Лекция 3

Диаграммы классов языка UML

Что сегодня рассмотрим

Рассмотрим основные понятия диаграмм классов языка UML и возможности применения этой диаграммной модели для проектирования реляционных БД.

UML

UML (Unified Modeling Language) является языком объектно-ориентированного моделирования.

Про сам язык — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML

Язык UML состоит из нескольких подъязыков (они все графические, диаграммные), в число которых входит подъязык диаграмм классов. Эти подъязыки (спецификации) называются метамоделями. В терминах UML концептуальная схема называется моделью. Модель может быть представлена из метамоделей.

UML позволяет моделировать разные виды систем: чисто программные, чисто аппаратные, программно-аппаратные, смешанные (включающие деятельность людей) и т.д.

Основные понятия языка UML

Диаграмма классов (в терминологии UML) - это диаграмма, на которой показан набор классов (и некоторых других сущностей, не имеющих явного отношения к проектированию БД), а также связей между этими классами. Диаграмма классов может включать комментарии и ограничения.

Как и в модели ER-диаграмм, в UML для родового обозначения связей используется термин «связь».

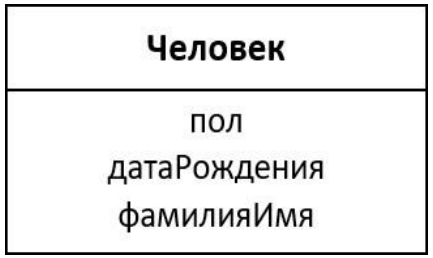
Для диаграмм классов UML могут задаваться ограничения на естественном языке или же на языке объектных ограничений OCL (Object Constraints Language).

Основные понятия языка UML

Класс - это именованное описание совокупности объектов с общими атрибутами, операциями, связями и семантикой.

Атрибут класса - это именованное свойство класса, описывающее множество значений, которые могут принимать экземпляры этого свойства. Множество всех атрибутов класса описывает структуру этого класса.

Класс может иметь любое число атрибутов (в том числе ноль). Свойство, выражаемое атрибутом, является свойством моделируемой сущности, общим для всех объектов данного класса, т.е. атрибут является абстракцией состояния объекта.

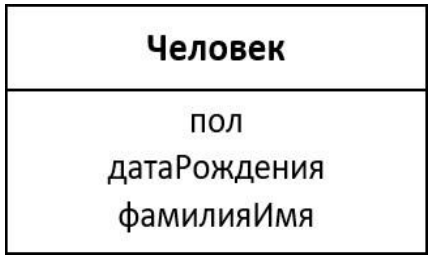


Основные понятия языка UML

У каждого класса должно быть уникальное имя (текстовая строка).

Имена атрибутов представляются в разделе класса, расположенном под именем класса. На практике для имен атрибутов рекомендуется использовать короткие прилагательные и существительные, отражающие смысл соответствующего свойства класса.

Первое слово в имени атрибута рекомендуется писать с прописной буквы, а все остальные слова – с заглавной.



Основные понятия языка UML

Операция класса - это именованная услуга, которую можно запросить у любого объекта этого класса. Операция – это то, что можно делать с объектом. Класс может содержать любое число операций (в частности, ни одной). Набор операций класса является общим для всех объектов данного класса.

Справа показан класс Человек с тремя операциями

Основные понятия языка UML

Операции класса определяются в разделе, расположенном ниже раздела с атрибутами. Можно ограничиться только указанием имен операций, оставив детальную спецификацию выполнения операций на более поздние этапы моделирования.

Для именования операций рекомендуется использовать глаголы, соответствующие ожидаемому поведению объектов данного класса. Описание операции может также содержать имена и типы всех параметров (и тип возвращаемого значения).

Основные понятия языка UML

- выдатьВозраст() - возвращает возраст человека, как разность текущей даты и значения атрибута датаРождения
- сохранитьТекущийДоход() - позволяет зафиксировать в состоянии объекта сумму и дату поступления дохода данного человека
- выдатьОбщийДоход() - возвращает суммарный доход данного человека за всё его время работы

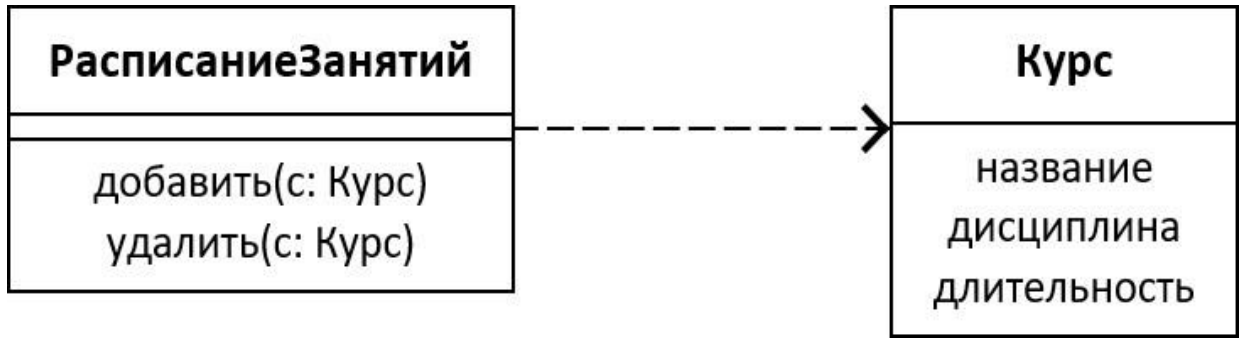
Основные понятия языка UML

В диаграмме классов могут участвовать связи трёх категорий:

- зависимость (dependency)
- обобщение (generalization)
- ассоциация (association)

Связь-зависимость

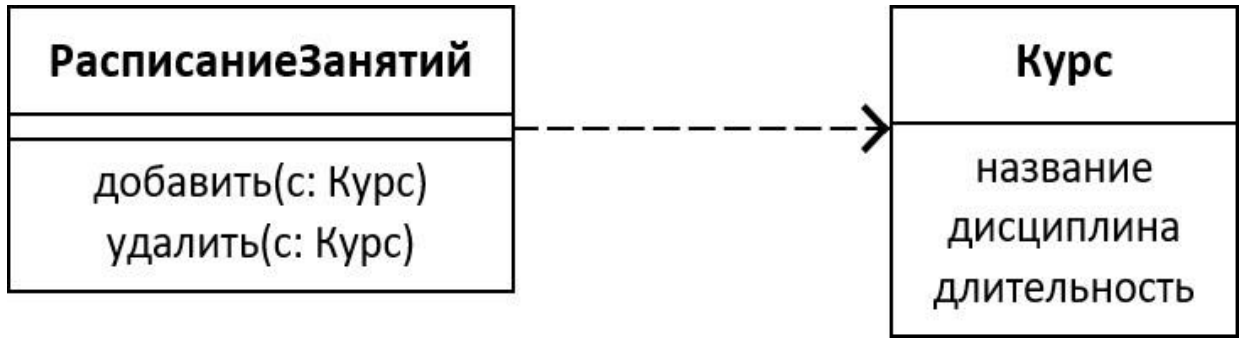
Зависимость - это связь, показывающая, что изменение в описании одного класса может повлиять на поведение другого класса, использующего первый класс. Как правило, зависимость отражает тот факт, что в сигнатуре операции одного класса параметром является объект другого класса.



Связь-зависимость

Понятно, что если интерфейс второго класса изменяется, это влияет на поведение объектов первого класса. Так, при изменении интерфейса класса Курс изменится поведение объектов класса РасписаниеЗанятий.

Связи-зависимости существенны для объектно-ориентированных систем, для РБД применения нет.



Связь-обобщение

Обобщение - это связь между суперклассом (родителем) и подклассом (потомком). Обобщения иногда называют связями «is a», имея в виду, что класс-потомок является частным случаем класса-предка.

Связь-обобщение

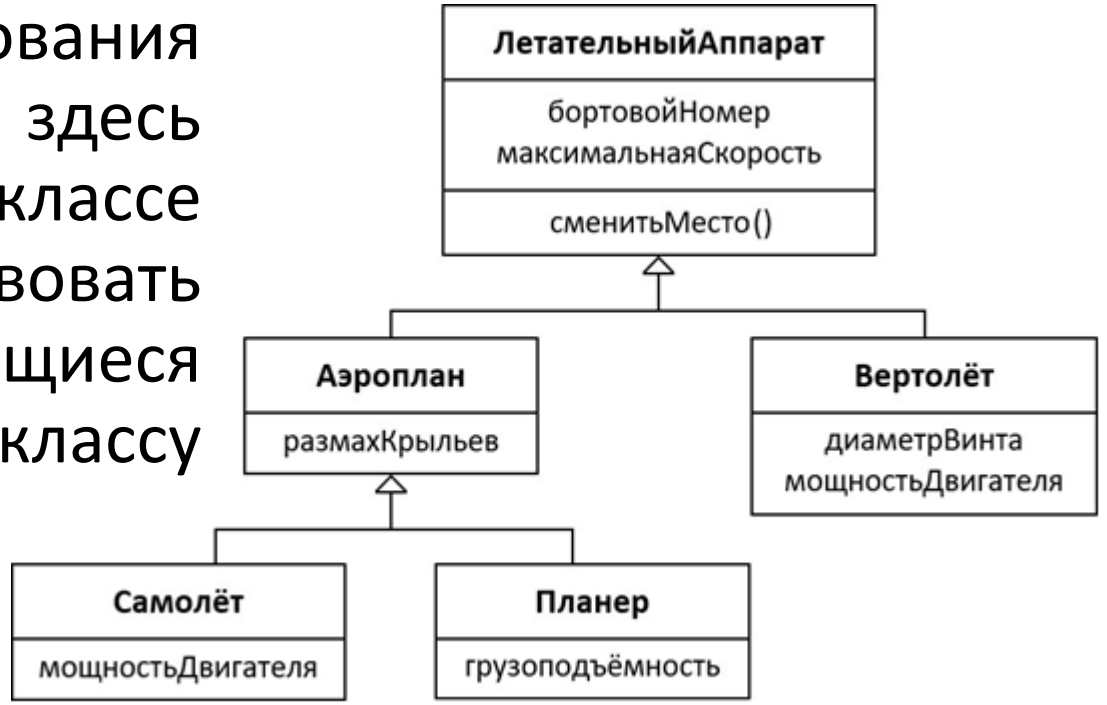
Класс-потомок наследует все атрибуты и операции класса-предка, но в нем могут быть определены дополнительные атрибуты и операции.

Объекты класса-потомка могут использоваться везде, где могут использоваться объекты класса-предка. Это свойство называют полиморфизмом по включению, имея в виду, что объекты потомка можно считать включаемыми во множество объектов класса-предка.

Связь-обобщение

На рисунке показан пример иерархии одиночного наследования: у каждого подкласса имеется только один суперкласс.

В отличие от механизма наследования типов сущностей ER-модели здесь отсутствует класс ПРОЧИЕ, т.е. в классе ЛетательныйАппарат могут присутствовать «собственные» объекты, не относящиеся ни к классу Аэроплан, ни к классу Вертолёт.

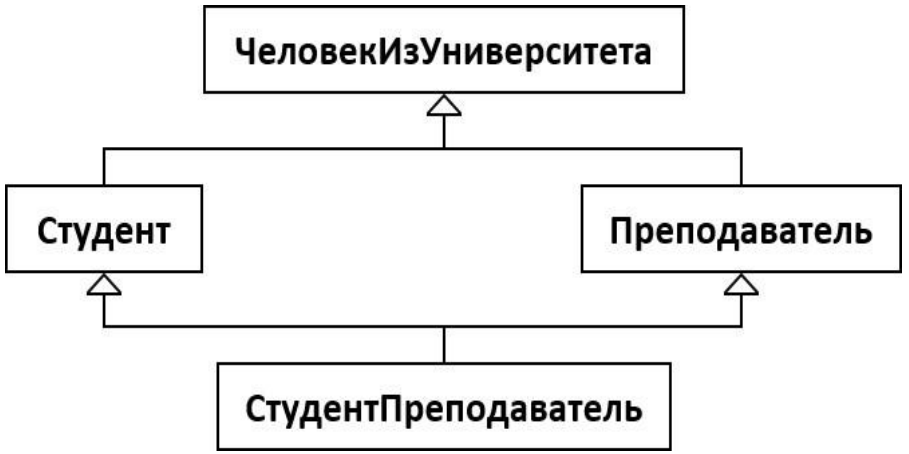


Связь-обобщение

Обычно при разработке одиночного наследования достаточно, но в диаграммах классов допускается и множественное наследование, когда один подкласс определяется на основе нескольких суперклассов.

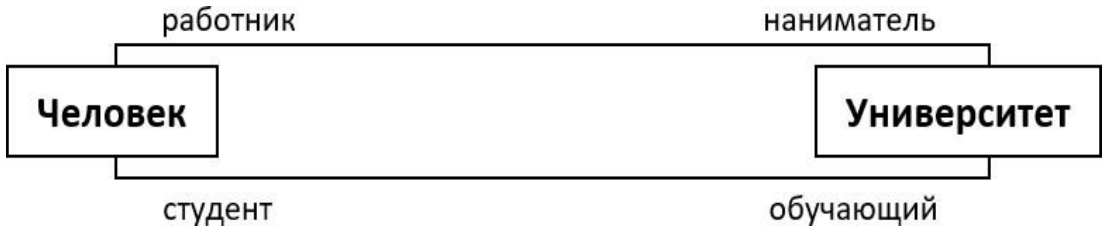
Связь-обобщение

На рисунке классы Студент и Преподаватель порождены из одного суперкласса ЧеловекИзУниверситета. Бывают случаи, когда студенты начинают преподавать, т.е. логически один и тот же объект класса ЧеловекИзУниверситета принадлежит сразу двум классам: и Студенту и Преподавателю. Тогда мы можем определить класс СтудентПреподаватель путем множественного наследования от суперклассов Студент и Преподаватель.



Связь-ассоциация

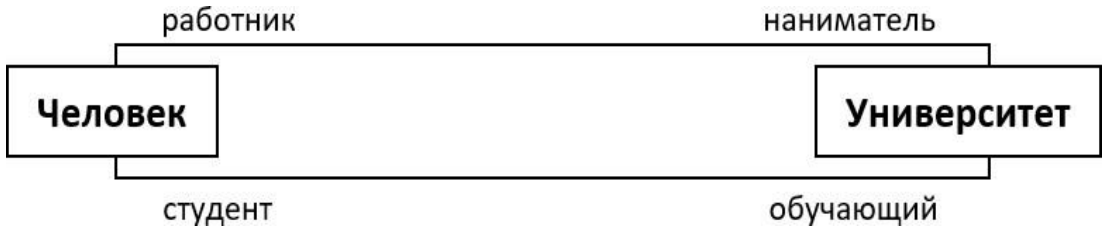
Ассоциация - это структурная связь, показывающая, что объекты одного класса некоторым образом связаны с объектами другого или того же самого класса. В ассоциации могут связываться два класса, и тогда она называется бинарной. Но допускается также создание ассоциаций, связывающих сразу n классов (они называются n-арными ассоциациями).



Связь-ассоциация

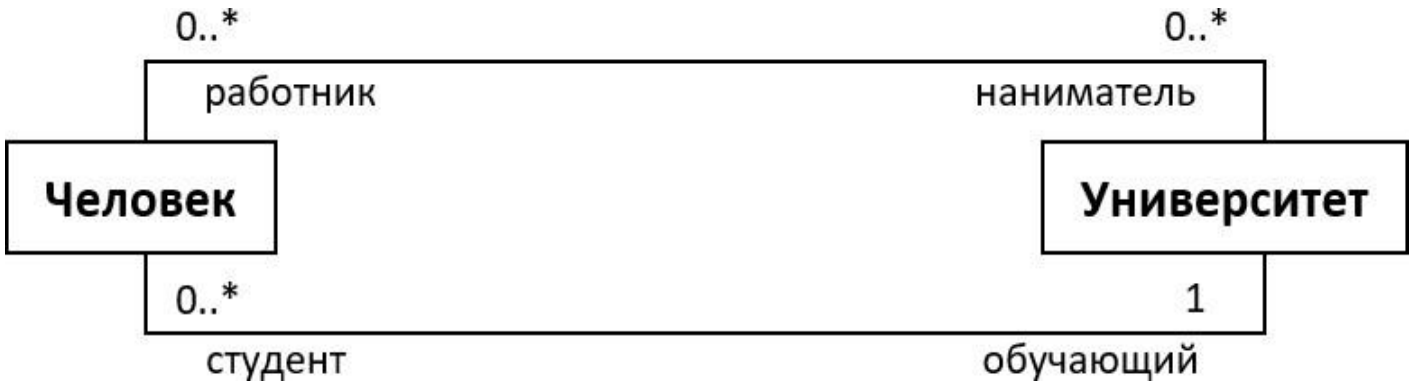
Ассоциации может быть присвоено имя, характеризующее природу связи. Смысл имени уточняется с помощью черного треугольника, который указывает направление чтения имя связи.

Другой способ именования ассоциации - это подпись роли каждого класса, участвующего в этой связи. Роль класса обозначает роль класса в данной связи.



Связь-ассоциация

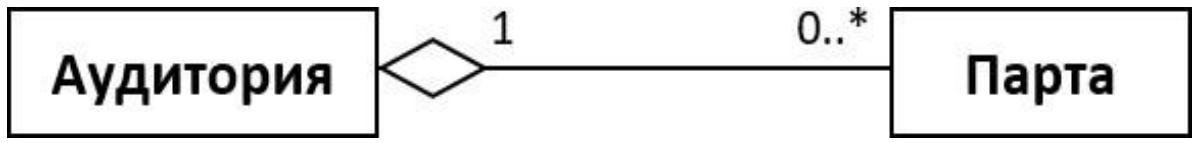
Кратность (multiplicity) роли ассоциации - это характеристика, указывающая, сколько объектов класса с данной ролью должно участвовать в каждом экземпляре ассоциации. Кратность роли ассоциации задаётся указанием конкретного числа или диапазона рядом с концом связи.



Связь-ассоциация

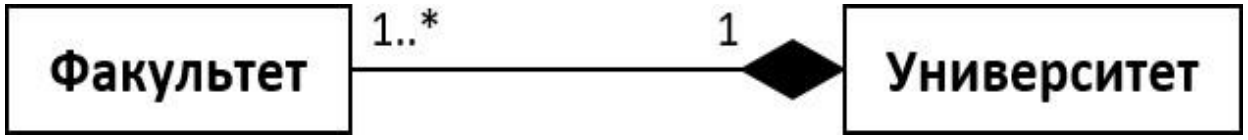
Иногда в диаграмме классов требуется отразить тот факт, что ассоциация между двумя классами имеет специальный вид «часть-целое». В этом случае класс «целое» имеет более высокий концептуальный уровень, чем класс «часть».

Ассоциация такого рода называется *агрегатной.*



Связь-ассоциация

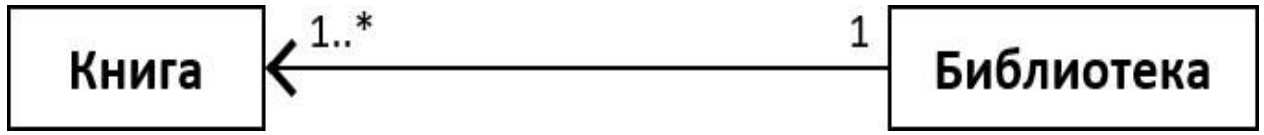
Бывают случаи, когда связь «части» и «целого» настолько сильна, что уничтожение «целого» приводит к уничтожению всех его «частей». Агрегатные ассоциации, обладающие таким свойством, называются *композитными (или композициями).* При наличии композиции объект-часть может быть частью только одного объекта-целого (композита). При обычной агрегатной ассоциации «часть» может одновременно принадлежать нескольким «целым».



Связь-ассоциация

При наличии простой ассоциации между двумя классами предполагается возможность навигации между объектами, входящими в один экземпляр ассоциации. Другими словами, если не оговорено иное, то навигация по ассоциации может проводиться в обоих направлениях.

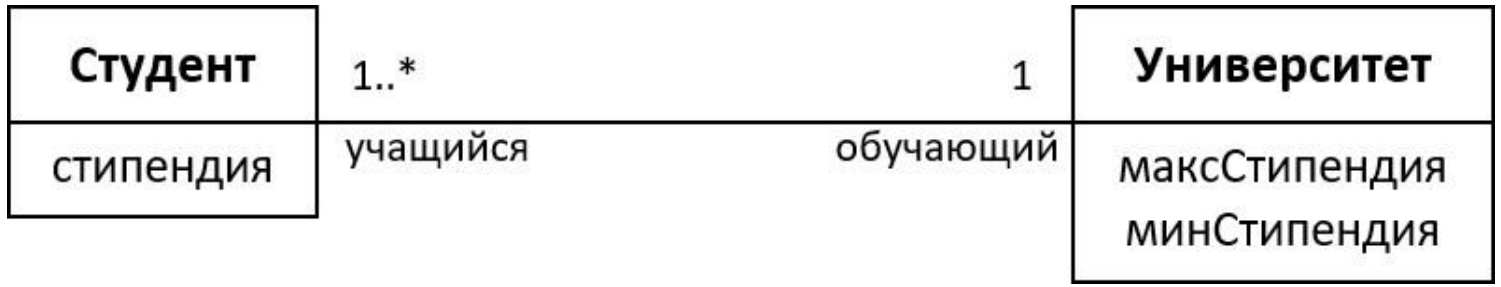
Однако бывают случаи, когда желательно ограничить направление навигации для некоторых ассоциаций.



Ограничения целостности и язык OCL

В диаграммах классов могут указываться ограничения целостности, которые должны поддерживаться в проектируемой БД. В UML допускаются два способа определения ограничений: на естественном языке и на языке OCL.

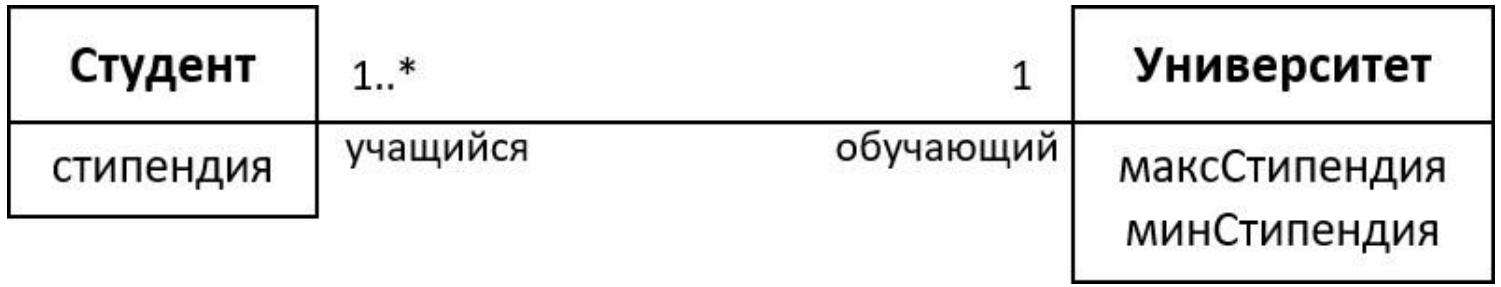
Пусть мы хотим, чтобы значение атрибута «стипендия» класса Студент принадлежало отрезку [минСтипендия; максСтипендия].



Ограничения целостности и язык OCL

Язык OCL (Object Constraints Language) позволяет описывать подобного рода ограничения. На языке OCL данный пример записывается следующим образом:

contextCтудент inv:
self.стипендия >= self.обучающий.минСтипендия
and self.стипендия <= self.обучающий.максСтипендия



Ограничения целостности и язык OCL

Язык OCL предназначен, главным образом, для определения ограничений целостности данных, соответствующих модели, которая представлена в терминах диаграммы классов UML. OCL может применяться для определения:

- ограничений, описывающих пред- и постусловия операций классов
- ограничений, представляющих собой инварианты классов

С точки зрения определения ограничений целостности БД более важны средства определения инвариантов классов.

Понятие инварианта класса

Под инвариантом класса в OCL понимается условие, которому должны удовлетворять все объекты данного класса. Если говорить более точно, инвариант класса – это логическое выражение, при вычислении которого для любого объекта данного класса в течение всего времени существования этого объекта получается булевское значение true.

При определении инварианта требуется указать имя класса и выражение, определяющее инвариант указанного класса. Синтаксически это выглядит следующим образом:

context <class_name> inv: <OCL-выражение>

Понятие инварианта класса

Синтаксически это выглядит следующим образом:

context <class_name> inv: <OCL-выражение>

- где class-name – имя класса, для которого определяется инвариант,
- inv – ключевое слово, говорящее о том, что определяется именно инвариант, а не ограничение другого вида,
- context – ключевое слово, которое говорит о том, что контекстом следующего после двоеточия OCL-выражения являются объекты класса <class-name>

Понятие инварианта класса

OCL является типизированным языком, поэтому у каждого выражения имеется некоторый тип. OCL-выражение в инварианте класса должно быть логического типа.

В общем случае OCL-выражение в определении инварианта основывается на композиции операций. В спецификации языка эти операции условно разделены на следующие группы:

- операции над предопределенными типами данных
- операции над объектами
- операции над коллекциями (множествами, мультимножествами и последовательностями)

Операции над предопределенными типами данных

В OCL в число предопределённых типов входят Boolean, Integer, Real и String.

Тип данных	Операции
Boolean	and, or, xor, not, implies, if-then-else
Integer	*, +, -, /, abs(), операции сравнения
Real	*, +, -, /, floor(), операции сравнения
String	concat(), size(), substring()

Операции над объектами

- Операция получения значения атрибута
  - возвращает текущее значение соответствующего атрибута
- Операция перехода по экземпляру связи-ассоциации
  - возвращает коллекцию всех объектов, которые ассоциированы с данным объектом через указываемый экземпляр ассоциации
- Операция вызова операции класса

Используется точечная нотация. Например: <объект>.<атрибут>

Операции над коллекциями

В OCL поддерживается обширный набор операций над значениями коллекционных типов данных (множествами, мультимножествами, последовательностями), но мы рассмотрим только важные при проектировании БД:

- операция select
- операция collect
- операции exists, forAll, count
- операции union, intersect, symmetricDifference

Используется стрелочная нотация: <коллекция> → <имя операции>

Примеры инвариантов

Пример 1. Определить ограничение «возраст служащих должен быть больше 18 и меньше 100 лет»:

context Служащий inv:
self.возраст > 18
and self.возраст < 100

Примеры инвариантов

Пример 2. Определить ограничение «в отделах с номерами больше 5 должны работать сотрудники старше 30 лет»:

context Отдел inv:
self.номер <= 5
or self.служащий → select (возраст <= 30) → count () = 0

Примеры инвариантов

Пример 2. Определить ограничение «в отделах с номерами больше 5 должны работать сотрудники старше 30 лет»:

context Служащий inv:
self.возраст > 30
or self.отдел.номер <= 5

Плюсы и минусы использования языка OCL при проектировании РБД

Язык OCL позволяет формально и однозначно определять ограничения целостности БД в терминах её концептуальной схемы.

К отрицательным сторонам использования OCL относится сложность языка и неочевидность некоторых его конструкций. Кроме того, строгость синтаксиса и линейная форма языка в некотором роде противоречат наглядности и интуитивной ясности диаграммной части UML.

Получение схемы РБД из диаграммы классов UML

Если не обращать внимания на различия в терминологии, то здесь выполняются практически те же шаги, что и в случае преобразования в схему реляционной БД ER-диаграммы.

Есть разве что моменты, связанные со спецификой диаграмм классов:

- Операции в классах
- Кратности ролей
- Однонаправленность связей
- Возможности OCL

Проектирование РБД

В задаче *проектирования реляционных БД* использование ER-диаграмм и языка UML будет отличаться в основном терминологией. ER-модель концептуально проще UML, в ней меньше понятий, терминов, вариантов применения. Но т.к. ER-модели разрабатывались именно для поддержки проектирования реляционных БД, они не содержат возможностей, выходящих за пределы этой задачи.

Язык UML же более универсальный, и не ограничивается только возможностями диаграмм классов.

Выбор конкретной концептуальной модели – это вопрос вкуса и сложившихся обстоятельств.

Итоги

В контексте проектирования РБД методы проектирования, основанные на использовании ER-диаграмм, и методы, основанные на использовании языка UML, различаются, главным образом, лишь терминологией.

Модель UML концептуально сложнее ER-модели, в ней больше понятий, терминов, вариантов применения, избыточных с точки зрения проектирования реляционных БД. В то же время, язык UML более универсальный, и не ограничивается только возможностями диаграмм классов.