ГЛАВА 2. КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ БАЗЫ ДАННЫХ

2.1. Сложный пример предметной области

Чтобы исследовать различные аспекты использования СУБД, мы рассмотрим сложный пример, приближенный к действительности,– задачу зачисления абитуриентов в вузы. Каждый вуз решает данную задачу по-своему, чаще всего без использования баз данных. Наш пример не является реальным примером зачисления абитуриентов в ННГУ или другой вуз, однако очень близок к тем задачам, которые стоят в действительности перед приемными комиссиями ННГУ и других вузов.

До начала приема документов у абитуриентов формируются основные документы вуза, регламентирующие прием. Обычно это правила приема (они составляются и утверждаются ежегодно), расписание экзаменов, списки признаков, наличие которых у абитуриента учитывается приемной комиссией при зачислении (дает абитуриенту дополнительное преимущество), и многие другие документы. Причиной ежегодного изменения правил приема и других регламентирующих документов является добавление новых специальностей и специализаций, изменение законов Российской Федерации в части предоставления льгот некоторым категориям граждан при поступлении в вузы, желание вуза улучшить процедуру зачисления, сделать ее более справедливой.

Информационная система, которая описана ниже, позволяет решить важные для приемной комиссии задачи – упорядочить доступ к информации, сделать информацию доступной в короткие сроки, разгрузить работников приемной комиссии и, самое главное, избежать ошибок при зачислении.

Проведение зачисления в вуз осуществляется в несколько этапов.

Первый этап – прием документов у абитуриента. При приеме документов абитуриент заявляет о своем желании участвовать в конкурсе на конкретную специальность, а вернее, на учебную программу. Он также может указать в заявлении, что в случае непрохождения на данную специальность желает участвовать в конкурсе на другие специальности этого же или другого факультета. Для каждой выбранной специальности абитуриент указывает ее приоритет для себя.

Обычно вузы проводят сквозной конкурс. Это означает, что абитуриент, не набравший достаточного количества баллов на выбранную им специальность, может участвовать в конкурсе на другие специальности, указанные им при поступлении.

Вуз обычно предоставляет выбор предметов, которые может сдавать абитуриент. Например, на экономический факультет ННГУ можно сдавать либо математику, либо информатику. Абитуриент при поступлении указывает, какой именно экзамен он будет сдавать.

Кроме информации о приоритете специальностей и о том, какой экзамен он будет сдавать, абитуриент сообщает о себе некоторые дополнительные сведения (фамилию, имя, отчество, паспортные данные, предшествующее образование, гражданство и так далее), а также предоставляет документы, подтверждающие его право на льготы и преимущества.

Второй этап – проведение вступительных испытаний. Испытания проводятся в форме экзаменов и последующего собеседования. На собеседовании работник приемной комиссии вместе с абитуриентом просматривает все документы и формирует балл к зачислению по каждой специальности, на которую претендует абитуриент.

Третий этап – зачисление абитуриентов, показавших лучшие результаты на вступительных испытаниях и имеющих другие документы, предоставляющие им преимущества и льготы при зачислении. В связи с тем, что законодательство России выделяет множество категорий граждан, имеющих льготы и преимущества при поступлении в вуз, приемная комиссия формирует приказы, состоящие из большого количества пунктов, в каждом из которых указывается, в соответствии с каким нормативным документом зачисляется данный абитуриент.

Необходимо отметить, что основное количество абитуриентов не имеют льгот и зачисляются исходя из того, сколько баллов они набрали на вступительных испытаниях и какие дополнительные документы они имеют. Говорят, что абитуриенты, не имеющие льгот, участвуют в общем конкурсе. Именно общий конкурс является основным для вуза. Именно информация об абитуриентах, участвующих в общем конкурсе, является объемной и требует автоматизированной обработки.

Можно также отметить, что поскольку задача зачисления актуальна для всех вузов, мы должны учесть ее возможное расширение, касающееся возможности подавать документы не только в один вуз, а сразу в несколько вузов. В самом деле, было бы вполне правильно, что абитуриент выбирает прежде всего специальность, а не вуз. Предпо-

ложим, что абитуриент хочет изучать информационные системы. Однако специальности, связанные с информационными системами, есть во многих вузах города. Логично было бы дать возможность абитуриенту сформулировать свой выбор следующим образом: «Для меня лучшим вариантом было бы поступить на ВМК ННГУ, но если я не пройду туда по конкурсу, то я бы хотел участвовать в конкурсе на радиофак НГТУ». Такой возможности у абитуриента пока нет, но она может появиться в том случае, если в качестве вступительных экзаменов будут засчитываться результаты централизованного тестирования.

2.2. Способы описания предметной области

Введем основные понятия, с помощью которых описывается предметная область.

Сущность (Entity) или объект – то, о чем будет накапливаться информация в информационной системе (нечто такое, за чем пользователь хотел бы наблюдать).

Если в системе обрабатывается информация об абитуриентах, сущностью может являться абитуриент, если обрабатывается информация об экзамене, то сущность – экзамен и т.п. Каждая сущность обладает определенным набором свойств (рассматриваем только свойства, представляющие интерес для пользователей в рамках проводимого исследования), которые запоминаются в информационной системе.

Так, например, в качестве свойств сущности АБИТУРИЕНТ можно указать фамилию, дату рождения, место рождения, в качестве свойств сущности ЭКЗАМЕН – предмет, дату проведения экзамена, экзаменаторов.

Совокупность сущностей, характеризующихся в информационной системе одним и тем же перечнем свойств, называется классом сущностей (набором объектов). Так, например, совокупность всех сущностей АБИТУРИЕНТ составляет класс сущностей АБИТУРИЕНТ, совокупность всех сущностей ЭКЗАМЕН составляет класс сущностей ЭКЗАМЕН.

Класс сущностей описывается перечнем свойств сущностей, составляющих этот класс.

Экземпляром сущности будем называть конкретную сущность (сущность с конкретными значениями соответствующих свойств). Пример класса сущностей АБИТУРИЕНТ и конкретного экземпляра сущности показан на рис. 15.

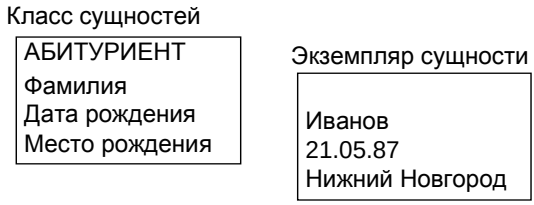


Рис.15. Класс сущностей и экземпляр сущности

Взаимоотношения сущностей выражаются связями (Relationships). Различают классы связей и экземпляры связей. Классы связей – это взаимоотношения между классами сущностей, а экземпляры связи – взаимоотношения между экземплярами сущностей.

Класс связей может затрагивать несколько классов сущностей. Число классов сущностей, участвующих в связи, называется степенью связи $n = 2, 3, \ldots$ Так, например, класс сущностей АБИТУРИЕНТ связан с классом сущностей ЭКЗАМЕН связью «сдает». Степень этой связи равна двум. В качестве примера связи степени три можно указать связь «родители» между тремя классами сущностей МАТЬ, ОТЕЦ, РЕБЕНОК. При n=2 связь называется бинарной.

Рассмотрим классификацию бинарных связей. В зависимости от того, сколько экземпляров сущности одного класса связаны со сколькими экземплярами сущности другого класса, различают следующие типы связей:

- Связь 1:1. Одиночный экземпляр сущности одного класса связан с одиночным экземпляром сущности другого класса. Примером является связь «соответствует» между классами сущностей ФАКУЛЬТЕТ и РАСПИСАНИЕ ЭКЗАМЕНОВ НА ФАКУЛЬТЕТ (каждому факультету соответствует свое расписание).
- Связь 1:*M*. Единый экземпляр сущности одного класса связан со многими экземплярами сущности другого класса. Примером является связь «зачисление» между классами сущностей ФАКУЛЬТЕТ и АБИТУРИЕНТ (на один факультет зачисляется много абитуриентов).
- Связь *M*:*N*. Несколько экземпляров сущности одного класса связаны с несколькими экземплярами сущности другого класса. Примером является связь «сдают» между классами сущностей АБИТУРИЕНТ и ЭКЗАМЕН (каждый абитуриент сдает несколько экзаменов, и каждый экзамен сдают много абитуриентов).

Числа, описывающие типы бинарных связей (1:1, 1:*M*, *M*:*N*), обозначают максимальное количество сущностей на каждой стороне связи. Эти числа называются максимальными кардинальными числами, а соответствующая пара чисел называется максимальной кардинальностью.

2.3. Описание информационного представления предметной области

В качестве основного понятия для описания предметной области, как уже отмечалось, используется понятие сущности (объекта), характеризуемой набором определенных свойств. Для информационного описания сущности вводится понятие атрибута.

Атрибут – поименованное свойство (характеристика) сущности. Атрибут представляет собой информационное отображение свойства сущности и принимает конкретное значение из множества допустимых значений. Так, например, для сущности АБИТУРИЕНТ атрибут «фамилия» у конкретного экземпляра сущности принимает конкретное значение «Иванов».

Таким образом, атрибут представляет информационное описание количественных или качественных свойств сущности, описывает состояние сущности, позволяет идентифицировать сущность. Информация о сущности представляется совокупностью атрибутов. Такую совокупность атрибутов часто называют записью об объекте.

Другим основным понятием для описания предметной области является понятие связи. Разные способы информационного описания связей будут рассмотрены позднее. В данном разделе отметим, что, в частности, для представления связей между экземплярами сущностей могут использоваться атрибуты. В этом случае связь устанавливается путем включения в совокупность атрибутов сущности атрибута, однозначно идентифицирующего экземпляр сущности, находящийся в отношении с исходным экземпляром сущности.

Так, рассмотрим класс сущностей ФАКУЛЬТЕТ, представленный одной совокупностью атрибутов (название, номер), и класс сущностей РАСПИСАНИЕ ЭКЗАМЕНОВ НА ФАКУЛЬТЕТ, представленный другой совокупностью атрибутов (название экзамена 1, дата экзамена 1, название экзамена 2, дата экзамена 2, название экзамена 3, дата экзамена 3). Для представления связи «экзамены» (тип связи 1:1) в совокупность атрибутов РАСПИСАНИЕ ЭКЗАМЕНОВ НА ФАКУЛЬТЕТ можно включить атрибут «название факультета».

2.4. Описание информационных потребностей пользователя

Информационные потребности пользователя представляются в виде запросов к информации об экземплярах сущностей. Запросы бывают простые и сложные. Сложные запросы составляются из простых с помощью логических операторов *.and., .or.* или *.not.*, поэтому без ограничения общности будем говорить только о простых запросах. Для реализации подавляющего числа запросов пользователю прежде всего необходимо найти интересующий его экземпляр сущности (с целью обработки, корректировки, удаления). Для идентификации экземпляра используется один или несколько атрибутов.

Атрибут или множество атрибутов, значения которых однозначно идентифицируют экземпляр записи, называется потенциальным ключом.

Потенциальный ключ, который выбран в качестве основного идентификатора, называется первичным ключом.

Потенциальный ключ, который состоит из двух или более атрибутов, называется составным ключом.

Часто для поиска используется атрибут или множество атрибутов, которые не идентифицируют экземпляр сущности единственным образом и характеризуют определенную группу записей. Такой атрибут (множество атрибутов) называется вторичным ключом.

Приведем шесть абстрагированных типов простых запросов, относящихся к экземпляру сущности E, атрибутам A и значениям атрибутов V [23].

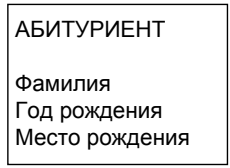
Тип	Форма	Смысл	Пример
1	$A(E) = ?$	Каково значение атрибута A экземпляра сущности E	Место рождения абитуриента Иванова
2	$A(?) \begin{matrix} = \\ \neq \\ < \\ > \end{matrix} V$	Какой экземпляр E имеет значение атрибута A, равное (неравное, меньшее, большее) V	Кто из абитуриентов имеет год рождения <1969 (старше 35 лет)
3	$?(E) \begin{matrix} = \\ \neq \\ < \\ > \end{matrix} V$	Какой атрибут или атрибуты экземпляра E имеют значение, равное (неравное, меньшее, большее) V	За какие экзамены абитуриент Иванов получил оценку «отлично»

4	?(*E*) = ?	Запрос на получение значений всех атрибутов экземпляра *E*	Сообщить всю информацию об абитуриенте Иванове
5	*A*(?) = ?	Перечислить значения данного атрибута для каждого экземпляра	Перечислить названия всех специальностей (сущность СПЕЦИАЛЬНОСТЬ)
6	?(?) $=$ $\neq$ $<$ *V* $>$	Перечислить все атрибуты экземпляров, имеющие значение, равное (неравное, меньшее, большее) *V*	Перечислить все атрибуты абитуриентов, получивших оценку «2»

2.5. Построение ER-диаграмм

Чаще всего концептуальная модель представляется в виде диаграммы сущностей – связей (entity – relationship) или ER-диаграммы. Процесс построения ER-диаграммы называется ER-моделированием. При этом используются следующие классические обозначения.

Класс сущностей представляется в виде четырехугольника. В четырехугольнике записано уникальное имя класса сущности (прописными буквами) и имена атрибутов строчными буквами.



Связи между сущностями обозначаются стрелками, рядом со стрелками указывается имя связи, а также максимальная кардинальность связи (максимальное число сущностей, которые могут участвовать в связи). Чтобы показать, что сущность обязана участвовать в связи (каждый экземпляр должен быть связан с экземпляром другого класса), на линию связи помещают перпендикулярную черту, а чтобы показать, что сущность может (но не обязана) участвовать в связи, на линию связи помещают овал.

При описании сущностей выделяют особые совокупности атрибутов – ключи и внешние ключи. Ключ уникально идентифицирует экземпляр сущности и, вместе с внешним ключом, используется для реализации связей.

На диаграммах атрибуты, входящие в первичный ключ, подчеркиваются (обозначение РК).

Важность понятий ключа и внешнего ключа будет проиллюстрирована далее на примерах.

Приведенные выше обозначения не являются общеупотребительными. Часто производитель программ, позволяющих рисовать ER-диаграммы, использует свою систему обозначений. Например, при использовании программы MS Visio в качестве основной используется так называемая реляционная нотация. В данной системе обозначений связи обозначаются стрелками между сущностями и названия связей не пишутся. Название сущности выделяется цветом, поля, входящие в первичный ключ, отделяются чертой от остальных атрибутов. Обязательные атрибуты отображаются с помощью полужирного шрифта. Кроме того, слева от атрибута указывается, входит ли данный атрибут в первичный ключ, а также является ли атрибут внешним ключом.

На практике использование различных способов записи ER-диаграмм не представляет особой сложности – беглое ознакомление с соответствующим разделом документации позволяет быстро освоить используемую систему обозначений.

2.6. Выявление и моделирование сущностей и связей

При разработке концептуальной модели, прежде всего, следует определить сущности. С этой целью нужно сделать следующее:

- необходимо понять, какая информация должна храниться и обрабатываться и можно ли это определить как сущность;
- присвоить этой сущности имя;
- выявить атрибуты сущности и присвоить им имя.

Выявив сущности, необходимо определить, какие связи имеются между ними.

При определении связей (естественно, рассматриваем только те связи, которые имеют отношение к решаемым задачам обработки данных) необходимо учитывать следующее:

- то, как экземпляр одной сущности связан с экземпляром другой сущности;
- то, как должны быть установлены связи, чтобы была возможность ответа на все запросы пользователей (исходя из их информационных потребностей).

Далее необходимо присвоить связям имена и определить тип связей.

В качестве примера моделирования сущностей и связей рассмотрим небольшой фрагмент предметной области зачисления абитуриентов в вуз, рассмотренный в разделе 2.1.

Наш фрагмент будет относиться к представлению в базе данных информации о специальностях, факультетах и университетах.

Первоначально можно было рассмотреть сущность СПЕЦИАЛЬНОСТЬ, которая включает в себя следующие атрибуты: название специальности, номер специальности по общероссийскому классификатору; факультет, на котором преподается данная специальность; вуз, в состав которого входит факультет, отделение. Рассмотрев данную сущность, мы видим, что значительная часть информации дублируется.

Для иллюстрации дублирования информации возьмем в качестве примера специальность «Прикладная математика». По этой специальности ведут обучение и, соответственно, осуществляют прием на дневное и вечернее отделения два факультета ННГУ. Каждая из этих специальностей уникальна с точки зрения нашей предметной области – зачисление на каждую производится отдельно. Соответственно, каждая специальность будет представлена отдельным экземпляром сущности. Однако информация по названию специальности и номеру специальности по общероссийскому классификатору будет дублироваться. Кроме того, для очной и вечерней форм обучения внутри одного факультета будет дублироваться вся информация, кроме названия отделения. Это признак того, что внутри данной сущности находится несколько сущностей.

С учетом вышесказанного можно выделить следующие сущности и связи.

УЧЕБНАЯ ПРОГРАММА – это сущность, которая описывает конкретную учебную программу. Для нее указаны специальность (название и номер), факультет и отделение, на которых она преподается.

Информация о специальностях – название и номер по общероссийскому классификатору – вынесена в отдельную сущность СПЕЦИАЛЬНОСТЬ.

Информация об отделениях будет находиться в сущности ОТДЕЛЕНИЕ.

Сущность ФАКУЛЬТЕТ определяет название и аббревиатуру факультета, а также вуз, в состав которого входит факультет.

Между сущностью УЧЕБНАЯ ПРОГРАММА и остальными сущностями выделяется связь «характеризуется». Конкретная учебная

программа характеризуется факультетом, специальностью, отделением.

Этот вариант является хорошим с точки зрения нашей предметной области, но возникает вопрос – необходимо ли было выделять отдельную сущность ВУЗ (см. итоговую диаграмму, рис. 16)? Может быть, достаточно было в сущности ФАКУЛЬТЕТ хранить атрибут «вуз»? Ответ на данный вопрос заключается в следующем.

Одно из основных свойств базы данных заключается в ее постоянной модификации в процессе эксплуатации. Вполне логично предположить, что помимо названия вуза, нам будет нужно хранить информацию о других его характеристиках – например, аббревиатуре и адресе вуза. Если это произойдет, то информация будет дублироваться. Таким образом, нужно выделить сущность ВУЗ, которая связана с сущностью ФАКУЛЬТЕТ (а не с сущностью УЧЕБНАЯ ПРОГРАММА!). Связь – «входит в состав». Факультет вычислительной математики и кибернетики (ВМК) входит в состав ННГУ.

Определим типы связей.

Сущность УЧЕБНАЯ ПРОГРАММА характеризуется сущностью СПЕЦИАЛЬНОСТЬ. Каждому экземпляру сущности УЧЕБНАЯ ПРОГРАММА ставится в соответствие один экземпляр сущности СПЕЦИАЛЬНОСТЬ. Каждому экземпляру сущности СПЕЦИАЛЬНОСТЬ соответствует ноль, один или много экземпляров сущности УЧЕБНАЯ ПРОГРАММА. Связь – один к многим, причем для сущности УЧЕБНАЯ ПРОГРАММА связь полная – каждый экземпляр участвует в связи, а для сущности СПЕЦИАЛЬНОСТЬ связь неполная – может существовать специальность в общероссийском справочнике специальностей, по которой не проводится подготовка в вузах, занесенных в базу данных.

Отметим, что для представления связей между сущностями (ссылки на другую сущность) используется идентификатор соответствующей сущности (обозначение id) – специально вводимый код сущности, который однозначно идентифицирует объект.

Итоговая диаграмма «сущность – связь» приведена на рисунке 16.

Обозначения в диаграмме:

University – сущность ВУЗ – справочник вузов. Название вуза хранится в атрибуте «name», аббревиатура вуза – в атрибуте «brief».

Branch – сущность ОТДЕЛЕНИЕ, в которой хранится информация об отделении. Обычно выделяют три отделения – дневное, очно-заочное (вечернее), заочное; name – название отделения.

Faculty – сущность ФАКУЛЬТЕТ – справочник факультетов вуза; name – название факультета, brief – аббревиатура, univer_id – ссылка на вуз.

Program – сущность УЧЕБНАЯ ПРОГРАММА – учебные программы, по которым проводит обучение вуз. Данная сущность связана с другими сущностями – spec (специальности), branch (отделения), faculty (факультеты).

Spec – сущность СПЕЦИАЛЬНОСТЬ – справочник специальностей (общероссийский классификатор). Содержатся данные о названии специальности (name) и номере специальности (spec_number).

PK – первичный ключ, FKN – вторичный ключ с номером N.

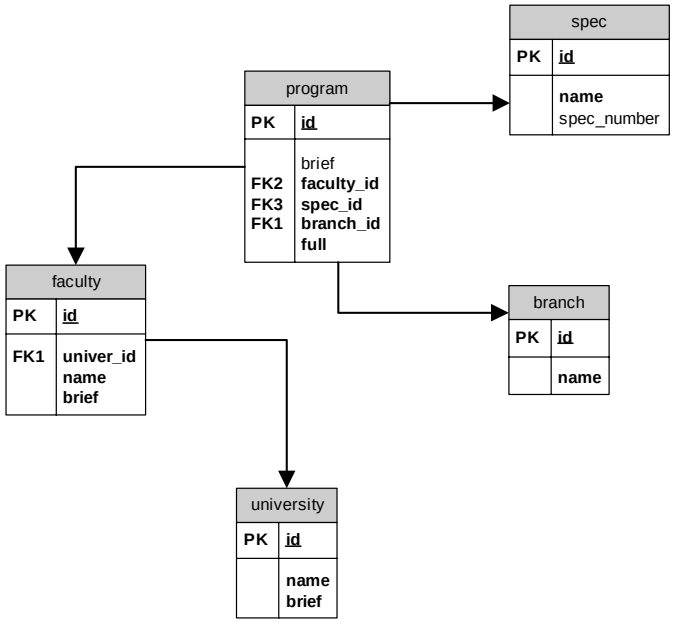


Рис. 16. Фрагмент ER-диаграммы

2.7. Построение концептуальной модели

Как уже отмечалось, концептуальная модель представляет собой обобщение представлений разных пользователей о данных. В связи с этим построение концептуальной модели, как правило, происходит в

два этапа. На первом этапе производится сбор и анализ характеристик данных и строятся так называемые модели локальных представлений (локальные модели). Чаще всего локальная модель отражает представление отдельного пользователя (отдельной функциональной задачи). Иногда такая модель может описывать и некоторую независимую область данных нескольких функциональных задач (нескольких приложений). Здесь необходимо отметить, что моделирование представлений отдельных пользователей приводит к снижению уровня интеграции данных, а моделирование совместных представлений группы пользователей – к повышению сложности проектирования. В связи с этим при выборе области данных для локального моделирования приходится выбирать компромиссное решение между вышеуказанными вариантами.

На втором этапе построенные локальные модели объединяются в обобщенную концептуальную модель.

2.7.1. Моделирование локальных представлений

Прежде всего, необходимо отметить, что построенная модель должна удовлетворять ряду требований:

- адекватно отражать представление пользователя о данных;
- давать возможность ответа на возможные запросы пользователя, причем делать это с минимальными затратами по количеству просматриваемых сущностей;
- представлять данные с минимальным дублированием.

Процесс построения модели, удовлетворяющей указанным требованиям, является творческим, и формализовать его, как правило, невозможно. Тем не менее можно указать некоторые способы порождения вариантов при моделировании. Выбор одного из таких вариантов на основе оценок объемов дублирования и числа просматриваемых объектов при ответах на запросы пользователей позволяет улучшить эксплуатационные характеристики проектируемой базы данных.

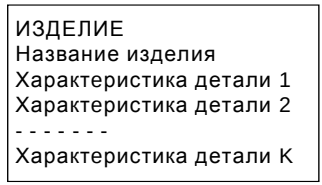
Вариативность моделирования обусловливается неоднозначностью выбора сущностей, атрибутов и связей. В одном варианте можно что-то взять за сущность, в другом варианте это же можно взять за атрибут (несколько атрибутов), в третьем варианте это можно определить как связь. Рассмотрим вышесказанное на простом примере.

Пример. Предметная область описывается следующим образом. Есть ряд изделий. Каждое изделие состоит из совокупности деталей. Возможны следующие запросы пользователей:

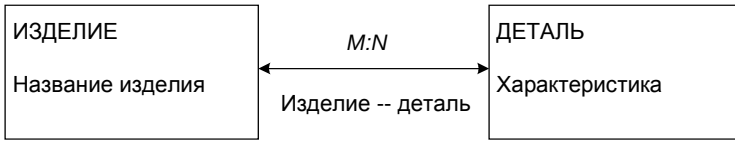
- Из каких деталей состоит конкретное изделие *B*?
- В какие изделия входит деталь *D*?

По-разному выбирая сущности, атрибуты и связи можно построить два варианта модели.

Вариант 1. Сущность – ИЗДЕЛИЕ. Атрибутами являются название изделия и характеристики деталей, его составляющих.



Вариант 2. Сущность – ИЗДЕЛИЕ. Сущность – ДЕТАЛЬ. Связь – изделие состоит из определенных деталей и деталь входит в определенные изделия.



Таким образом, даже для такой простейшей предметной области можно построить две модели.

Оценим число просматриваемых сущностей для каждого варианта при ответе на два вышеуказанных запроса пользователей. Для определенности предположим, что конкретный экземпляр сущности в наборе отыскивается посредством перебора (возможны и другие предположения).

Обозначим M – число изделий; N – число деталей; k_B – число деталей, входящих в изделие B, l_D – число изделий, в которые входит деталь D. Результаты представлены в следующей таблице.

Номер запроса	Вариант 1	Вариант 1
1	$\frac{1+M}{2}$	$\frac{1+M}{2}+k_B$
2	M	$\frac{1+N}{2}+l_D$

Поясним оценки, получаемые при запросе 1. Для ответа на запрос по модели первого варианта просматриваем экземпляры набора сущностей ИЗДЕЛИЕ и ищем экземпляр с названием *B*. В лучшем случае это может быть первый же экземпляр, в худшем – последний. Среднее число просматриваемых экземпляров будет равно $(1+M)/2$. Найдя нужный экземпляр, получаем его атрибуты. Ответ на запрос получен. Для ответа на запрос по модели второго варианта нужно найти экземпляр изделия с названием *B* (аналогично варианту 1), далее по связи перейти в набор сущностей ДЕТАЛЬ и там выбрать (по связи) детали, входящие в изделие *B* (выбрать k_B экземпляров). Общая оценка для этого случая будет $(1+M)/2+k_B$. Подсчет остальных оценок аналогичен. Сравним варианты.

Обозначим частоту i-го запроса q_i (i = 1, 2).

Тогда сравнение вариантов по вышеуказанным оценкам сводится к сравнению величин

$$q_1 (1+M)/2 + q_2 M$$

и

$$q_1\left[(1+M)/2 + k_B\right] + q_2\left[(1+N)/2 + l_D\right].$$

Меньшая из этих величин и определяет наилучший из представленных вариантов по оценке числа действий при ответах на запросы пользователей.

Нетрудно видеть, что в варианте 1 больше дублирования информации. Характеристика одной и той же детали может встречаться в нескольких изделиях. Рациональный выбор варианта основывается на компромиссе между оценками числа действий и оценками дублирования информации и зависит от конкретных соотношений этих оценок.

После того как выбран рациональный вариант локальной модели, производится редактирование введенных наименований сущностей, атрибутов и связей. Здесь выполняются следующие действия:

- устраняются расплывчатые наименования (все наименования должны однозначно пониматься каждым пользователем);
- устраняются синонимы (различные наименования одного и того же понятия);
- устраняются омонимы (одно и то же наименование разных понятий).

Эти действия, вообще говоря, носят итерационный характер, т.к. после их выполнения вновь могут возникать и расплывчатые наименования, и синонимы, и омонимы.

Завершающим шагом локального моделирования является указание ключей для каждого набора сущностей. Здесь необходимо указать как первичные, так и, по возможности, вторичные ключи.

2.7.2. Объединение локальных моделей

На этом этапе ранее построенные модели локальных представлений отдельных пользователей (или групп пользователей) объединяются в единую концептуальную модель. Объединение локальных моделей производится следующими путями:

- слияние идентичных элементов;
- установление связей между наборами сущностей разных моделей;
- введение новых агрегированных элементов для представления связей между элементами разных моделей;
- обобщение различных подобных типов сущностей, позволяющее трактовать эти сущности как одну обобщенную сущность.

Рассмотрим каждый из этих путей.

Слияние идентичных элементов

Два или более элементов модели идентичны, если они имеют одинаковое смысловое значение.

Рис. 17. Модели с идентичным элементом

Объединение моделей с идентичными элементами осуществляется путем «слияния» этих элементов в один.

Два набора сущностей СПЕЦИАЛЬНОСТЬ в модели 1 и 2 имеют одинаковое смысловое значение (рис. 17), и могут быть заменены одним набором сущностей (рис. 18).

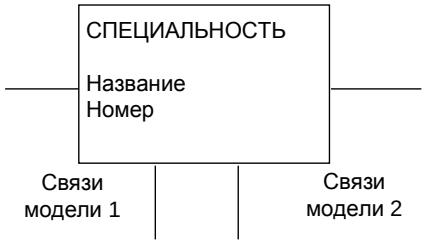


Рис.18. Объединенная модель

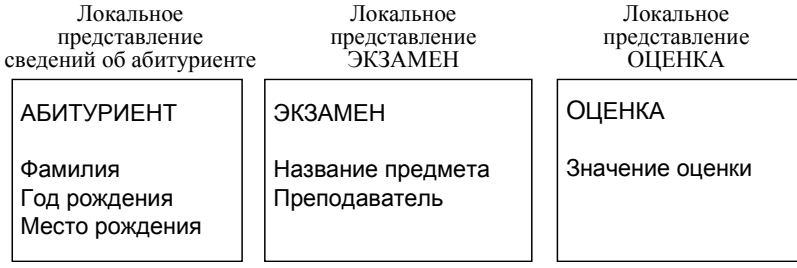
Установление связей между наборами сущностей разных моделей

При рассмотрении наборов сущностей объединяемых моделей необходимо выявление связей между ними, т.к. именно эти связи и определяют в конечном итоге интегрированную базу данных.

Введение агрегированных элементов

При объединении моделей связь между элементами разных моделей может рассматриваться как новый элемент.

Рассмотрим в качестве примера моделирование информационного представления сдачи абитуриентом вступительных экзаменов. Можно выделить ряд локальных представлений.



Оценка принимает значения «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Объединяя локальные представления, устанавливаем новые связи (рис. 19).

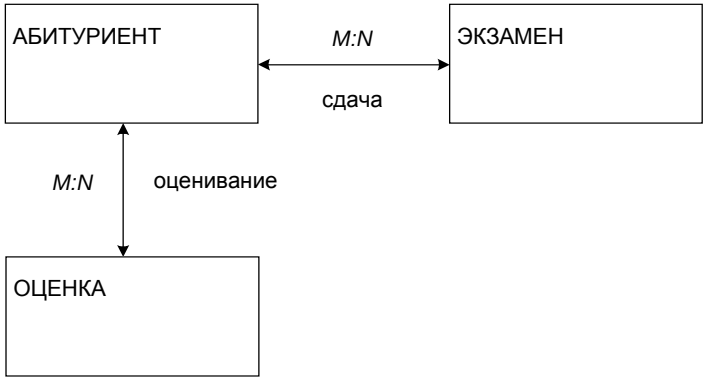
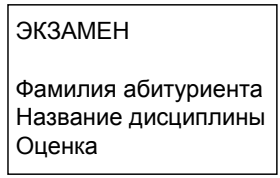


Рис.19. Объединение локальных представлений

Как уже отмечалось, одним из показателей «зрелости» модели является возможность ответа на запросы пользователей, и установление связей преследует именно эту цель. Нетрудно видеть, что какие бы связи в рассматриваемой модели ни вводились, невозможно ответить на запрос «какую оценку получил студент А по дисциплине В». В таком случае необходимо использовать принцип агрегации – необходимую связь между элементами модели ввести как некоторый новый элемент. В данном примере можно определить этот новый агрегированный элемент так:



Далее процесс объединения локальных моделей продолжается обычным образом.

Обобщение подобных типов сущностей

Рассмотрим ситуацию, когда каждый факультет формирует свое отделение приемной комиссии. Работники факультетской приемной комиссии учитывают данные об абитуриентах, поступающих на этот факультет, используя информацию о специальностях факультета.

В этом случае можно выделить несколько локальных моделей – модель факультета ВМК, модель экономического факультета и так далее. В локальную модель факультета ВМК входят сущности СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА ВМК и АБИТУРИЕНТЫ ФАКУЛЬТЕТА ВМК, в локальную модель экономического факультета входят, соответственно, сущности СПЕЦИАЛЬНОСТИ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА и АБИТУРИЕНТЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА.

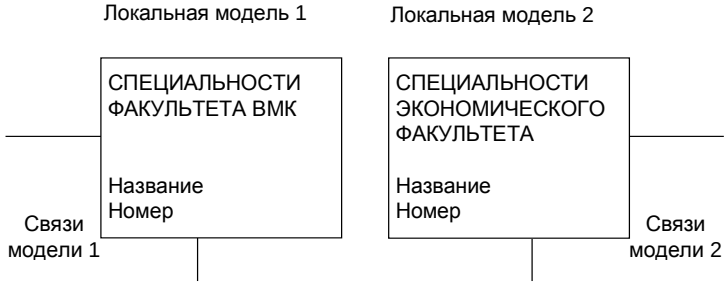


Рис. 20. Модели с подобным элементом

Два набора сущностей СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА ВМК и СПЕЦИАЛЬНОСТИ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА в моделях 1 и 2 имеют одинаковое смысловое значение и могут быть заменены одним набором сущностей с добавлением нового атрибута – факультет (рис. 21).

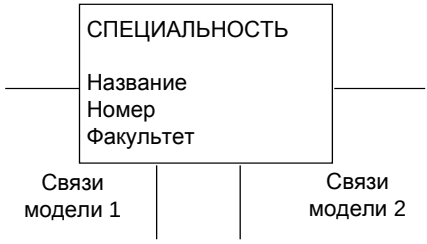


Рис. 21. Обобщенная модель

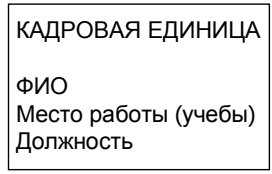
Отметим, что в данном случае подобным образом можно слить и все остальные сущности локальных моделей факультетов, так как

сущности АБИТУРИЕНТЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА и АБИТУРИЕНТЫ ВМК также имеют одинаковое смысловое значение. Однако в общем случае каждая локальная модель может содержать сущности и связи, которых нет в других локальных моделях.

Рассмотрим другой пример.

Предположим, что мы храним данные о студентах (фамилия, имя, отчество, курс, группа) и о преподавателях (фамилия, имя, отчество, кафедра, должность). Соответственно, в предметной области выделяем две сущности – СТУДЕНТ и ПРЕПОДАВАТЕЛЬ.

Эти разные сущности можно в некоторых случаях трактовать как подобные. Для обобщения соответствующих сущностей необходимо, прежде всего, обобщить их атрибуты. Заметим, что атрибуты «ФИО» у обеих сущностей совпадают, атрибуты «Кафедра» и «Курс», «Группа» показывают место работы (учебы) и их можно заменить обобщенным атрибутом «Место работы (учебы)». Атрибут «Должность» можно использовать и у сущности СТУДЕНТ, если в качестве значения соответствующего атрибута использовать значение «студент». Тогда две сущности ПРЕПОДАВАТЕЛЬ и СТУДЕНТ можно трактовать как подобные и заменить их на обобщенную сущность. Дадим этой обобщенной сущности название КАДРОВАЯ ЕДИНИЦА.



У студента атрибут «Место работы (учебы)» будет принимать значение «Курс. Группа», у преподавателя – название кафедры. Обобщенная модель представлена на рис. 22.

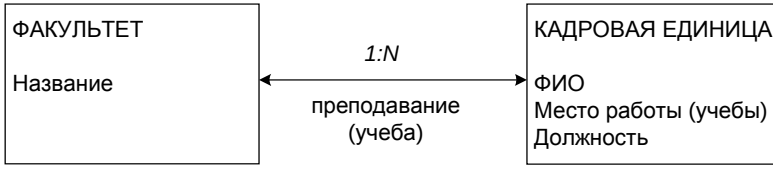


Рис. 22. Обобщенная модель

В этом случае почти в два раза упрощается структура концептуальной модели, и соответственно, структура базы данных. Для работы с данными о преподавателях и студентах достаточно одного набора программ. Таким образом, обобщение подобных типов объектов может существенно сократить последующие затраты на программирование.

В процессе объединения локальных представлений, как и при локальном моделировании, производится редактирование наименований (т.к. здесь появляются новые наименования). Процесс объединения также носит итерационный характер и продолжается до тех пор, пока не будут интегрированы все представления, согласованы и устранены все противоречия, отредактированы все наименования.

Полученное в результате объединения локальных представлений обобщенное представление и является концептуальной моделью.

2.8. Пример построения диаграммы «сущность – связь» для предметной области зачисления абитуриентов

В предметной области можно выделить следующие группы пользователей: сотрудники приемной комиссии и администраторы данных.

Рассмотрим первую группу – сотрудников приемной комиссии, которые обрабатывают информацию об абитуриентах.

Данная группа пользователей принимает документы абитуриентов (на какую специальность поступает абитуриент; на какие специальности, в случае непрохождения на основную, он хочет поступить; какие экзамены будет сдавать). Эти пользователи также заносят оценки, выставленные членами предметной комиссии в ведомости, проводят итоговое собеседование с абитуриентами, добавляя дополнительные баллы за наличие соответствующих документов, готовят приказы к зачислению, исходя из полученных оценок и дополнительных баллов.

Диаграмма «сущность – связь» для данной группы пользователей приведена на рисунке 23. Разберем подробно, как она получилась.

При рассмотрении разобьем модель на две части – одна часть касается принятия документов абитуриента и проведения экзаменов, вторая – зачисления.

Информация о том, какие экзамены будет сдавать абитуриент, составляет сущность ОЦЕНКИ (mark). Пока абитуриент не получил оценку, значение оценки в сущности ОЦЕНКА остается неопределенным.

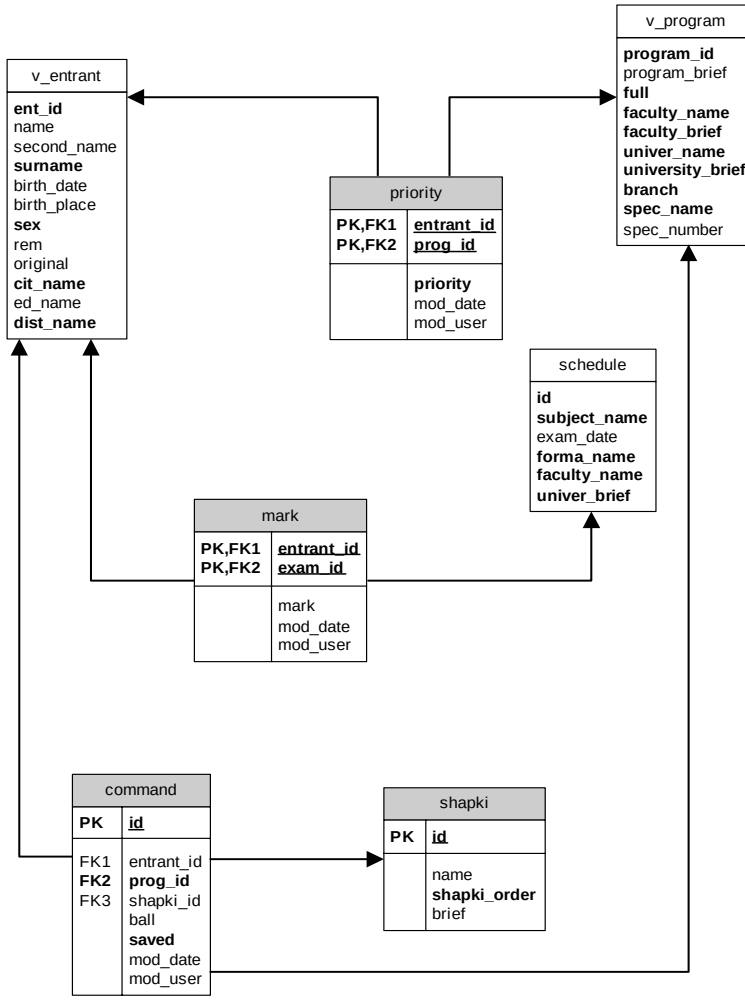


Рис. 23. Представление приемной комиссии

В сущности ОЦЕНКА (mark) выделяются следующие атрибуты.

– Оценка (mark) – значение оценки или неопределенное значение (null), если абитуриенту назначен экзамен, но он его еще не сдавал.

– Дата модификации экземпляра сущности (mod_date).
– Пользователь, который последним модифицировал экземпляр сущности (mod_user).
– Атрибуты entrant_id и exam_id служат для реализации связи данной сущности с сущностями АБИТУРИЕНТ и РАСПИСАНИЕ. Данные атрибуты являются внешними ключами, ссылающимися на соответствующие сущности.

Приоритеты абитуриентов составляют сущность ПРИОРИТЕТЫ (priority).

В сущности ПРИОРИТЕТЫ выделяются следующие атрибуты.

– Значение приоритета (priority).
– Дата модификации экземпляра сущности (mod_date).
– Пользователь, который последним модифицировал экземпляр сущности (mod_user).
– Атрибуты entrant_id и prog_id служат для реализации связи данной сущности с сущностями АБИТУРИЕНТ и СПЕЦИАЛЬНОСТЬ. Данные атрибуты являются внешними ключами, ссылающимися на соответствующие сущности.

Сущность АБИТУРИЕНТ (v_entrant) состоит из всех атрибутов, относящихся к абитуриенту. Надо заметить, что после некоторого анализа будет понятно, что в сущности АБИТУРИЕНТ, приведенной на рисунке 23, скрыто еще несколько сущностей. То же относится и к сущностям РАСПИСАНИЕ (schedule) и УЧЕБНАЯ ПРОГРАММА (v_program).

В сущности АБИТУРИЕНТ выделяются следующие атрибуты.

– Код абитуриента (ent_id).
– Имя абитуриента (name).
– Отчество абитуриента (second_name).
– Фамилия абитуриента (surname).
– Дата рождения (birth_date).
– Место рождения (birth_place).
– Пол (sex).
– Примечание (rem).
– Представлен подлинник документа (original). В данном атрибуте содержится информация о том, сдал ли абитуриент в приемную комиссию подлинник документа о предшествующем образовании. По современному законодательству абитуриент имеет право пода-

вать документы в несколько вузов одновременно, соответственно, сдавая не оригинал документа, а копию. Однако он допускается к зачислению только в одном вузе – в том, в который он принес подлинник. Если абитуриент пытается сдавать экзамены в несколько вузов, то после сдачи экзаменов, но до зачисления, он выбирает один и приносит туда подлинник. Вуз устанавливает некоторую пороговую дату, до которой абитуриент может принести подлинник документа. После этой даты вуз не принимает подлинник и не допускает абитуриента к участию в конкурсе.

– Гражданство (cit_name).
– Предшествующее образование (ed_name).
– Документ о медали или дипломе с отличием (dist_name).

Сущность РАСПИСАНИЕ (schedule) содержит информацию о том, какие экзамены на какие факультеты какого вуза назначены приемной комиссией.

Атрибуты сущности РАСПИСАНИЕ.

– Предмет (subject_name).
– Дата экзамена (exam_date).
– Форма проведения экзамена (forma_name).
– Факультет (faculty_name).
– Аббревиатура вуза (univer_brief).

Сущность СПЕЦИАЛЬНОСТЬ (v_program) содержит информацию по учебным программам, на которые могут поступать абитуриенты. Мы уже говорили о том, что учебная программа и специальность – это, на самом деле, разные сущности. Специальность определяется названием специальности и ее номером согласно общероссийскому классификатору. Учебная программа отличается уточнением факультета, вуза и отделения. С этой точки зрения логичнее было бы назвать сущность УЧЕБНАЯ ПРОГРАММА. Однако термин «специальность» является устоявшимся и понятен работникам приемной комиссии. В то же время термин «учебная программа» был введен искусственно и понятен администратору данных, но не пользователю. Поскольку мы в настоящий момент строим диаграмму «сущность – связь» для пользователя, то нужно, по возможности, сохранять терминологию, привычную для предметной области. Выделение из рассмотренной сейчас сущности СПЕЦИАЛЬНОСТЬ нескольких сущностей относится к задаче, решаемой при построении обобщенной диаграммы «сущность – связь».

Атрибуты сущности СПЕЦИАЛЬНОСТЬ.

– Краткое название учебной программы или, в терминах специалиста предметной области, краткое название специальности (program_brief).
– Полное название специальности, а если более правильно, то полное название учебной программы (full).
– Название факультета (faculty_name).
– Аббревиатура факультета (faculty_brief).
– Название вуза (univer_name).
– Аббревиатура вуза (university_brief).
– Отделение (branch).
– Название специальности (spec_name).
– Номер специальности (spec_number).

Другие части диаграммы будут пояснены ниже.

Сущности АБИТУРИЕНТ, РАСПИСАНИЕ, ОЦЕНКА, ПРИОРИТЕТ, СПЕЦИАЛЬНОСТЬ связаны между собой. Абитуриент получает оценку по конкретному экзамену. Соответственно, выделяются связи «абитуриент получает оценку», «оценка ставится за экзамен». Абитуриент формирует свои приоритеты на специальности (на учебные программы). Связи – «абитуриент определяет приоритет», «специальности соответствует приоритет».

Связи сформулированы неочевидно, поскольку являются надуманными. На самом деле надо было выделить две связи, каждая из которых связывает три сущности. Абитуриент получает оценку за экзамен (сущности АБИТУРИЕНТ, ЭКЗАМЕН, ОЦЕНКА). Абитуриент определяет приоритеты специальностей (сущности АБИТУРИЕНТ, СПЕЦИАЛЬНОСТЬ, ПРИОРИТЕТ). Причина выделения бинарных связей проста – только бинарные связи поддерживаются в реляционной модели, которая используется в большинстве современных СУБД.

Рассмотрим часть модели, касающуюся зачисления (рис. 24).

Сущности АБИТУРИЕНТ и СПЕЦИАЛЬНОСТЬ были описаны ранее.

Сущность ПУНКТЫ ПРИКАЗА (shapki) содержит информацию об основаниях, согласно которым зачисляются абитуриенты.

Атрибуты сущности ПУНКТЫ ПРИКАЗА.

– Содержание пункта (name).
– Краткое название пункта (brief).
– Номер пункта по порядку (shapki_order).

Сущность COMMAND содержит информацию о том, какие абитуриенты на какие специальности зачислены и согласно какому пункту приказа.

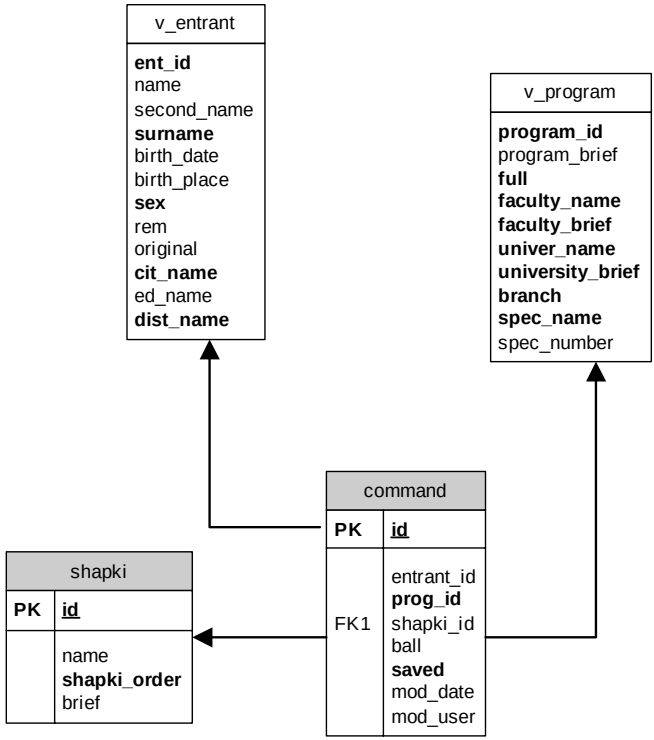


Рис. 24. Представление о зачислении (фрагмент рис. 23)

Помимо связующих атрибутов (entrant_id, prog_id, shapki_id), хранится информация о дате последнего изменения экземпляра сущности (mod_date) и пользователе, осуществившем последнее изменение (mod_user), а также информация о набранном балле (ball) и признак того, что информация запомнена и не нуждается в пересчете (saved).

Связь между сущностями формулируется так: «абитуриент зачислен на специальность в соответствии с пунктом приказа». Формулировать разбиение данной связи на бинарные мы не будем.

Вторая группа пользователей – администраторы данных.

Администраторы данных имеют доступ ко всей информации, представленной в базе данных. Они могут изменять структуру данных и заполнять справочники. Для всех остальных пользователей справочники являются таблицами на чтение, то есть можно просмотреть данную информацию, но нельзя изменить. Например, пользователь не может изменить расписания. Это действие может сделать только администратор данных. Локальное представление администраторов данных является одновременно и глобальным представлением, то есть полной концептуальной моделью.

Для нашего случая отличиями полной концептуальной модели от локальной модели сотрудников предметной комиссии будут представление сущностей РАСПИСАНИЕ, СПЕЦИАЛЬНОСТИ, АБИТУРИЕНТ в виде совокупностей нескольких сущностей, а также добавление сущности, необходимой для автоматического зачисления абитуриентов.

Разбиение сущности СПЕЦИАЛЬНОСТЬ на несколько сущностей было описано в пункте 2.6. «Выявление и моделирование сущностей и связей».

Сущность АБИТУРИЕНТ (рис. 25) разбивается на несколько сущностей в связи с тем, что при изменении базы данных может дублироваться информация о гражданстве, предшествующем образовании и о полученной медали (дипломе с отличием). Обоснование данного разбиения предоставляется читателю.

Рассмотрим сущность РАСПИСАНИЕ (рис. 26).

В приведенной на рисунке диаграмме показано не только разбиение сущности РАСПИСАНИЕ, но и связанные с ней сущности АБИТУРИЕНТ и ОЦЕНКА. Обоснование данного разбиения также предоставляется читателю, а формальное объяснение такого разбиения для случая реляционной модели приводится в соответствующем параграфе.

Помимо уже выделенных сущностей нам потребуется еще одна. Она будет использоваться при создании алгоритма автоматического зачисления. Это сущность ПРЕДВАРИТЕЛЬНЫЕ ПРИКАЗЫ (pre_command).

Необходимость создания этой сущности неочевидна исходя из тех данных, которые нам известны сейчас. Для понимания того, зачем нужна эта сущность, рассмотрим более подробно процедуру зачисления.

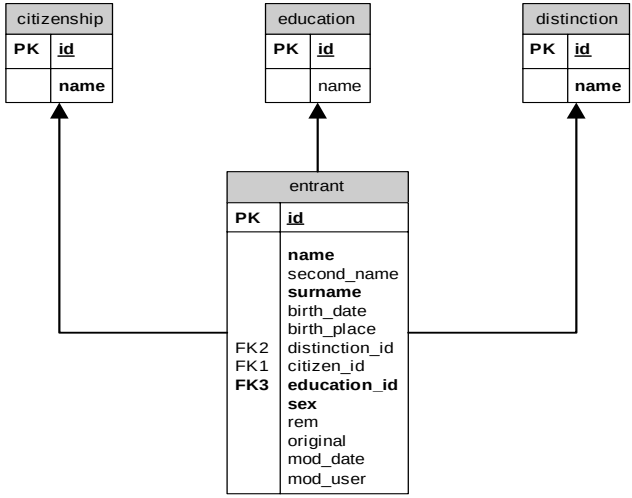


Рис. 25. Разбиение сущности АБИТУРИЕНТ

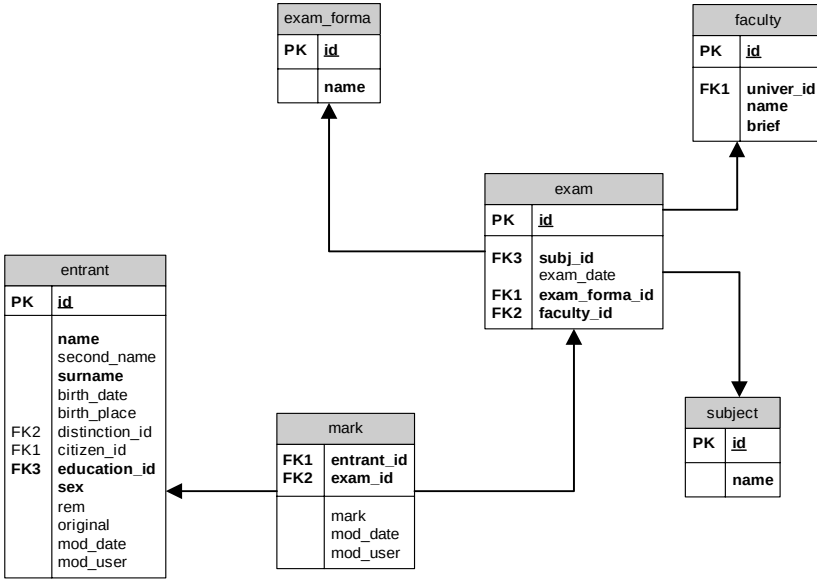


Рис. 26. Разбиение сущности РАСПИСАНИЕ

Итак, экзамены закончились и абитуриенты получили некоторые оценки. Общий проходной балл вычисляется так: количество баллов, набранное на экзаменах, умножается на два и к полученному числу добавляются десятые и сотые доли балла, исходя из документов, которые имеет абитуриент. Например, наличие медали или диплома с отличием приводит к добавлению 0,5 балла. Существуют и другие признаки, которые учитываются более сложным образом. Например, похвальная грамота по предмету засчитывается как 0,1 балла только в том случае, если этот (или аналогичный) предмет входит в перечень экзаменов. Если на ВМК ННГУ нужно сдавать математику, то похвальная грамота по алгебре или геометрии даст абитуриенту 0,1 балла.

Существуют и более сложные признаки, сформулировать которые в общем виде очень трудно. Все это приводит к тому, что решение о добавлении баллов принимает работник приемной комиссии. Абитуриенту предоставляется возможность оспорить данное решение. Таким образом, в сущности ПРЕДВАРИТЕЛЬНЫЕ ПРИКАЗЫ хранится информация об общем набранном абитуриентом балле на каждую специальность, на которую он претендует. Этот балл складывается из результатов экзаменов и добавленных «вручную» десятых и сотых долей балла.

Рассмотрим ситуацию, когда дополнительных баллов нет. Обосновано ли в таком случае создание сущности ПРЕДВАРИТЕЛЬНЫЕ ПРИКАЗЫ? Конечно, нет. Если пользователь не меняет балла, набранного на экзаменах, то вся информация, хранящаяся в данной сущности, является производной, то есть может быть получена путем сложного запроса к другим сущностям.

Атрибуты сущности ПРЕДВАРИТЕЛЬНЫЕ ПРИКАЗЫ.

- Связующие атрибуты (entrant_id, prog_id).
- Набранный абитуриентом балл (ball).
- Информация о том, кто последним модифицировал данный экземпляр сущности и когда это было сделано (mod_user, mod_date).
- Признак того, что экземпляр сущности проверен и не нуждается в автоматической обработке (saved). Данный атрибут требует пояснений. До проведения собеседования атрибут «набранный балл» является вычислимым, так как он формируется исходя из полученных абитуриентом оценок. Было бы слишком долго считать балл «вручную», поэтому в базе данных должна существовать

хранимая процедура, которая заполняет таблицу «предварительные приказы» начальными значениями. В качестве начального значения для атрибута «набранный балл» выбирается суммарная оценка на экзаменах, умноженная на два.

Предположим, что мы запустили данную процедуру, которая внесла информацию в сущность ПРЕДВАРИТЕЛЬНЫЕ ПРИКАЗЫ. Один из факультетов провел собеседование и изменил информацию по некоторым абитуриентам. В этот момент выяснилось, что некоторые оценки на другом факультете были внесены неверно. Мы должны или запустить процедуру заново, но тогда мы теряем результаты уже проведенного собеседования, или изменять результаты вручную, но тогда резко увеличивается вероятность ошибки.

Выходом в этой ситуации и является добавление атрибута «saved». Если этот атрибут установлен в «истинно», то данный экземпляр сущности не должен быть подвергнут изменению.

Итоговая диаграмма «сущность – связь» получается большая и приведена на рисунке 27 в сокращенном виде. Для воссоздания диаграммы полностью необходимо использовать фрагменты диаграмм, рассмотренные выше.

2.9. Ограничения целостности

Под целостностью базы данных понимается то, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область (правильная) информация.

Огромный объем данных, вводимых в базу данных – причем разные данные могут вводиться разными пользователями – обусловливает большое число ошибок ввода (занесения).

Заметим, что при традиционной «бумажной» обработке информации также достаточно часто встречаются данные, записанные неверно. Но человек, работая с определенными данными, неявно использует для контроля имеющиеся у него представления об этих данных. Например, сотрудник отдела кадров, увидев в карточке работника год рождения 1693, сразу заметит эту ошибку и предположит, что просто переставлены две цифры и реальный год рождения 1963. То есть в представлениях сотрудника заключены некоторые логические ограничения на данные. Очевидно, что для контроля правильности вводимых данных при работе с базой данных целесообразно сформировать и использовать ограничения.

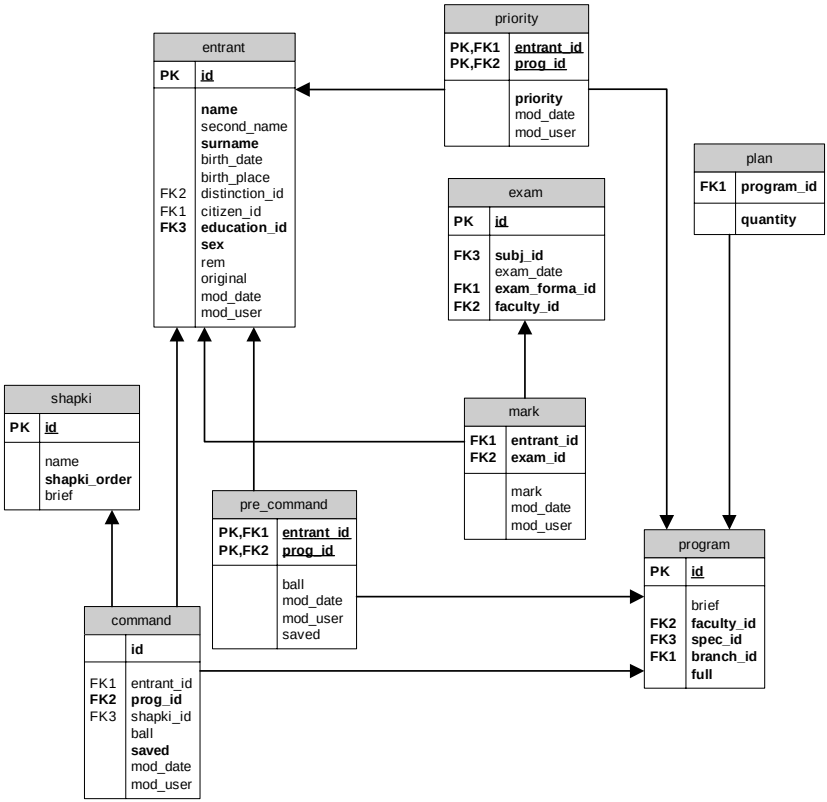


Рис. 27. Общий вид ER-диаграммы

Соответствующие ограничения можно разделить на две группы.

1. Внешние ограничения.

Эти ограничения связаны с адекватностью отражения предметной области. Например, сотрудник организации не может быть моложе 17 и старше 90 лет. Соответствующее ограничение на год рождения (GR) можно записать следующим образом:

Текущий год – 17 > GR > Текущий год – 90.

Одним из способов задания таких ограничений является перечисление конечного множества допустимых значений какого-либо атрибута (так называемый «перечислимый» тип данных).

Например, должность преподавателя в вузе может принимать одно из следующих значений: профессор, доцент, старший преподаватель, преподаватель, ассистент. Вводимое значение должности для конкретного экземпляра, не совпадающее с одним из перечисленных значений, является ошибкой.

2. Ограничения, описанные с помощью специальных конструкций.

Например, база демографических показателей содержит, в частности, атрибуты Число мужчин, Число женщин, Число лиц обоих полов (по разным возрастным группам, по разным регионам и т.п.). Для контроля достоверности вводимых данных можно использовать следующую формулу:

Число лиц обоих полов = Число мужчин + Число женщин.

Если равенство не достигается, какое-то из введенных чисел недостоверно.

Такие конструкции строятся исходя из специфики данных рассматриваемой предметной области. Можно построить много конструкций следующего вида: сумма значений по заданному атрибуту по всем экземплярам сущностей должна совпадать со значением определенного атрибута в экземпляре другой сущности.

Таким образом, на стадии ER-моделирования для повышения достоверности данных необходимо сформулировать соответствующие ограничения на данные. В идеальном случае каждое значение атрибута должно каким-то образом контролироваться. Использование этих ограничений позволяет существенно повысить достоверность данных в базе данных.

2.10. Средства автоматизированного проектирования концептуальной модели

Средства автоматизированного проектирования концептуальной модели привлекают к себе в настоящее время большой интерес и используются в процессе создания структуры базы данных и интерфейса пользователя для доступа к данным.

Причина применения этих средств состоит в использовании в подавляющем большинстве реальных разработок баз данных спиральной модели жизненного цикла программного обеспечения, что предусматривает последовательное создание нескольких версий программного обеспечения. Каждая следующая версия включает в себя предыдущую

(возможно, не полностью) и является ответом на замечания пользователя, полученные в результате тестирования предыдущей версии.

Напомним, что альтернативным способом является каскадная схема разработки программного обеспечения. Каскадный подход хорошо подходит для тех задач, для которых в самом начале разработки можно достаточно полно и точно сформулировать все требования заказчика. В случае построения баз данных каскадный подход является неприемлемым.

При создании баз данных первая модель программного обеспечения, к сожалению, очень редко является удачной. Чаще всего заказчик отвергает первую версию, так как она недостаточно полно отвечает его требованиям. Причина такой ситуации заключается в том, что заказчик не может сразу, до создания начальной версии программы, четко и полно сформулировать свои требования. Обычно после получения первого варианта программного обеспечения заказчик выдвигает дополнительные требования, которые нельзя реализовать в рамках созданной базы данных.

Это вынуждает разработчиков вносить изменения в структуру базы данных, а также, соответственно, в интерфейс пользователя для доступа к базе данных. Таких итераций может быть несколько до момента получения решения, адекватного запросам заказчика.

Но даже после получения удовлетворительного решения процесс разработки базы данных не завершается. Жизнь не стоит на месте, и запросы заказчика меняются с течением времени. Часть этих изменений можно реализовать без изменения структуры базы данных, изменяя только интерфейс пользователя, другие же требуют изменения и интерфейса, и структуры базы данных. Надо заметить, что подобные изменения являются очень болезненными – работа по их внесению может оказаться трудоемкой и, что самое неприятное, потребовать замены большого количества отлаженного программного кода. Иными словами, замененный код был написан впустую, на самом деле его не нужно было писать.

Таким образом, создание работоспособной базы данных можно условно разделить на три этапа – проектирование базы данных, в процессе которого создаются рабочие прототипы, кодирование – создание структур баз данных и законченного интерфейса пользователя и сопровождение готовой базы данных.

Основная идея применения средств автоматизированного проектирования баз данных заключается в том, что процесс ручного кодирова-

ния начинается только после окончания процесса проектирования. На стадии проектирования схема базы данных и интерфейс пользователя для доступа к базе данных создаются автоматически, исходя из описания концептуальной модели, с помощью так называемых CASE-средств (Computer Aided Software/System Engineering). Конечно, созданный таким образом интерфейс не является законченным программным продуктом, однако он позволяет заказчику оценить возможности конечного продукта и внести свои коррективы. Только после одобрения заказчиком рабочего прототипа разработчики приступают к ручному кодированию – созданию законченного приложения.

При сопровождении все повторяется, за тем исключением, что генерируется не все приложение целиком, а только часть, которую надо изменять.

На практике чаще всего CASE-средства используются для создания схемы базы данных в виде диаграмм «сущность – связь» и генерации структур баз данных для конкретной СУБД. После получения от заказчика изменений разработчики вносят соответствующие исправления в диаграмму «сущность – связь» и заново генерируют структуры баз данных. Средства автоматической генерации интерфейсов используются реже.

В настоящее время практически каждый производитель СУБД предлагает собственный программный продукт автоматизированного проектирования. Это Oracle Designer (Oracle), PowerDesinger (Sybase) и другие. Демонстрационные версии данных программных продуктов можно загрузить с соответствующих сайтов (www.oracle.com, www.sybase.com).

Кроме того, на рынке представлены решения третьих фирм, не производящих СУБД. Одними из самых распространенных являются программные продукты фирмы AllFusion – AllFusion ERwin Data Modeler и AllFusion Process Modeler (ранее – BPwin) и другие. На российском рынке данные программы предлагает фирма Interface Ltd. (www.interface.ru). Программа AllFusion Process Modeler предназначена для моделирования бизнес-процессов. Результатами ее работы могут быть не только диаграммы, но и сгенерированный для различных сред код для доступа к базам данных. Для этого, однако, необходимо еще создать диаграммы «сущность – связь» с помощью AllFusion ERwin Data Modeler.

Поскольку данный учебный курс не предполагает знакомства со средствами описания бизнес-процессов, мы рассмотрим только ERwin

Data Modeler – программный продукт, непосредственно использующийся при создании баз данных.

По данным Interface Ltd. AllFusion ERwin Data Modeler (ранее – ERwin) позволяет проектировать, документировать и сопровождать базы данных, хранилища данных и витрины данных (data marts).

Создав наглядную модель базы данных можно оптимизировать структуру БД и добиться её полного соответствия требованиям и задачам организации. Визуальное моделирование повышает качество создаваемой базы данных, продуктивность и скорость её разработки.

На сайте Interface Ltd. доступна для загрузки демонстрационная версия AllFusion ERwin Data Modeler, которая представляет собой полнофункциональную версию, ограниченную по времени.

Основные характеристики AllFusion ERwin Data Modeler:

- Поддержка различных способов записи диаграмм «сущность – связь» (нотаций).

Нотация диаграмм «сущность – связь» в AllFusion ERwin Data Modeler несколько отличается от нотации классических моделей «сущность – связь».

Эти отличия проявляются, прежде всего, в отсутствии атрибутов у связи. Наличие атрибутов у связи при классическом способе записи свидетельствует, что внутри связи скрыта некоторая сущность. В нотациях, используемых в AllFusion ERwin Data Modeler, связь не может иметь атрибутов и изображается в виде линии, а ее название пишется выше и/или ниже линии.

Атрибуты внутри сущности в AllFusion ERwin Data Modeler разделяются на две группы линией. Ключевые атрибуты находятся сверху черты, остальные атрибуты снизу.

Сильная сущность обозначается прямоугольником, слабая сущность – прямоугольником с закругленными углами. Идентифицирующая связь – сплошная линия, неидентифицирующая – прерывистая.

- Поддержка проектирования информационных хранилищ.
- Поддержка совместного проектирования.
- Поддержка триггеров, хранимых процедур и шаблонов.
- Развитые средства проверки корректности моделей данных Reverse Engineering (генерация модели данных на основе анализа существующей базы данных), включая восстановление связей по индексам.
- Автоматическая генерация SQL DDL для создания баз данных.
- Полная совместимость и поддержка двадцати типов СУБД.

Приведем в качестве примера использования данного CASE-средства диаграмму «сущность – связь» фрагмента базы данных абитуриентов. В сущности entrant сохраняются анкетные данные об абитуриентах (entrant). Помимо этой сущности на диаграмме приведены сущности-справочники: виды предшествующего образования (education), данные о гражданстве (citizenship), а также данные о наличии у абитуриента золотой или серебряной медали или диплома с отличием (distinction). Сущности связаны между собой связью «имеет». Абитуриент имеет предшествующее образование, гражданство и данные об отличии диплома или аттестата.

На рисунках 28 и 29 приведены диаграммы «сущность – связь» на логическом и физическом уровнях. Разделение на уровни используется в ERwin для удобства – на самом деле диаграмма одна, однако в зависимости от потребностей разработчика она может быть представлена в двух видах. Необходимо отметить, однако, что можно создавать элементы, которые присутствуют только на логическом или только на физическом уровне.

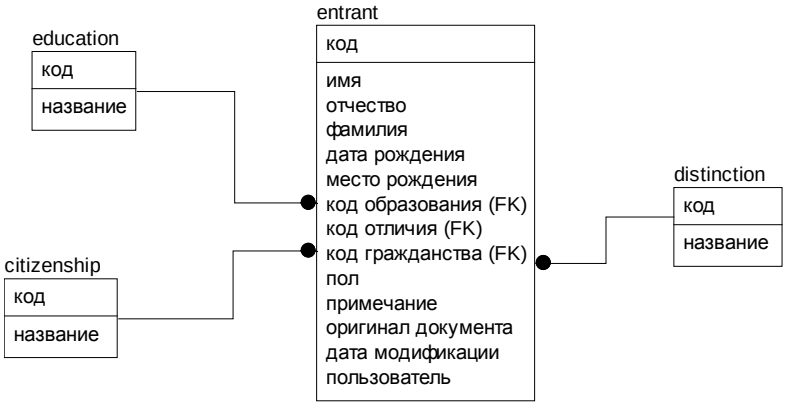


Рис. 28. Логический уровень ER-диаграммы

Логический уровень позволяет сконцентрироваться на описании бизнес-правил, а физический – на представлении модели в конкретной СУБД. В частности, названия полей могут отличаться на логическом и физическом уровнях. В нашем примере в СУБД используются английские названия полей, а на логическом уровне мы определяем русские названия.

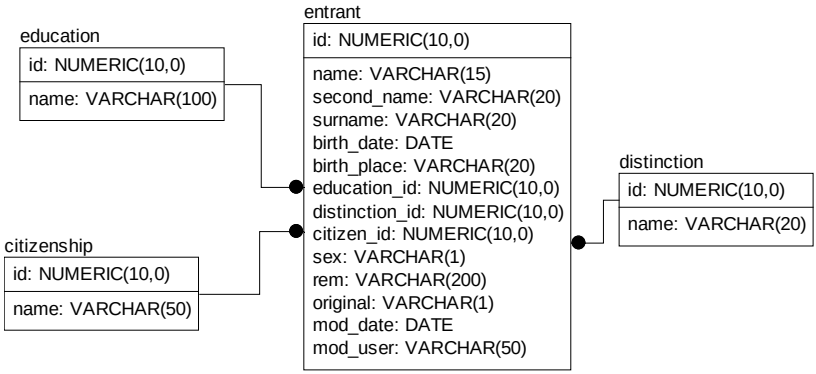


Рис. 29. Физический уровень ER-диаграммы

На физическом уровне диаграмма будет выглядеть по-разному в зависимости от того, для какого сервера баз данных будет использоваться данная модель и, соответственно, какие типы данных есть в данной СУБД. Приведенная на рисунке 29 диаграмма построена для СУБД Oracle 8.0.

После создания модели мы можем сгенерировать с помощью CASE-средства соответствующую структуру баз данных. Генерация осуществляется с помощью выполнения автоматически создаваемого скрипта. Настройки, задаваемые при генерации, влияют на полученный скрипт.

Фрагмент скрипта для создания структур баз данных приведен ниже.

```
…
CREATE TABLE education (
       name                  VARCHAR(100) NULL,
       id                    NUMERIC(10,0) NOT NULL,
       PRIMARY KEY (id)
);
CREATE TABLE distinction (
       name                  VARCHAR(20) NOT NULL,
       id                    NUMERIC(10,0) NOT NULL,
       PRIMARY KEY (id)
);
CREATE TABLE citizenship (
```

```
        name                    VARCHAR(50) NOT NULL,
        id                      NUMERIC(10,0) NOT NULL,
        PRIMARY KEY (id)
);
CREATE TABLE entrant (
        name                    VARCHAR(15) NOT NULL,
        id                      NUMERIC(10,0) NOT NULL,
        second_name             VARCHAR(20) NULL,
        surname                 VARCHAR(20) NOT NULL,
        birth_date              DATE NULL,
        birth_place             VARCHAR(20) NULL,
        education_id            NUMERIC(10,0) NOT NULL,
        distinction_id          NUMERIC(10,0) NOT NULL,
        citizen_id              NUMERIC(10,0) NOT NULL,
        sex                     VARCHAR(1) NOT NULL,
        rem                     VARCHAR(200) NULL,
        original                VARCHAR(1) NOT NULL,
        mod_date                DATE NULL,
        mod_user                VARCHAR(50) NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (education_id)
                                REFERENCES education,
        FOREIGN KEY (distinction_id)
                                REFERENCES distinction,
        FOREIGN KEY (citizen_id)
                                REFERENCES citizenship
);
…
```

Окончание скрипта не приведено.

CASE-средство ERwin Data Modeler в совокупности с AllFusion Process Modeler и другими продуктами фирмы AllFusion предоставляет стандартные возможности, характерные и для других CASE-средств, таких, как Oracle Designer и PowerDesigner (Sybase).