Тема 11. Реляционная модель данных

была предложена известным исследователем в области баз данных в 1969 году и представляет собой хранилище данных, содержащее *совокупность двухмерных таблиц.*
Любая таблица реляционной базы данных (РБД) состоит из *строк и столбцов* (называемых также *записями и полями).* Строки таблицы содержат сведения о представленных в ней фактах (или документах, или людях, одним словом, - об однотипных объектах). На пересечении поля и записи находятся *конкретные значения* содержащихся в таблице данных.
Наличие постоянных и разовых пользователей в автоматизированных информационных системах (АИС) и, следовательно, наличие потока регламентированных и произвольных по содержанию запросов требуют разработки специальных подходов к определению *границ предметной области* (ПрО) и проектированию *состава элементов* информационной модели.

Рассмотрим основные подходы к созданию инфологической модели предметной области.

Прикладными базами данных называются БД, которые могут объединять все данные, необходимые для решения одной или нескольких прикладных задач, спроектированные по подходу «от запросов пользователей».

Функциональный подход. Этот метод реализует принцип «*от запросов пользователей»* и применяется тогда, когда существует только поток регламентированных запросов, не ожидается развитие системы, и обслуживаются информационные потребности некоторой группы лиц и/или комплекса задач, для которых создается рассматриваемая БД.
Предметный подход. Предметный подход или подход *«от реального мира»,* предполагает наличие потока произвольных по содержанию запросов и развитие АИС во времени, позволяет выполнить прогноз смыслового содержания ожидаемой совокупности произвольных запросов, и у разработчиков есть чёткое представление о самой ПрО и о том, какую именно информацию они хотели бы хранить в БД. Этот подход базируется на предположении, что произвольные запросы пользователей соответствуют тематической направленности АИС.

Предметными базами данных называются БД, которые объединяют данные относящиеся к какой – либо предметной области (например, финансам, обучению, торговле и т. п.) и соотносящимся с предметами организации, а не с её информационными приложениями.

Предметные БД позволяют *обеспечивать поддержку любых текущих и будущих приложений*, поскольку набор их элементов данных включает в себя наборы элементов данных прикладных БД. Вследствие этого, предметные БД *создают основу для обработки неформализованных, изменяющихся и неизвестных запросов и приложений* (приложений, для которых невозможно заранее определить требования к данным). Такая гибкость и приспосабливаемость позволяет создавать на основе предметных БД достаточно стабильные АИС, то есть системы, в которых большинство изменений можно осуществить без вынужденного переписывания старых приложений.

Прикладное проектирование позволяет существенно ускорить создание высокоэффективных АИС, структура которых учитывает наиболее часто встречаемые пути доступа к данным, поэтому оно до сих пор привлекает некоторых разработчиков. Однако, по мере роста числа приложений таких АИС быстро увеличивается число прикладных БД, резко возрастает уровень дублирования данных и повышается стоимость их ведения.

Подход «от реального мира» предпочтительно использовать в качестве основного, подхода «от запросов пользователей» - для уточнения границ предметной области.

Таким образом, каждый из рассмотренных подходов к проектированию воздействует на результаты проектирования в разных направлениях.

Проектирование с использованием метода «сущность – связь». Желание достичь и гибкости, и эффективность привело к формированию методологии проектирования, использующей как *предметный*, так и *прикладной* подходы.

В общем случае **предметный** подход используется для построения первоначальной информационной структуры, а **прикладной** для ее совершенствования с целью повышения эффективности обработки данных.

Метод "Сущность - связь" (entity - relation, ER - method) является комбинацией предметного и прикладного методов и обладает достоинствами обоих.

Основная цель проектирования БД — это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в

разных местах сведений об одном и том же объекте. Так называемый «чистый» проект БД — «Каждый факт в одном месте».

При проектировании БД решаются две основных проблемы.

1. Каким образом отобразить объекты предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было, по возможности, лучшим (эффективным, удобным и т. д.)? Часто эту проблему называют проблемой *логического* проектирования баз данных.
2. Как обеспечить эффективность выполнения запросов к базе данных, то есть каким образом, имея в виду особенности конкретной системы управления базой данных (СУБД), расположить данные во внешней памяти, создание каких дополнительных структур (например, индексов) потребовать и т. д.? Эту проблему называют проблемой *физического* проектирования баз данных.

ЭТАПЫ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

Процесс проектирования включает в себя следующие этапы.

1. Инфологическое проектирование.

2. Определение требований к операционной обстановке, в которой будет функционировать информационная система.

3. Выбор СУБД и других инструментальных программных средств.

4. Логическое проектирование БД

5. Физическое проектирование БД.

1. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Задача инфологического моделирования базы данных — получение семантических (смысловых) моделей, отражающих информационное содержание конкретной ПрО. На этом этапе выполняется восприятие реальной действительности, абстрагирование, изучение и описание предметной области.
Внешний уровень может включать несколько различных представлений БД со стороны различных групп пользователей. При этом каждый пользователь имеет дело с представлением предметной области, выраженным в наиболее

понятной и удобной для него форме. Такое представление содержит только те *сущности (объекты), атрибуты* и *связи,* которые интересны ему при решении профессиональных задач.
Различные представления на внешнем уровне могут пересекаться, то есть, использовать общие описания абстракций предметной области.
На внешнем уровне создастся *инфологическая модель* БД (внешняя схема), полностью независимая от платформы (т. е. вычислительной системы, на которой будет использоваться).

Инфологическая модель является человеко-ориентированной: средой ее хранения может быть память человека, а не ЭВМ. Инфологическая модель не изменяется до тех пор, пока какие-то изменения в реальном мире не потребуют изменения в ней некоторого определения, чтобы эта модель продолжала отражать предметную область.

Анализ предметной области начинается с выделения из воспринимаемой реальности ПрО, определяются ее границы, происходит абстрагирование от несущественных частей для данного конкретного применения базы данных.

В результате этих действий определяются объекты, их свойства и связи, которые будут существенны для будущих пользователей системы.

После этого изучается предметная область, накапливаются знания о ней. Эти знания представляются в какой-либо языковой системе, обычно это неформализованное описание с использованием естественного языка, математических формул, диаграмм, связей и т. д.

Выполняется структуризация знаний о предметной области: выделяются и классифицируются множества составляющих ПрО, стандартиз-уется терминология.

Таким образом, инфологическая модель ПрО представляет собой *описание* структуры и динамики ПрО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПрО и связей между ними, а их *типов,* связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.
Проектировщик разбивает ПрО на ряд *локальных* объектов, каждый из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи).

Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштаба ПрО. Обычно она разбивается на локальные объекты таким образом, чтобы каждый из них соответствовал отдельному внешнему приложению и содержал 6-7 сущностей.

Сущность - это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие *(например, СОТРУДНИК или АВТОМОБИЛЬ),* так и абстрактные (*например, ЭКЗАМЕН или ДИАГНОЗ).*

Для каждой сущности выбираются свойства (атрибуты). Различают следующие атрибуты:

1. ***Идентифицирующие и описательные атрибуты*** Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются *потенциальными ключами*. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается о*дин* первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются *описательными* и заключают в себе интересующие свойства сущности.
2. ***Составные и простые атрибуты***. *Простой* атрибут состоит из одного компонента, его значение неделимо. *Составной* атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных *(например, ФИО или адрес).* Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
3. ***Однозначные и многозначные атрибуты*** (могут иметь соответственно одно или много значений дня каждого экземпляра сущности).
4. ***Основные и производные атрибуты***. Значение *основного* атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты).
Спецификация **атрибута** состоит из его *названия*, указания *типа данных и описания ограничения целостности* - множества значений (или домена), которые может принимать данный атрибут.

Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику). Различают связи типа «сущность-сущность», «сущность-

атрибут» и «атрибут-атрибут» для отношений между атрибутами, которые характеризуют одну и ту же сущность или одну и ту же связь типа «сущность-сущность».

Каждая **связь** характеризуется *именем, обязательностью, типом* и *степенью.* Различают **факультативные** и **обязательные** связи. Если вновь порожденный объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является **факультативной**.

По **типу** различают множественные связи «один к одному» (1:1), «один ко многим» (1:N) и «многие ко многим» (M:N). ER-диаграмма, содержащая различные типы связей, приведена на рис. 1. Обратите внимание, что обязательные связи на рис. 1 выделены двойной линией.

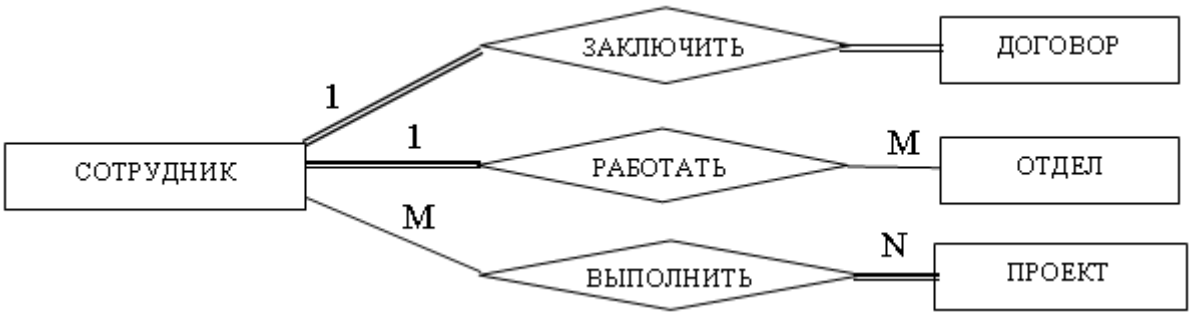


Рис1. ER-диаграмма с примерами типов множественных связей

После того, как созданы локальные представления (рис 2), выполняется их объединение. При небольшом количестве локальных объектов (не более пяти) они объединяются за один шаг. В противном случае обычно выполняют *бинарное* объединение в несколько этапов.

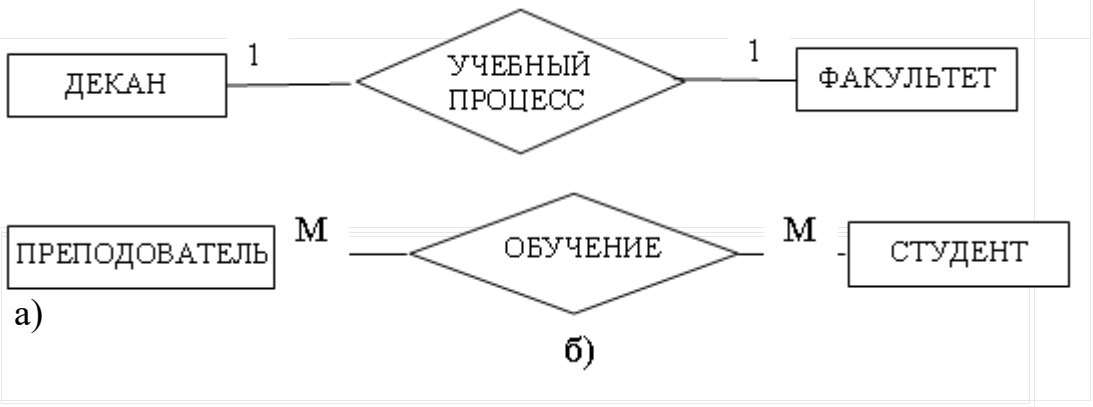


Рис.2. Локальные представления предметной области «Факультет»

При объединении проектировщик может формировать конструкции, производные по отношению к тем, которые были использованы в локальных представлениях. Такой подход может преследовать следующие цепи:

• объединение в единое целое фрагментарных представлений о различных свойствах одного и того же объекта;

• введение абстрактных понятий, удобных для решения задач системы, установление их связи с конкретными понятиями, использованными в модели;

• образование классов и подклассов подобных объектов (например, класс «изделие» и подклассы типов изделий, производимых на предприятии).

На этапе объединения необходимо выявить и устранить все противоречия. *Например, одинаковые названия семантически различных объектов или связей или несогласованные ограничения целостности на одни и те же атрибуты в разных приложениях.*
Устранение противоречий вызывает необходимость возврата к этапу моделирования локальных представлений с целью внесения в них соответствующих изменений.
Модели локальных представлений, то есть представление БД с точки зрения *конкретных* пользователей - это *внешние инфологические модели.*

По завершении объединения результаты проектирования являют собой концептуальную инфологическую модель предметной области.

Концептуальный уровень - обобщающее представление базы данных, описывающие то, какие данные хранятся в БД, а также связи, существующие между ними.
Концептуальный уровень является промежуточным в трехуровневой архитектуре. Содержит логическую структуру всей базы данных. Фактически, это полное представление требований к данным с стороны организации, которое не зависит от соображений относительно *способа их хранения*. На концептуальном уровне необходимо выделить:

§ сущности, их атрибуты и связи;

§ ограничения, накладываемые на данные;

§ семантическую информацию о данных;

§ информацию о мерах обеспечения безопасности.

Концептуальный уровень поддерживает каждое внешнее представление. Поэтому на этом уровне содержатся любые доступные пользователю данные, за исключением сведений о методах хранениях этих данных.
На концептуальном уровне создается *даталогическая модель* (концептуальная схема БД), представляющая собой описание инфологической модели (внешней схемы) на языке определения данных конкретной СУБД. Это модель являться компьютеро - ориентированной (зависит от применяемой на компьютере СУБД).

1.1 Проблемы ER-моделирования

В процессе создания инфологической модели на языке ER-диаграмм, могут возникать нежелательные ситуации, которые в литературе называются *ловушками соединения.* Причины этих проблем кроются в неправильной интерпретации семантики предметной области, в том числе смысла некоторых связей между выделенными сущностями. Очень важно своевременно выявлять в модели данных ловушки соединения, иначе они могут привести к неадекватному описанию предметной области и необходимости перестройки всей концептуальной модели.
Наиболее распространенными являются два вида ловушек соединения: *ловушки разветвления* и *ловушки разр*ыва.
Ловушка разветвления имеет место в том случае, если модель отображает связь между сущностями, но путь между отдельными экземплярами этих сущностей *однозначно* не определяется.

Ловушка разветвления возникает в случае, когда две или больше связей «один-ко-многим» разветвляются из одной сущности. На рис. 3 показан пример потенциальной ловушки разветвления, где две связи типа 1:M выходят из одной и той же сущности ФАКУЛЬТЕТ. Проанализировав модель, можно сделать вывод, что на одном факультете осуществляется обучение по нескольким специальностям и на факультете учится множество студентов. Проблема может возникнуть при попытке выяснить, по какой специальности обучается каждый из студентов факультета.

Рис. 3 Пример ловушки разветвления

Для обнаружения этой проблемы удобно пользоваться *семантическими сетями* (рис 4). С помощью семантической сетевой модели на конкретном примере невозможно дать однозначный ответ на вопрос: «По какой специальности обучается студент Уткин?» - это ловушка разветвления. Эта неприятность произошла из-за неправильной трактовки связей между сущностями ФАКУЛЬТЕТ, СПЕЦИАЛЬНОСТЬ, СТУДЕНТ. Устранить такой дефект можно только путем перестройки исходной модели.

Результат адекватного преобразования модели представлен на рис. 5. В таком варианте легко определяется, что студент Уткин учится на инженерно-строительном факультете по специальности «Управление и информатика в технических системах».

Если проверить полученную структуру на уровне отдельных экземпляров сущностей (рис. 6), можно убедиться, что по преобразованной модели легко дать однозначный ответ на поставленный выше вопрос.

Экземпляры Экземпляры Экземпляры Экземпляры Экземпляры

сущности связи сущности связи сущности

СТУДЕНТ ОБУЧЕНИЕ ФАКУЛЬТЕТ ПРОФИЛЬ СПЕЦИАЛЬНОСТЬ

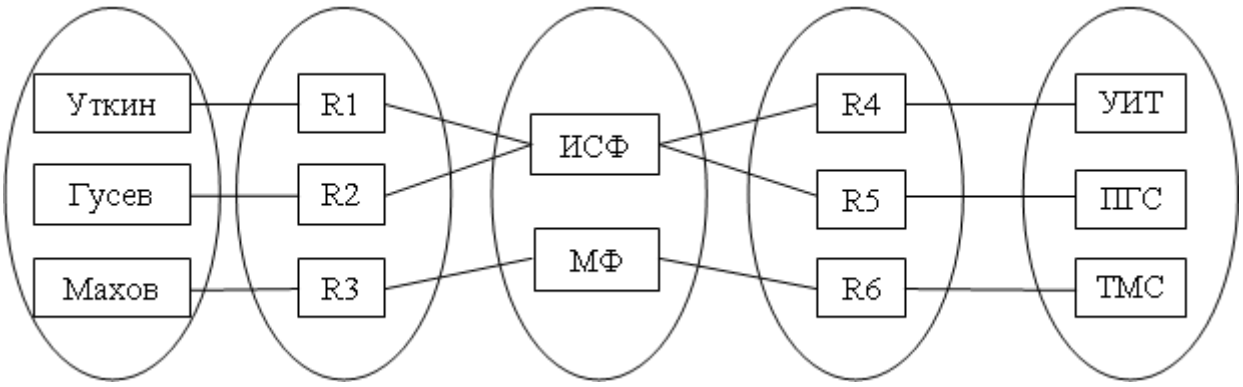
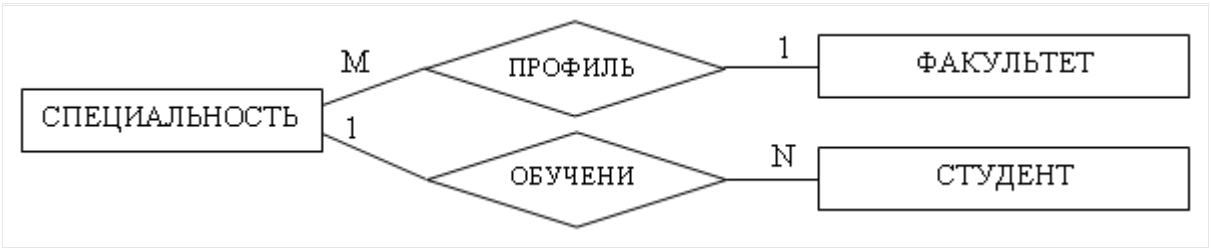


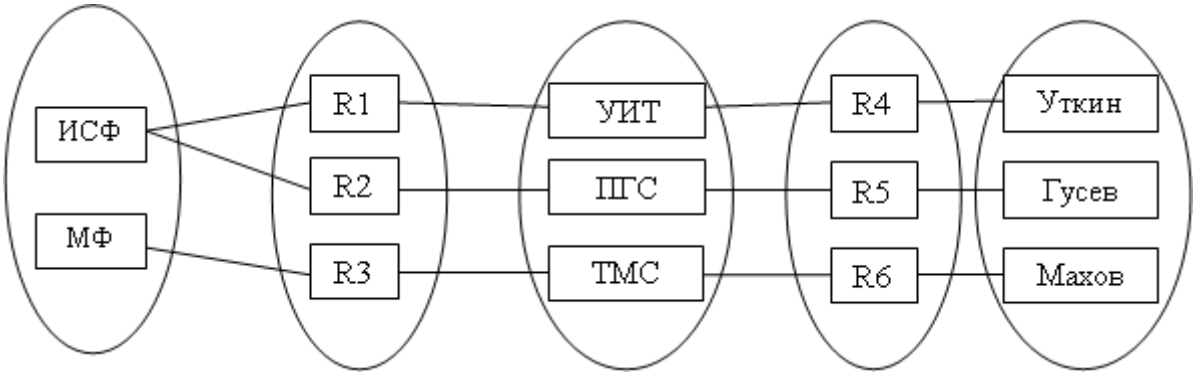
Рис. 4 Семантическая сеть ER – модели с ловушкой разветвления

Рис. 5 Преобразование ER – модель



Экземпляры Экземпляры Экземпляры Экземпляры Экземпляры

сущности связи сущности связи сущности



ФАКУЛЬТЕТ ПРОФИЛЬ СПЕЦИАЛЬНОСТЬ ОБУЧЕНИЕ СТУДЕНТ

Рис 6 Семантическая есть преобразованной ER – модели

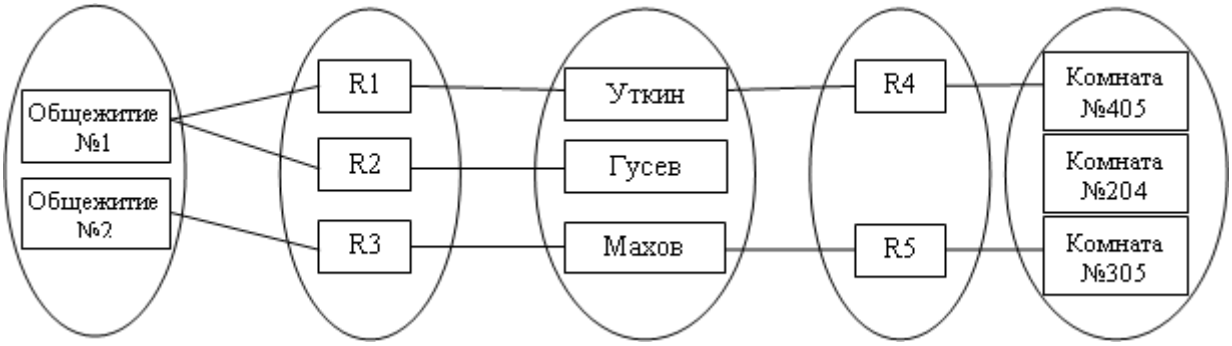
Ловушка разрыва появляется в том случае, если в модели предполагается наличие связи между сущностями, но не существует пути между отдельными экземплярами этих сущностей.

Ловушка разрыва возникает при наличии связи, образующей часть пути между связанными сущностями. На рис 7 потенциальная ловушка разрыва показана на примере связей между сущностями ОБЩЕЖИТИЕ, СТУДЕНТ и КОМНАТА (семантическая сеть ER-модели - на рис. 8).

Рис.7 Пример ловушки разрыва

Экземпляры Экземпляры Экземпляры Экземпляры Экземпляры

сущности связи сущности связи сущности



ОБЩЕЖИТИЕ РЕГИСТРАЦИЯ СТУДЕНТ ПРОЖИВАНИЕ КОМНАТА

Рис. 8 Семантическая сеть ER - модели с ловушкой разрыва

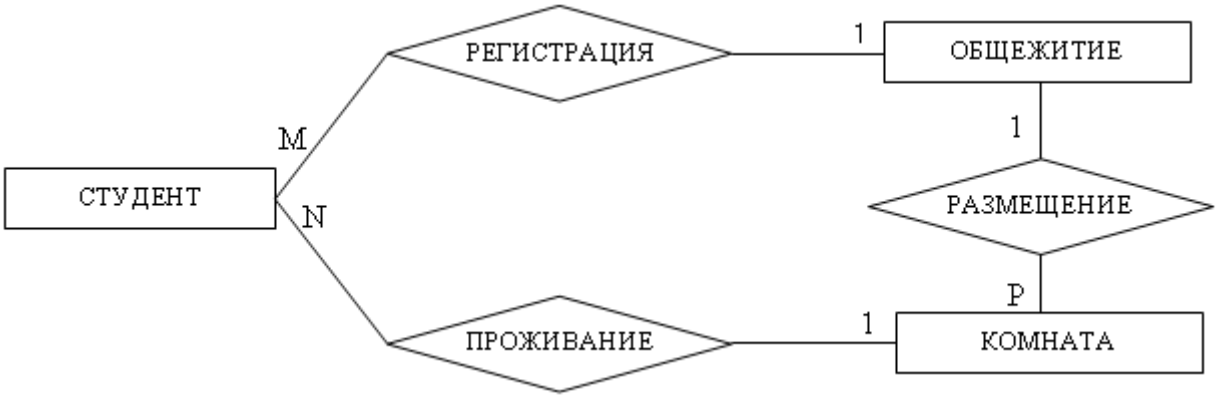


Рис.9 Преобразованная ER - модель

Устранить эту проблему можно только путем перестройки ER-модели для правильного взаимоотношения между этими сущностями (рис. 9). В модель добавлена связь Размещение между сущностями ОБЩЕЖИТИЕ и КОМНАТА.

Экземпляры Экземпляры Экземпляры Экземпляры Экземпляры

сущности связи сущности связи сущности

ОБЩЕЖИТИЕ РЕГИСТРАЦИЯ СТУДЕНТ ПРОЖИВАНИЕ КОМНАТА

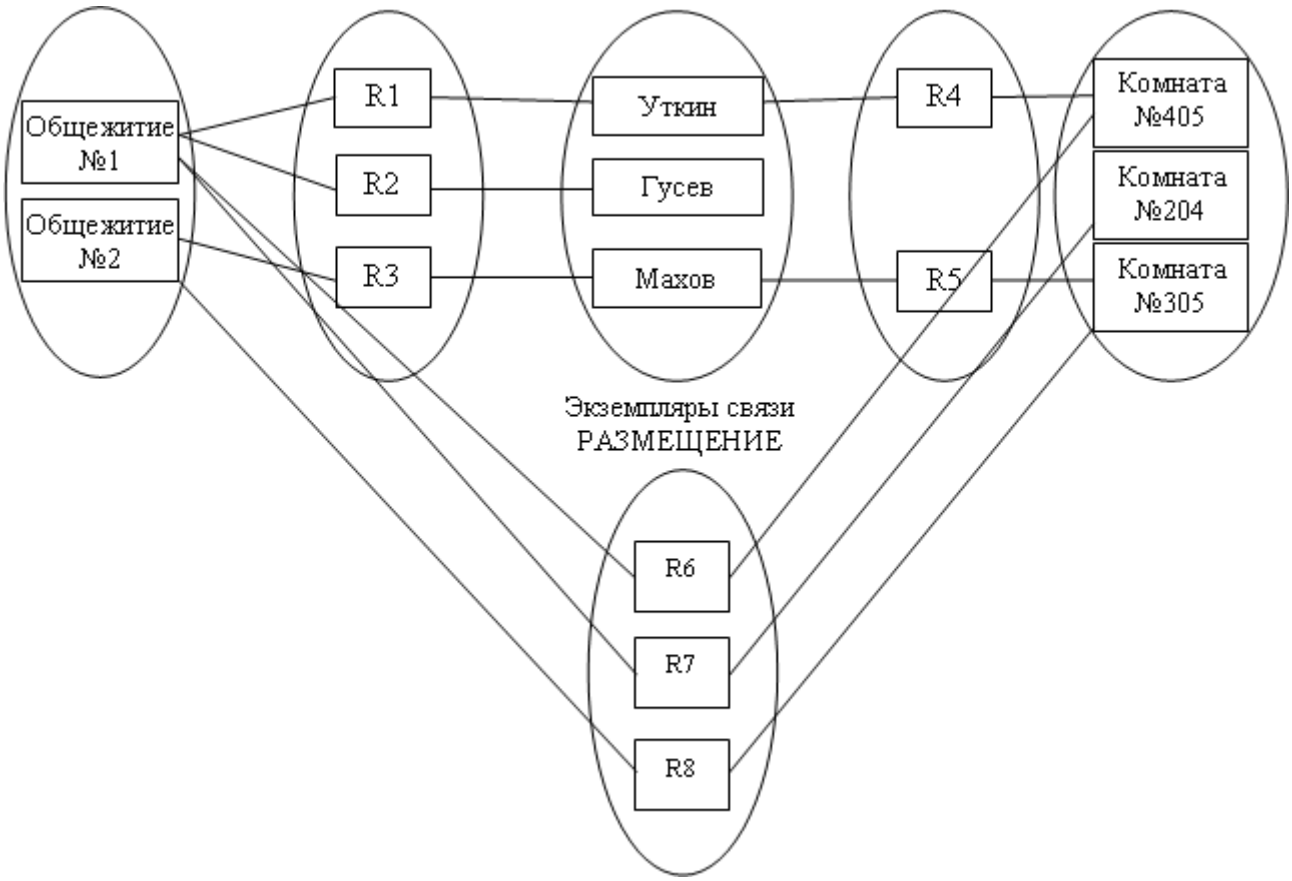


Рис. 10 Семантическая сеть преобразованной ER - модели

Если исследовать преобразованную ER - модель на уровне отдельных сущностей (как показано на рис. 10), то можно дать ответ на поставленный вопрос: «Комната с условным номером 204 находится а общежитии №1».

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К ОПЕРАЦИОННОЙ ОБСТАНОВКЕ

На этом этапе производится оценка требований к вычислительным ресурсам, необходимым для функционирования системы, определение типа и конфигурации конкретной ЭВМ, выбор типа и версии операционной системы. Объем вычислительных ресурсов зависит от предполагаемого объёма проектируемой базы данных и от интенсивности их использования. Если БД будет работать в многопользовательском режиме, то требуется подключение ее к сети и наличие соответствующей многозадачной операционной системы.

Для выполнения этого этапа необходимо знать (хотя бы ориентировочно) объём работы БД *(например, количество книг, авторов и заказчиков),* а также иметь представление о характере и интенсивности запросов.

Объём внешней памяти, необходимый для функционирования системы, складывается из двух составляющих:

× память, занимаемая модулями СУБД (ядро, утилиты, вспомогательные программы);

× память, отводимая под данные *(Мд)*. Наиболее существенным обычно является *Мд.*
Объём памяти М*д*, требуемый для хранения данных, можно приблизительно оценить по формуле:

$$M_д = 2 \sum_{l=1}^{n} l_i \times (N_i + N_{ai})$$

где *li*,- длина записи в *i*-й таблице (в байтах);
Ni - максимально возможное количество записей в *i*-й таблице;
Na - количество записей в архиве i-й таблицы.

Коэффициент 2 (два) перед суммой нужен для того, чтобы выделить память для хранения индексов, промежуточных данных, для выполнения объемных операций (например, сортировки) и т. п.

Например, посчитаем приблизительно, какой объем внешней памяти потребуется для хранения данных в книжном издательстве. Примем ориентировочно, что:
· одновременно осуществляется около пятидесяти проектов, работа над проектом продолжается в среднем два месяца (по 0,3К);
· в компании работает 100 сотрудников (по 0,2К на каждого сотрудника);
· издательство сотрудничает с тридцатью авторами (по 0,2К);
· в день обслуживается порядка двадцати заявок (по 0,1К);
· устаревшие данные переводятся в архив.
Тогда объём памяти для хранения данных за первый год примерно составит:
М*д =2(100*0,2+6(50*0,3)+30*0,2+250(20*0,1)) = 1232 К ≈1,2 М,*
где 250 - количество рабочих дней в году, а 12 мес./2 мес = 6 Объем памяти будет увеличиваться ежегодно на столько же при сохранении объёма работы.

Объем памяти, занимаемый программными модулями пользователя, обычно невелик по сравнению с объёмом самих данных, поэтому может не учитываться. Требуемый объём оперативной памяти определяется на основании анализа интенсивности запросов и объёма результирующих данных.

3. ВЫБОР СУБД И ДРУГИХ ПРОГРАММНЫХ СРЕДСТВ

На весь процесс проектирования БД и реализацию информационной системы принципиальным образом влияет выбор СУБД. Теоретически при выборе СУБД нужно принимать во внимание десятки факторов. Но практически разработчики руководствуются лишь собственной интуицией и несколькими наиболее важными критериями, к которым, в частности, относятся:

× тип модели данных, которую поддерживает данная СУБД, ее адекватность потребностям рассматриваемой предметной области;

× характеристики производительности системы;

× запас функциональных возможностей для дальнейшего развития информационной системы;

× степень оснащенности системы инструментарием для персонала администрирования данными;

× удобство и надежность СУБД в эксплуатации;

× стоимость СУБД и дополнительного программного обеспечения.

4. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БД

На этапе логического проектирования разрабатывается логическая структура БД, соответствующая логической модели ПрО. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД.

База данных создаётся на основании *схемы базы данных.* Инфологическую модель данных, построенную в виде ER-диаграммы, следует преобразовать в схему БД. Преобразование ER-диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, в таблицы-отношения (таблицы БД).
Для этого необходимо выполнить следующие *шаги процедуры проектирования* модели.
1. Представить каждый *стержень* (независимую сущность) таблицей базы данных (базовой таблицей) и специфицировать *первичный ключ* этой базовой таблицы.
2. Представить каждую *ассоциацию* (связь вида «многие-ко-многим» или «многие-ко-многим-ко-многим» и т. д. между сущностями) как базовую

таблицу. Использовать в этой таблице внешние ключи для идентификации участников ассоциации и специфицировать ограничения, связанные с каждым из этих внешних ключей.
3. Представить каждую *характеристику* как базовую таблицу с внешним ключом идентифицирующим сущность, описываемую этой характеристикой. Специфицировать ограничения на внешний ключ этой таблицы и её первичный ключ – по всей вероятности, комбинации этого внешнего ключа и свойства, которое гарантирует «уникальность в рамках описываемой сущности».
4. Представить каждое *обозначение*, которое не рассматривалось в предыдущем пункте, как базовую таблицу с внешнем ключом, идентифицирующим обозначаемую сущность. Специфицировать связанные с каждым таким внешним ключом ограничения.

5. Представить каждое свойство как поле в базовой таблице, представляющей сущность, которая непосредственно описывается этим свойством.

6. Для исключения из проекта непреднамеренных нарушений каких-либо принципов нормализации, выполнить процедуры нормализации.

7. Если в процессе нормализации было произведено разделение каких-либо таблиц, то следует *модифицировать инфологическую модель* базы данных и повторить перечисленные шаги.

8. Указать ограничения целостности проектируемой БД и дать (при необходимости) краткое описание полученных таблиц и их полей.

4.1 Особенности проектирования реляционной база данных

Проектирование реляционной базы данных проходит в том же порядке, что и проектирование БД других моделей данных, но имеет свои особенности.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных, которые обусловлены отсутствием средств явного представления типов множественных связей между объектами ПО и неразвитостью средств описания ограничений целостности на уровне модели данных.

Для решения подобных проблем проводится нормализация отношений. Нормализация схемы отношения выполняется путем декомпозиции схемы.

5. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БД

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, то есть отображения логической структуры БД в структуру хранения. Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам “физической” БД. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL). Принятые на этом этапе решения оказывают определяющее влияние на производительность системы.

Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только а соответствии с его правами доступа.