Introduction to Object-Oriented Databases

[面向对象数据库]

▲Object-Relational Data Models [对象关系数据模型]

Extend the relational data model by including object orientation (constructs to deal with added data types).
[通过面向对象的技术来扩展关系型数据库模型]
Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
[允许元组的属性具有复杂类型，包括非原子值（嵌套关系）]
Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
[保留基础的关系型模型情况下扩展模型（尤其是申明访问数据的模型）]
Upward compatibility with existing relational languages.
[与现有的关系型数据库语言兼容]

▲Motivation:

Permit non-atomic domains (atomic indivisible) such as set of integers or set of tuples
[允许非原子级的领域单元比如整数集&元组集]
Allows more intuitive modeling for applications with complex data
[允许更复杂数据的程序进行直观的建模]

▲Intuitive definition:

Allow relations whenever we allow atomic (scalar) values - relations within relations
[当认同原子级数据时我们就可以认同该关系，关系的嵌套]
Retains mathematical foundation of relational model
[保留关系型模型中的数学基础]
Violates first normal form (1NF)
[此做法违反 1NF]

▲Example of a Nested Relation [嵌套关系]

Example: library information system

title	*author_array*	*publisher*	*keyword_set*
		(name, branch)	
Compilers	[Smith, Jones]	(McGraw-Hill, NewYork)	{parsing, analysis}
Networks	[Jones, Frick]	(Oxford, London)	{Internet, Web}

Each book has title, a list (array) of authors (order matters), publisher with subfields name and branch, and a set of keywords (set)

4NF Decomposition of Nested Relation

Suppose for simplicity that title uniquely identifies a book
Decompose books into 4NF using the schemas:

Table1 (title, author, position)
Table2 (title, keyword)
Table3 (title, pub-name, pub-branch)

4NF design requires users to include joins in their queries.
[4NF 要求用户在查询中包含 join]

title	*author*	*position*
Compilers	Smith	1
Compilers	Jones	2
Networks	Jones	1
Networks	Frick	2

authors

title	*keyword*
Compilers	parsing
Compilers	analysis
Networks	Internet
Networks	Web

keywords

title	*pub_name*	*pub_branch*
Compilers	McGraw-Hill	New York
Networks	Oxford	London

在 SQL 里，主外键就是继承，外键继承的主键属性未定义(unnamed row types)，对于特殊的方法可以添加数据类型。

▲构造函数:

- Constructor functions are used to create values of structured types

```
create function Name(firstname varchar(20), lastname varchar(20))
returns Name
begin
   set self.firstname = firstname;
    set self.lastname = lastname;
end
```

- To create a value of type *Name*, we use
  new *Name*('John', 'Smith')
- Normally used in insert statements

```
insert into Person values
   (new Name('John', 'Smith),
    new Address('20 Main St', 'New York', '11001'),
    date '1960-8-22');
```

▲继承类：

- Suppose that we have the following type definition for Person:

```
create type Person
    (name varchar(20),
    address varchar(20))
```

- Using inheritance to define the student and teacher types

```
create type Student
 under Person
 (degree      varchar(20),
  department  varchar(20))
create type Teacher
 under Person
 (salary      integer,
  department  varchar(20))
```

- Subtypes can redefine methods by using **overriding method** in place of method in the method declaration

▲表的继承

- Tables created from subtypes can further be specified as **subtables**
- E.g. **create table** *people* **of** *Person;*
  create table *students* **of** *Student* **under** *people;*
  create table *teachers* **of** *Teacher* **under** *people;*

目前 SQL 不支持多重继承，主表更新时附表也会跟着改变数据。

▲**Example of array and multiset declaration [数组和多集声明示例:]:**

```
create type Publisher as
  (name          varchar(20),
   branch        varchar(20));
 create type Book as
  (title         varchar(20),
   author_array  varchar(20) array [10],
   pub_date      date,
   publisher     Publisher,
   keyword-set   varchar(20) multiset);
create table books of Book;
```

▲**Creation of Collection Values [允许值集编辑器的创建]**

Array construction [构建数组]

```
array ['Silberschatz',` Korth',` Sudarshan']
```

Multisets [多重集合]

```
multiset ['computer', 'database', 'SQL']
```

To create a tuple of the type defined by the books relation

```
('Compilers', array[` Smith',` Jones'],
new Publisher (` McGraw-Hill',` New York'),
multiset [` parsing',` analysis' ])
```

To insert the preceding tuple into the relation books

```
insert into books
values
   ('Compilers', array[` Smith',` Jones'],
         new Publisher (` McGraw-Hill',` New York'),
         multiset [` parsing',` analysis' ]);
```

▲**Approaches to make persistent objects [创建持久对象的方法]:**

Persistence by class - explicit declaration of persistence

Persistence by creation - special syntax to create persistent objects

Persistence by marking - make objects persistent after creation

Persistence by reachability - object is persistent if it is declared explicitly to be so or is reachable from a persistent object

▲**Degrees of permanence of object identity**

Intraprocedure: only during execution of a single procedure

Intraprogram: only during execution of a single program or query

Interprogram: across program executions, but not if data storage format on disk changes

Persistent: interprogram, plus persistent across data reorganizations

▲**Object-Relational Mapping (ORM) [对象关系映射] systems** built on top of traditional relational databases, the Hibernate ORM system is widely used

ORM 相当于中继数据，可以运用于任何语言的数据库。对象可以在数据库中检索，但对象只是暂时的，需要提供对象到关系的映射。限制是开销，尤其是批量修改时。

▲Comparison of Databases

Relational systems: simple data types, powerful query languages, high protection.

Persistent-programming-language-based OODBs: complex data types, integration with programming language, high performance.

Object-relational systems: complex data types, powerful query languages, high protection.

Object-relational mapping systems: complex data types integrated with programming language, but built as a layer on top of a relational database system