11. Элементы ER модели (модели "сущность-связь"). Атрибуты, сущности. Виды и

## типы связей между сущностями. Прямое и обратное проектирование

В реальном проектировании структуры базы данных применяются другой метод - так называемое, ***семантическое моделирование***. Семантическое моделирование представляет собой моделирование структуры данных, опираясь на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность-связь*** (***ER - Entity-Relationship***).

#### Основные понятия ER-диаграмм

**Сущность (Entity)** это класс однотипных объектов, информация о которых должна быть учтена в модели.

Каждая сущность должна иметь наименование (Name), выраженное существительным в единственном числе.

Примерами сущностей могут быть такие классы объектов как "Сотрудник", "Дисциплина", "Кафедра".

Каждая сущность в модели изображается в виде прямоугольника с наименованием.

**Экземпляр сущности** это конкретный представитель данной сущности.

Например, представителем сущности "Сотрудник" может быть "Сотрудник Буланкин".

Экземпляры сущностей должны быть *различимы*, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

**Атрибут сущности (Attribute)** это именованная характеристика, являющаяся некоторым свойством сущности.

Наименование атрибута (Attribute Name) должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Примерами атрибутов сущности "Сотрудник" могут быть такие атрибуты как "Табельный номер", "Фамилия" и т.д.

Атрибуты изображаются в пределах прямоугольника, определяющего сущность.

**Ключ сущности (Primary Key)**- это *неизбыточный* набор атрибутов, значения которых в совокупности являются *уникальными* для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность. Сущность может иметь несколько различных ключей.

Ключевые атрибуты изображаются на диаграмме подчеркиванием.

**Связь (Relationship)** это некоторая ассоциация между *двумя* сущностями. Одна сущность может быть связана с другой сущностью или сама с собою.

Связи позволяют по одной сущности находить другие сущности, связанные с нею.

Например, связи между сущностями могут выражаться следующими фразами - "СОТРУДНИКИ читают ДИСЦИПЛИНЫ ", "каждый СОТРУДНИК принадлежит одной КАФЕДРЕ".

Графически связь изображается линией, соединяющей две сущности.

Каждая связь имеет два конца и одно или два **наименования (Verb Phrase)**. Наименование обычно выражается в неопределенной глагольной форме: "находиться", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих **типов связи(Cardinality)**:

Связь типа **один-к-одному (One)** означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь **один-ко-многим (More)** означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи.

Левая сущность (со стороны "один") называется **родительской (Parent-to-Child)**, правая (со стороны "много") – **дочерней (Child-to- Parent)**. Пример такой связи приведен на.

Связь типа **много-ко-многим** означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Является *временным* типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Каждая связь может иметь одну из двух **модальностей связи (Relationship Type)**:

Модальность **может (Identifying Relationship)** означает, что экземпляр одной сущности *может быть связан* с одним или несколькими экземплярами другой сущности, *а может быть и не связан* ни с одним экземпляром.

Модальность **должен (Identifying Relationship)** означает, что экземпляр одной сущности *обязан быть связан не менее чем с одним* экземпляром другой сущности.

Связь может иметь *разную модальность* с разных концов.

Описанный графический синтаксис позволяет *однозначно* читать диаграммы, пользуясь следующей схемой построения фраз:

<Каждый экземпляр СУЩНОСТИ 1> <МОДАЛЬНОСТЬ СВЯЗИ> <НАИМЕНОВАНИЕ СВЯЗИ> <ТИП СВЯЗИ> <экземпляр СУЩНОСТИ 2>.

Каждая связь может быть прочитана как слева направо, так и справа налево. Связь на рис. 13 читается так:

Слева направо: "каждый сотрудник может вести несколько дисциплин".

Справа налево: "Каждую дисциплину обязан вести ровно один сотруднику".

***При разработке ER-моделей мы должны получить следующую информацию о предметной области****:*

1) Список сущностей предметной области.

2) Список атрибутов сущностей.

3) Описание взаимосвязей между сущностями.

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы. Легко заметить, что полученные таблицы сразу находятся в 3НФ.

Выводы: Реальным средством моделирования данных является не формальный метод нормализации отношений, а так называемое **семантическое моделирование**. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность-связь*** (***ER - Entity-Relationship***). Диаграммы сущность-связь позволяют использовать наглядные графические обозначения для моделирования сущностей и их взаимосвязей.

Различают **логические** и **физические** ER-диаграммы.

**Логические** не учитывают особенностей конкретных СУБД.

**Физические** диаграммы строятся по логическим и представляют собой прообраз конкретной базы данных.

Сущности, определенные в концептуальной диаграмме становятся таблицами, атрибуты становятся колонками таблиц (при этом учитываются допустимые для данной СУБД типы данных и наименования столбцов), связи реализуются путем **миграции** ключевых атрибутов родительских сущностей и создания внешних ключей.

Процесс генерации физической схемы базы данных из логической модели данных называется прямым проектированием (Forward Engineering). Когда Вы генерируете физическую схему, ERwin позволяет Вам включать триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в Вашей СУБД.

Аналогично, процесс генерации логической модели из физической базы данных называется обратным проектированием (Reverse Engineering). ERwin позволяет Вам быстро создать модель данных путем обратного проектирования имеющейся базы данных. После того как Вы создали модель ERwin, Вы можете произвести обратное проектирование структуры базы данных, а затем легко перенести его в другой формат базы данных.