Фельдман Яков Адольфович
Jacob@pbm.onego.ru

Интуитивные объекты

В программировании различают *физические* и *логические* структуры данных. Но пользователи мыслят структурами более высокого уровня, назовем их *интуитивными* (термин мой). Известны интуитивные структуры: объекты OLAP и объекты TreeLogy. Предлагаю свой вариант: *интуитивные объекты*.

Параметр или ***типовой элемент данных*** это *пара* (имя, формат). ***Элемент данных*** это *пара* (параметр, значение) У интуитивного объекта всегда есть *набор* элементов данных. Набор (упорядоченный список) типовых элементов это *тип*. Для таких типов есть *наследование*. Например тип *сотрудник* ***унаследован*** от типа *человек*. При наследовании список *типа-отца* ***дописывается*** новыми типовыми элементами и так получается список *типа-ребенка*. Наследование объединяет все типы (как вершины) в *дерево наследования типов* (ДНТ).

У интуитивного объекта есть один и только один тип-центр (одна вершина ДНТ) и один и только один тип-множество (поддерево ДНТ, содержащее корень и тип-центр). Например *человек* Иванов может быть *сотрудником* и *студентом.* Тип-множество объекта *Иванов* это {объект, человек, сотрудник, студент}. Любой элемент этого множества может быть объявлен тип - центром. В течение жизни объекта его тип-множество и тип-центр могут меняться. При этом происходит приобретение и/или потеря данных.

Тип-вершина может получить ***ярлык*** (Синонимы: *тень*, *слабая копия*). Число ярлыков не ограничено, но у каждого ярлыка один и только один ***оригинал***. *Ярлык от ярлыка* это *ярлык* от оригинала первого ярлыка. Ярлык можно добавить в ДНТ. *Под ярлык* ничего добавить нельзя. *Нельзя* добавить ярлык так, чтобы путь от ярлыка в *корень* ДНТ проходил *через* оригинал. Путь из оригинала в корень (***главный путь***) один и только один. Если разрешить *перескакивать из оригинала на ярлык*, то получится ***слабый путь*** в корень. Слабых путей может быть несколько.

Если у объекта А тип центр Т, то все типы-вершины на пути из Т в корень ДНТ *обязательно* входят в тип-множество объекта А. При демонстрации объекта А сперва демонстрируется список элементов главного пути. Затем демонстрируются элементы лежащие на слабых путях из Т в корень, если они не попали в главный путь и если эти тип-вершины попали в тип-множество объекта А. Тип-множество может включать больше вершин, но их элементы не демонстрируются.

Каждая тип вершина имеет свой список функций (возможно пустой). Этот список дописывается по тем же правилам.

Права пользователей на создание, удаление, изменение объектов управляются еще двумя деревьями: деревом объектов (ДО) и деревом вложенности типов (ДВТ). ДВТ построено на тех же оригинал - вершинах, что и ДНТ и имеет тот же корень. ДО содержит все объекты. Корень ДО имеет тип - центром корень ДВТ. Пользователи – тоже объекты и тоже входят в ДО. Когда пользователь входит в систему, он видит поддерево ДО начиная с себя как с корня. Все это поддерево – его *зона ответственности*. В этой зоне он может создавать, изменять, удалять, перемещать объекты. Он может менять их тип, но так, чтобы объединение всех тип-множеств его объектов не увеличивалось (это объединение называется *зоной компетенции*)

Можно создавать ярлыки в ДО и ДВТ (по тем же правилам – абзац третий).

Создать объект Б типа ТБ под объектом А типа ТА можно если и только если (А лежит в зоне ответственности **и (**в ДВТ вершина ТБ (или ее ярлык) - прямой потомок вершины ТА **или** ТБ = *папка* **или ((**ТА=ТБ **и** ТА - *рекурсивный тип* **))**

Если поместить ярлык объекта А в зону ответственности пользователя П, то это увеличит зону ответственности П на одну вершину, а *зону видимости* пользователя П на все поддерево с корнем в А. В зоне видимости можно видеть данные, но нельзя ничего менять.

Управление данными с помощью ДО, ДВТ и ДНТ позволяет строить и модифицировать систему без программистов. Только 4% работ требует знание SQL и 1% - знание JAVA. Типами управляет один человек – администратор = архитектор. Как пользователь он соответствует корню ДО.

Данная модель реализована на основе бесплатного ПО (http-сервер, СУБД) с применением JAVA/JSP и получила название FTS. Основное назначение – единая система управления предприятием. Экспериментально реализовано несколько задач.

В двух школах Петрозаводска уже год применяется Система Анкетирования и Тестирования (CAT) на основе FTS. Опыт показал *очень* высокую *гибкость* системы и ее *дружелюбное поведение*.

По затратам на создание и модификацию ИС технология FTS аналогов не имеет. (Я не рассматриваю область, где данные вводит автомат. Там объекты однотипны, структуры вечны, модификация исключена)