);
insert into emp values (emp_seq.nextval,
   TPerson('James Bond', 3-Jul-24, TAddress('ENG, 'London', '54321'));
select empno, person.Name, person.Age() from emp
order by person.Age() desc;
```

Объектные таблицы

- Запись реляционной таблицы может иметь объектный тип. В этом случае ее можно рассматривать
  - как реляционную таблицу из нескольких столбцов (их столько, сколько полей в объектном типе)
  - как таблицу из одного столбца, содержащего *объекты-строки*

```
create table PersonTab of TPerson;

insert into PersonTab values (
   TPerson('James Bond', 3-Jul-24, TAddress('ENG, 'London', '54321'));
insert into PersonTab values ('John Doe', 3-Jul-24, 'USA', 'NY', '12345');

select * from PersonTab p order by p.Age() desc;
select value(p) from PersonTab p order by value(p) desc;
```

Идентификаторы объектов

- Объект-строка имеет скрытый назначаемый системой атрибут *ОИД – уникальный идентификатор* экземпляра.
- ОИД представляет собой 16-байтовое целое число. ОИД используется при построении ссылок на экземпляры объектных типов.
- Пользователь может специфицировать ОИД объекта-строки как первичный ключ

```
create table PersonTab of TPerson (Name primary key)
        object id primary key;
```

Ссылки на объектный тип

- Встроенный тип REF используется для организации ссылок на экземпляры объектов-строк.

```
declare
   PJohnDoe ref TPerson;

select ref(p) into PJohnDoe from PersonTab p
where p.Name='John Doe';
update PersonTab p set p.Address.City='LA'
where ref(p)=PJohnDoe;
```

Раскрытие ссылок


```
create type TProject as object (
    ProjNo number,
    Name char(50),
    Budget number,
    MgrRef ref TPerson);

create table ProjectTab of TProject;

insert into ProjectTab values (1, 'Mission impossible', 0, PJohnDoe);

select ProjName, Budget, deref(MgrRef) from ProjectTab
    where deref(MgrRef).Age() < 40;

update ProjectTab set MgrRef=PJohnDoe where Budget > 100000;
```

Объектные типы-таблицы


```
create type TProjectTab as table of TProject;
create table Dept (
    DeptNo number primary key,
    Name char(50),
    Projects TProjectTab) nested table Projects store as ProjectTab;
-- Удаление вложенной таблицы
insert into Dept values (1, 'Sales', NULL);
-- Создание вложенной таблицы
update Dept set Projects=TProjectTab() where DeptNo=1;
-- Вставка записей во вложенную таблицу
insert into table (select Projects from Dept where DeptNo=1)
    values (1, 'Mission Impossible', 0, PJohnDoe);
-- Обновление записей вложенной таблицы
update Dept set Projects= TProjectTab(TProject(1, 'Mission Is Possible', 1000000,
PJamesBond)) where DeptNo=1;
```

Объектные типы-массивы


```
create type TBox as varray(3) of number;
create table Goods (
    id number primary key,
    name char(20),
    Sizes TBox not null);

insert into Goods values (1, 'Juice', TBox(10, 5, 20));

select id, name, Sizes(1) as Length, Sizes(2) as Width, Sizes(3) as
Height,
Sizes(1)*Sizes(2)*Sizes(3) as Volume
        from Goods;
```

Висячие ссылки


```
create type TProject as object (
    ProjNo number,
    Name char(50),
    Budget number,
    MgrRef ref TPerson);

create table ProjectTab of TProject;

select ProjName, Budget, deref(MgrRef) from ProjectTab
    where MgrRef is not dangling;

update ProjectTab set MgrRef=PJohnDoe where MgrRef is dangling;
```

Циклические ссылки


```
create type TDept; -- Незавершенное определение типа

create type TEmp as object (
        name char(30),
        dept ref TDept,
        mgr ref TEmp);

create type TEmpList as table of TEmp;

create type TDept as object (
        name char(30),
        mgr ref TEmp,
        staff TEmpList);
```

Ограничение области действия ссылок

- Область действия ссылки на объект-столбец может быть ограничена объектами-строками указанных таблиц (при этом типы объекта-столбца и объекта-строки должны совпадать).

```
create type TProject as object (
    ProjNo number,
    Name char(50),
    Budget number,
    MgrRef ref TPerson scope is PersonTab);

alter table ProjectTab
    add (scope for MgrRef is EmpTab);
```

NULL-объекты


```
create type TContact as object (
        name varchar2(30),
        phone varchar2(20) );
create table daycontacts (
        contact TContact,
        day date );
-- Выделяется место для размещения экземпляра
insert into daycontacts values (
        TContact(NULL, NULL), '03.07.2024' );
-- Место для размещения экземпляра НЕ выделяется
insert into daycontacts values (
        NULL, '03.07.2024' );
```

Указание значений по умолчанию


```
create TPeople as table of TPerson;

create table Department (
        deptno char(5) primary key,
        name char(20),
        mgr TPerson default
                TPerson(0,'John Doe',null),
        emps TPeople default
                TPeople(
                        TPerson(1,'John Smith',null),
                        TPerson(7, 'James Bond', null)) )
nested table emps store as empstab;
```

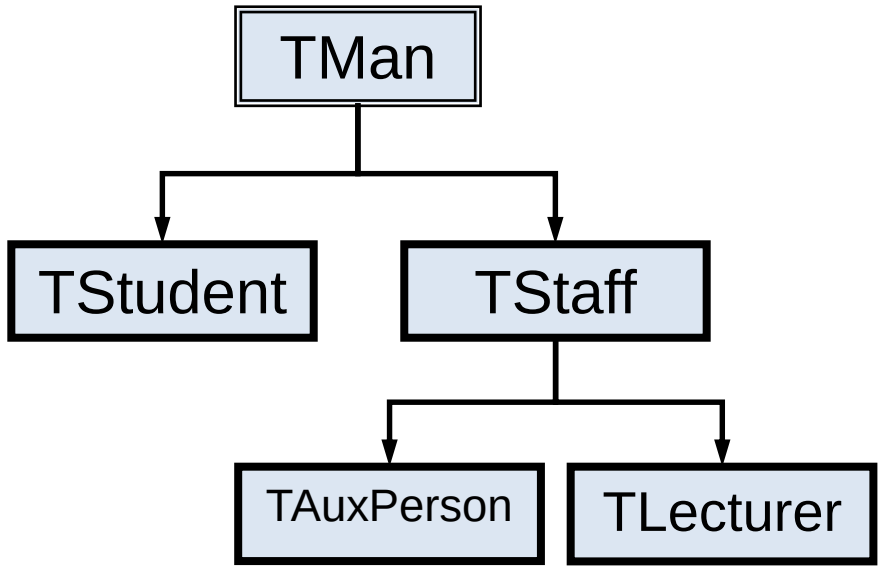
Ограничения целостности

- Ограничения целостности могут быть заданы только при непосредственном создании таблицы (не при создании пользовательского типа).

```
create table emps (
        emp TPerson,
        mgr ref TPerson,
        constraint pk_name emp.name primary key,
        constraint fk_mgr foreign key (mgr) references emps (emp),
        constraint unq_arph unique
                (emp.phonenum.area, contact.phonenum.phone),
        constraint nn_address check (emp.address is not null) );
```

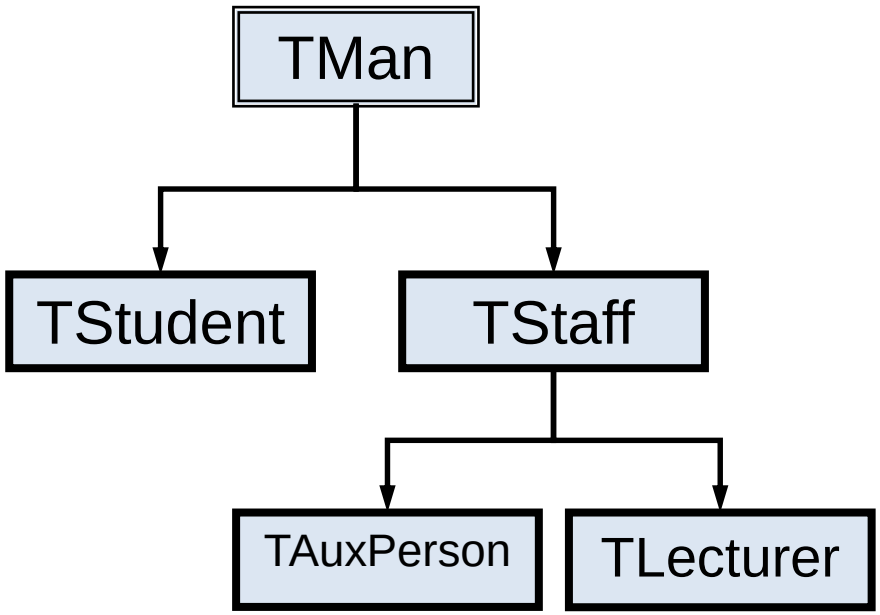
Подтипы и супертипы (наследование)


```
create type TMan as object (
            name char(50),
            gender char(1) ) not final;

create type TStudent under TMan
   as object ( group char(20) ) not final;

create type TStaff under TMan
   as object ( salary number,
               office char(3) ) not final;

create type TLecturer under TStaff
   as object ( degree char(10),
               dept char(50) ) not final;

create type TAuxPerson under TStaff
   as object ( job char(20) ) not final;
```



Подтипы и супертипы (наследование)


```
select supertype_name, subtype_name
from user_types
order by supertype_name, subtype_name ;

SUPERTYPE_NAME SUBTYPE_NAME
-----------------------------------------------
NULL           TMAN
TMAN           TSTAFF
TMAN           TSTUDENT
TSTAFF         TAUXPERSON
TSTAFF         TLECTURER
```



Подтипы и супертипы (наследование)


```
create type TCheck as object (
        when date;
        person TMan );
create table chkpoint of TCheck;

insert into chkpoint values (          sysdate,
        TMan('John Doe', 'M') );
insert into chkpoint values (          sysdate,
        TStudent('Иванов И.И.', 'М', 'МП-401') );
insert into chkpoint values (          sysdate,
        TStaff('Сидорова М.М.', 'Ж', 1500, '8')  );
insert into chkpoint values (          sysdate,
        TLecturer('Петрова А.П.', 'Ж', 3500, '8', 'ктн', 'каф. АСУ') );
insert into chkpoint values ( sysdate,
        TAuxPerson ('James Bond', 'M', 7000, '007', 'Cleaner') );
```

Подтипы и супертипы (наследование)


```
select * from chkpoint;

NAME                 GENDER
-------------------------------------
James Bond           M
John Doe             M
Иванов И.И.          M
Петрова А.П.         Ж
Сидорова М.М.        Ж
```

```
select * from chkpoint
where person is of (only TStaff);

NAME     GENDER SALARY OFFICE
--------------------------------
Сидорова М.М. Ж   1500   8
```

```
select * from chkpoint
where person is of (TStudent);

NAME        GENDER GROUP
-----------------------------------------
Иванов И.И. M          МП-401
```

```
select * from chkpoint
where person is of (TStaff);

NAME                 GENDER SALARY OFFICE
---------------------------------
James Bond           M          7000      007
Петрова А.П.         Ж          3500      8
Сидорова М.М.        Ж          1500      8
```

Объектные представления

- *Объектное представление (object view)* – виртуальная объектная таблица на базе реляционной либо объектной таблицы. Используется для конвертирования данных из реляционных таблиц в объектные и освоения объектно-реляционных техник работы с базой данных.
- Объектные представления не-обновляемы; их обновление реализуется с помощью триггеров INSTEAD OF.

```
create table emp (
    empno number primary key,
    name char(50).
    birth date,
    salary number);
create type TEmp (
    EmpNo number primary key,
    Name char(50).
    Birth date,
    Salary number);
```

```
create view HighlyPaidEmp of TEmp
with object OID (EmpNo) as
    select * from emp
    where Salary>1500;
```

Минусы OP-возможностей

- Отсутствие полной поддержки ОО-концепций (нет виртуальных методов).
- Сложный синтаксис OP-запросов.
- Сложность внесения изменений в объектную схему данных.

Пример. Для добавления атрибута в объектный тип-строку объектной таблицы нужно

(1) создать временную таблицу,
(2) сохранить данные из объектной таблицы во временную таблицу,
(4) удалить объектную таблицу,
(5) переопределить объектный тип,
(6) создать объектную таблицу заново,
(7) вставить в объектную таблицу данные из временной таблицы,
(8) удалить временную таблицу.

Заключение

- Рассмотрены основные положения Второго манифеста
- Рассмотрены объектные расширения в стандарте SQL:1999
- Рассмотрены объектно-реляционные черты СУБД Oracle