1.1. Введение в АИС и базы данных

Развитие вычислительной техники и появление емких внешних запоминающих устройств прямого доступа предопределило интенсивное развитие автоматических и автоматизированных систем разного назначения и масштаба, в первую очередь заметное в области бизнес-приложений. Такие системы работают с большими объемами информации, которая обычно имеет достаточно сложную структуру, требует оперативности в обработке, часто обновляется и в то же время требует длительного хранения. Примерами таких систем являются автоматизированные системы управления предприятием, банковские системы, системы резервирования и продажи билетов и т.д.

Другими направлениям, стимулировавшим развитие, стали с одной стороны, системы управления физическими экспериментами, реализующими сверхоперативную обработку в реальном масштабе времени огромных потоков данных от датчиков, а с другой - автоматизированные библиотечные информационно-поисковые системы ***(Слайд 2)***.

Это привело к появлению новой информационной технологии интегрированного хранения и обработки данных — *концепции баз данных,* в основе которой лежит механизм предоставления обрабатывающей программе из всех хранимых данных только тех, которые ей необходимы, и в форме, требуемой именно этой программе. При этом сама форма (структура данных и форматы полей, входящих в эту структуру) описывается на логическом, т.е. «видимом» из программы, уровне. Более того, поскольку различные программы могут по-разному «видеть» (а, следовательно, и использовать) одни и те же данные, то система должна сделать невидимыми - «прозрачными» для программы все данные, кроме тех, которые для нее являются «своими».

Банк данных (БнД) - это система специально организованных данных, программных, языковых, организационных и технических средств, предназначенных для централизованного накопления и коллективного многоцелевого использования данных. Термин «банк данных» используется сравнительно редко, а некоторыми авторами признается даже архаичным. В современной, в основном переводной литературе понятию банк данных соответствует понятие *системы баз данных*, хотя «банк данных» вполне адекватное и более широкое понятие.

Под *базой данных (БД)* обычно понимается именованная совокупность данных, отображающая состояние объектов и их отношений в рассматриваемой предметной области. Характерной чертой баз данных является *постоянство*: данные *постоянно* накапливаются и используются; состав и структура данных, необходимых для решения тех или иных прикладных задач обычно *постоянны* и стабильны во времени; отдельные или

даже все элементы данных могут меняться – но и это есть проявление постоянства – *постоянная* актуальность.

С*истема управления базами данных (СУБД)* - это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Иногда в составе банка данных выделяют *архивы*. Основанием для этого является особый режим использования данных, когда только часть данных находится под оперативным управлением СУБД. Все остальные данные (собственно архивы) обычно располагаются на носителях, оперативно не управляемых СУБД. Одни и те же данные в разные моменты времени могут входить как в базы данных, так и в архивы. Банки данных могут не иметь архивов, но если они есть, то в состав банка данных может входить и система управления архивами.

Проблемы совместного использования данных и периферийных устройств компьютеров и рабочих станций быстро породили модель вычислений, основанную на концепции файлового сервера - сеть создает основу для коллективной обработки, сохраняя простоту использования персонального компьютера, позволяет совместно использовать данные и периферию.

В этом смысле главной отличительной чертой баз данных является использование централизованной системы управления данными, причем как на уровне файлов, так и на уровне элементов данных. Централизованное хранение совместно используемых данных приводит не только к сокращению затрат на создание и поддержание данных в актуальном состоянии, но и к сокращению избыточности информации, упрощению процедур поддержания непротиворечивости и целостности данных.

Эффективное управление внешней памятью является основной функцией СУБД. Эти, обычно специализированные, средства настолько важны с точки зрения эффективности, что при их отсутствии система просто не сможет выполнять некоторые задачи уже потому, что их выполнение будет занимать слишком много времени. При этом, ни одна из таких специализированных функций, как построение индексов, буферизация данных, организация доступа и оптимизация запросов, не являются видимыми для пользователя и обеспечивают независимость между логическим и физическим уровнями системы: прикладной программист не должен писать программы индексирования, распределять память на диске и т.д.

Основные требования, предъявляемые к банкам данных, можно сформулировать следующим образом. ***(слайд 3)***

1.2. Компоненты банка данных

Определение банка данных предполагает, что с функционально-организационной точки зрения банк данных является сложной человеко-машинной системой, включающей в себя все подсистемы, необходимые для надежного, эффективного и продолжительного во времени функционирования.

В структуре банка данных выделяют следующие компоненты (подсистемы):

- информационная база;
- лингвистические средства;
- программные средства;
- технические средства;
- организационно-административные подсистемы и нормативно-методическое обеспечение.

(Слайд 4)

Информационная база

Данные, отражающие состояние определенной предметной области и используемые информационной системой, принято называть *информационной базой.* Информационная база состоит из двух компонент: 1) коллекции записей собственно данных и 2) описания этих данных — метаданных.

Данные отделены от описаний, но в то же время данные не могут использоваться без обращения к соответствующим описаниям.

Уже из определения базы данных и приведенных ранее основных требований следует, что данные могут использоваться (т.е., представляться) по-разному. С одной стороны, разные прикладные задачи требуют разных наборов данных, в совокупности обеспечивающих функциональную полноту информации, а с другой – они должны быть различны для различных категорий субъектов (разработчиков или пользователей). Также должны быть различными и способы описания самих данных, их природы, формы хранения, условий взаимной непротиворечивости.

В литературе по базам данных упоминаются три уровня представления данных — концептуальный, внутренний и внешний ***(Слайд 5)***.

Эти уровни представлений введены исходя из различного рассмотрения БД. Например, прикладному программисту требуются не все данные БД, а только некоторая их часть, используемая в его программе. Внешний уровень представления обеспечивает именно эту форму обмена данными.

Внутренний уровень - глобальное представление БД, определяет необходимые условия для организации хранения данных на внешних запоминающих устройствах.

Описание БД на концептуальном уровне представляет собой обобщенный взгляд на данные с позиций предметной области (разработчика приложений, пользователя или внешней информационной системы).

Внешний уровень представления данных не затрагивает физической организации (размещения) данных во внешней памяти, поэтому его называют иногда логическим уровнем. Соответственно внутренний уровень называют физическим уровнем ***(слайд 6).***.

Лингвистические средства

Многоуровневое представление БД предполагает соответствующие описания данных на каждом уровне и согласование одних и тех же данных на разных уровнях. С этой целью в состав СУБД включаются специальные языки для описания представлений внутреннего и внешнего уровней. Кроме того, СУБД должна включать в себя язык манипулирования данными (ЯМД). Желательно, также наличие тех или иных дополнительных сервисных средств, например, средств генерации отчетов.

Работа с базами данных предполагает несколько этапов: описание БД; описание частей БД, необходимых для конкретных приложений (задач, групп задач); программирование задач или описание запросов в соответствии с правилами конкретного языка и использованием языковых конструкций для обращения к БД; загрузка БД и т. д. ***(Слайд 7)***

Для выражения обобщенного взгляда на данные применяют *язык описания данных (ЯОД)* внутреннего уровня, включаемый в состав СУБД (отсюда следует, что одна и та же БД может описываться по-разному на ЯОД различных СУБД). Описание представляет собой модель данных и их отношений, т. е. структур, из которых образуется БД.

ЯОД позволяет определять схемы базы данных, характеристики хранимых и временных данных, параметры организации их хранения в памяти, а также может включать в себя средства поддержки целостности базы данных, ограничения доступа, секретности.

ЯМД обычно включает в себя средства запросов к базе данных и поддержания базы данных (добавление, удаление, обновление данных, создание и уничтожение БД, изменение определений БД, обеспечение запросов к справочнику БД).

Исторически первым типом структур данных, который был включен в языки программирования, была иерархическая структура. Некоторые ранние СУБД также предполагали использование в качестве основной модели иерархические структуры типа

дерева. Основанием для такого выбора было удобство представления (моделирования) естественных иерархических структур данных, существующих, например, в организациях.

В ряде предметных областей структура данных имеет более сложный вид, в котором поддерживаются связи типа «многие к одному», и которые могут быть представлены ориентированным графом. Такие структуры называют сетевыми. Для управления БД сетевой структуры международной ассоциацией Кодасил была предложена обобщенная архитектура системы с ЯОД схемы (модели БД) и подсхемы (модели части БД для конкретного приложения), а также ЯМД для оперирования с данными БД в прикладных программах.

Функциональные характеристики языков отражают возможности описания данных, средств представления запроса, обновления, поддержки целостности и секретности, включения в языки программирования, управления форматом ответов, средств запроса к словарю данных БД и т.д.

Качественные характеристики языков запросов могут определяться такими свойствами, как полнота, селективная мощность, простота изучения и использования, степень процедурности и модульности, унифицированность, производительность и эффективность. Рассмотрим некоторые из этих понятий.

Селективная мощность языков запросов характеризует возможность выбора данных по разным критериям. Данное понятие плохо поддается формализации: можно сказать, что язык с большей селективной мощностью позволяет сформулировать большинство запросов так, что ответ на них содержит меньше ненужных данных. Языки, обладающие малой селективной мощностью, в общем случае уже требуют привлечения дополнительных средств для анализа ответов на запросы (например, оценки пользователя).

Простота изучения является во многом субъективной оценкой и может быть в некоторой мере охарактеризована степенью его близости к естественному языку, требуемым для его освоения временем и необходимым уровнем подготовки пользователя.

Высокий *уровень процедурности*, свойственный реляционным языкам, определяется присущими реляционной модели свойствами, в частности, полным отделением логической структуры данных от структур хранения и стратегий доступа. Снижение уровня процедурности увеличивает свободу в выборе способов реализации языка, что позволяет осуществить его реализацию более оптимальным способом. Но необходимо отметить, что меньшая степень процедурности еще не означает автоматически меньшую сложность написания запросов. Некоторые сложные запросы

можно более просто сформулировать в виде алгоритма поиска ответа, в то время как его формулировка в декларативном виде может оказаться достаточно трудной.

Модульность построения языка характеризует возможность существования нескольких уровней языка и зависит от специфических свойств математической теории, лежащей в его основе. Минимальный уровень языка, обычно легко понимаемый пользователем, бывает достаточным для формулирования большинства запросов, и лишь формулировка сложных запросов может потребовать использования всех выразительных средств языка, о существовании которых пользователи начального уровня могут и не знать. Языки, не обладающие модульностью, требуют от пользователя знания почти всего объема средств языка, что усложняет процесс их изучения.

Наиболее распространенным языком для работы с базами данных является SQL (Structured Query Language), в своих последних реализациях предоставляющий не только средства для спецификации и обработки запросов на выборку данных, но так же и функции по созданию, обновлению, управлению доступом и т.д.

По существу SQL уже соединяет в себе и язык описания данных и язык манипулирования данными. Он не является полноценным языком программирования и, в случае его использования для организации доступа к БД из прикладных программ, SQL-выражения встраиваются в конструкции базового языка.

Являясь внутренним языком баз данных, SQL естественно отражает особенности конкретной СУБД. Сегодня это единственный стандартизованный язык фактографических баз данных, достаточно мощный и в тоже время, простой для понимания и использования язык. Сегодня, благодаря независимости от конкретных СУБД и межплатформенной переносимости, SQL стал языком распределенных баз данных и языком шлюзов, позволяющим совместно использовать СУБД разного типа.

Программные средства

Обработка данных и управление этой обработкой в вычислительной среде, а также взаимодействие с операционной системой и прикладными программами осуществляется комплексом программных средств ***(Слайд 8)***. В составе комплекса обычно выделяют следующие компоненты:

- **ядро**, обеспечивающее управление данными во внешней и оперативной памяти, а также протоколирование изменений;
- **процессор языка базы данных**, обеспечивающий обработку (трансляцию или компиляцию) и оптимизацию запросов на выборку и изменение данных;

- **подсистему (библиотеку) поддержки программных вызовов**, которая обслуживает прикладные программы управления данными, взаимодействующие с СУБД через средства пользовательского интерфейса;
- **сервисные программы** (системные и внешние утилиты), обеспечивающие настройку СУБД, восстановление после сбоев и ряд дополнительных возможностей по обслуживанию.

Большинство СУБД работают в среде операционной системы и тесно с ней связаны. Многопользовательские приложения, обработка распределенных запросов, защита данных требуют эффективно использовать ресурсы, управление которыми обычно является функцией ОС. Использование многопроцессорных систем и мультипоточных технологий обработки данных позволяет эффективно обслуживать параллельно выполняемые запросы, но требует координации использования ресурсов между ОС и СУБД. Соответственно, управление доступом и обеспечение защиты также обычно интегрируются с соответствующими средствами операционной системы.

Именно централизованное управление данными обеспечивает:

- сокращение избыточности хранимых данных;
- совместное использование хранимых данных;
- стандартизацию представления данных, упрощающую эксплуатацию БД;
- разграничение доступа к данным;
- целостность данных, обеспечиваемую процедурами, предотвращающими включение в БД неверных данных и ее восстановление после отказов системы.

Технические средства (*Слайд 9*)

Большинство банков данных создается и функционирует на основе универсальных вычислительных машин. Следует упомянуть и достаточно интенсивно развивавшееся в 80-90гг. направление создания машин баз данных – аппаратной реализации «нечисловой» обработки, в том числе параллельной и конвейерной обработки, ассоциативных процессоров и памяти.

Сегодня для реализации промышленных БД используются специализированные *серверы баз данных* – машины с повышенной отказоустойчивостью, высокопроизводительными подсистемами ввода-вывода и развитой периферией. Однако, для больших баз данных, функционирующих в промышленном режиме, обеспечение эффективной и бесперебойной работы должно основываться на использовании адекватных аппаратных средств.

Устройства ввода-вывода и накопители внешней памяти - традиционно узкое место любой базы данных. Объем и быстродействие накопителей являются, очевидно, важными параметрами. Однако, столь же значима и отказоустойчивость. Здесь следует отметить необходимость согласованных решений при распределении ролей между аппаратными и программными компонентами управления операциями ввода-вывода. Например, наличие буферной памяти в накопителе ускоряющей ввод-вывод (аппаратное кэширование) при сбоях системы во время выполнения операции записи в БД может привести к потере данных: переданные для записи данные еще будут находиться в буфере, а т.к. СУБД уже отметит операцию записи как уже завершившуюся и откат для восстановления данных станет невозможен.

Для повышения надежности хранения часто используют специализированные дисковые подсистемы – RAID (Redundant Array of Inexpensive Disk). Один логический RAID-диск - это несколько физических дисков, объединенных в одно устройство, управляемое специализированным контроллером, что позволяет распределять основные и системные данные между несколькими носителями (дисками), в том числе дублировать данные.

Не менее значима роль центрального процессора. Многие промышленные СУБД поддерживают многопроцессорную обработку. Использование еще одного процессора позволит ускорить обработку, однако следует учитывать, что на практике многопроцессорные системы требуют повышенного внимания при приобретении оборудования: надежно работают только сертифицированные системы, использующие соответствующие периферийные устройства.

Для распределенных и удаленных баз данных также важно сетевое окружение: связное оборудование и сетевые протоколы. Здесь важны не только показатели быстродействия, но и поддерживаемые ими возможности обеспечения безопасности.

Организационно-административные подсистемы

Организационно-методические средства не являются технической компонентой системы, однако трудно рассчитывать на устойчивое и долговременное функционирование банка данных, если будут отсутствовать необходимые методические и инструктивные материалы, регламентирующие работу пользователей, различных по своему статусу и уровню подготовленности.

Пользователи баз данных

В информационных системах, создаваемых на основе СУБД, способы организации данных и методы доступа к ним перестали играть решающую роль, поскольку оказались скрытыми внутри СУБД. Массовый, так называемый *конечный пользователь*, как правило, имеет дело только с внешним интерфейсом, поддерживаемым СУБД ***(Слайд 10)***.

Эти преимущества, как уже понятно, не могут быть реализованы путем механического объединения данных в БД. Предполагается, что в системе существует (как неотъемлемая составная часть) специальное должностное лицо (группа лиц) — *администратор базы данных (АБД),* который несет ответственность за проектирование и общее управление базой данных. АБД определяет информационное содержание БД. С этой целью он идентифицирует объекты БД и моделирует базу, используя язык описания данных. Получаемая модель служит в дальнейшем справочным документом для администраторов приложений и пользователей. Администратор решает также все вопросы, связанные с размещением БД в памяти, выбором стратегии и ограничений доступа к данным. В функции АБД входят также организация загрузки, ведения и восстановления БД и многие другие действия, которые не могут быть полностью формализованы и автоматизированы.

Администратор приложений (или, если таковой специально не выделяется - администратор БД) определяет для приложений подмодели данных. Тем самым разные приложения обеспечиваются собственным «взглядом» но не на всю БД, а только на требуемую для конкретного приложения («видимую») ее часть. Вся остальная часть БД для данного приложения будет «прозрачна».

Прикладные программисты имеют, как правило, в своем распоряжении один или несколько языков программирования, с помощью которых генерируются прикладные программы.

2.1. Классификация баз данных

Классификация баз и банков данных может быть произведена по разным признакам (и относящихся к разным компонентам и сторонам функционирования БнД), среди которых можно выделить, например, следующие ***(Слайд 2)***.

По *форме представляемой информации* можно выделить фактографические, документальные, мультимедийные, в той или иной степени соответствующие цифровой, символьной и другим (не цифровой и не символьной) формам представления информации в вычислительной среде. К последним можно отнести картографические, видео, аудио, графические и другие БД.

По *типу хранимой (не мультимедийной) информации* можно выделить фактографические, документальные, лексикографические БД. Лексикографические базы – это классификаторы, кодификаторы, словари основ слов, тезаурусы, рубрикаторы и т.д., которые обычно используются в качестве справочных совместно с документальными или фактографическими БД. Документальные базы подразделяются по уровню представления информации – полнотекстовые (так называемые «первичные» документы) и библиографическо-реферативные («вторичные» документы, отражающие на адресном и содержательном уровне первичный документ).

По *типу используемой модели данных* выделяют три классических класса БД: иерархические, сетевые, реляционные. Развитие технологий обработки данных привело к появлению постреляционных, объектноориентированных, многомерных БД, которые в той или иной степени соответствуют трем упомянутым классическим моделям.

По ***топологии хранения*** данных различают локальные и распределенные БД.

По ***типологии доступа*** **и** ***характеру использования*** хранимой информации БД могут быть разделены на специализированные и интегрированные.

По ***функциональному назначению*** (характеру решаемых с помощью БД задач и, соответственно, характеру использования данных) можно выделить операционные и справочно-информационные. К последним можно отнести ретроспективные БД (электронные каталоги библиотек, БД статистической информации и т.д.), которые используются для информационной поддержки основной деятельности, и не предполагают внесение изменений в уже существующие записи, например, по результатам этой деятельности. Операционные БД предназначены для управления различными технологическими процессами. В этом случае данные не только извлекаются из БД, но и изменяются (в том числе добавляются) в том числе в результате этого использования.

По ***сфере возможного применения*** можно различать универсальные и специализированные (или проблемно-ориентированные) системы.

По ***степени доступности*** можно выделить общедоступные и БД с ограниченным доступом пользователей. В последнем случае говорят об управляемом доступе, индивидуально определяющем не только набор доступных данных, но и характер операций которые доступны пользователю.

Следует отметить, что представленная классификация не является полной и исчерпывающей. Она в большей степени отражает исторически сложившееся состояние дел в сфере деятельности, связанной с разработкой и применением баз данных.

Типология баз данных с точки зрения информационных процессов

БД могут соотноситься с различными уровнями *информационных процессов:* уровень информационных технологий (ИТ), уровень системы (ИС), уровень информационных ресурсов (ИР). ***(слайд 3)***

На уровне информационных технологий БД определяется как взаимосвязанная совокупность файлов ОС, содержащих данные о предметной области решаемой задачи. При этом основное внимание уделяется *физической структуре БД*.

На уровне информационных систем БД рассматривается как компонент, представляющий собой информационную модель предметной области. Здесь наиболее важной является проблема *логической структуры БД.*

При рассмотрении БД на уровне информационных ресурсов БД трактуется как элемент мировых ИР. Основной характеристикой здесь является *содержание БД*, хотя и структуры данных также немаловажны.

2.2. Фактографические и документальные БД

Главное отличие фактографических и документальных БД состоит в структуре единицы хранения информации.

Под *единицей хранения информации* будем понимать совокупность данных, которая с точки зрения информационной системы представляет собой единое целое. Единица хранения определяет свойства целостности и непротиворечивости данных.

С точки зрения структуры единицы хранения принято различать хорошо структурированные данные и слабо структурированные данные.

Хорошо структурированные данные — это данные, в которых каждую единицу хранения информации можно представить в качестве конечного набора атрибутов. При этом каждый из них будет принимать точно определенное значение.

Слабоструктурированные данные — это данные, в которых каждую единицу хранения также представляют конечным числом атрибутов, но значение атрибута точно не определено, зависит от контекста использования и может иметь в свою очередь сложную структуру.

Фактографические БД – БД, ориентированные на хранение хорошо структурированных данных. Единицей хранения в таких БД служит описание «факта» конечным четко определенным множеством характеристических свойств.

При построении концептуальной модели таких БД предметная область (ПрО) естественно декомпозируется на объекты и связи между ними. Каждое

характеристическое свойство объекта имеет атомарное значение, которое не зависит от контекста использования.

Документальные БД – предназначены для хранения слабо структурированных данных. Единицей хранения при этом является документ, заданный конечным (но не фиксированным) набором полей в общем случае произвольной длины.

При построении документальных БД обычно ПрО представляется как совокупность в общем случае не взаимодействующих объектов. Набор характеристических свойств объекта конечен, но не фиксирован. Значение характеристического свойства может быть множественным и может зависеть от контекста использования ***(слайд 4).***

С точки зрения методов и алгоритмов поиска фактографические БД рассматривают как информационное обеспечение поиска данных, а документальные БД – как информационное обеспечение поиска информации.

Отличия этих двух видов поиска представлены на слайде ***(слайд 5)*.**

При поиске данных обычно ищут полное совпадение запроса с элементом данных. При поиске данных результаты выводятся простой индукцией, например, если А и В, то С. Поиск информации намного ближе к методам дедукции: отношения описываются только степенью уверенности или неуверенности. В информационном поиске, как правило, стратегия поиска построена по принципу усечения первоначальных результатов поиска, что и приводит к логике «от общего к частному». Из этого следует детерминистское описание модели поиска данных и вероятностная модель информационного поиска.

При информационном поиске наличие атрибута не всегда является необходимым и достаточным для отнесения записей к множеству отыскиваемых. Это означает, что каждая из записей (документов) относится к некоторой части информационной потребности пользователя. Это свойство соответствия документов потребности называется *релевантностью*. Различают формальную и истинную релевантность. Первая имеет обычно численное выражение и рассчитывается поисковой системой, вторая — это оценка пользователя в части соответствия реальной потребности, порожденной проблемной ситуацией в основной деятельности пользователя.

При поиске данных все найденные данные, которые совпали с запросом, выдаются пользователю. При информационном поиске, возможна ситуация, что практически все документы БД в той или иной мере могут считаться релевантными запросу и уже поэтому документы будут упорядочены, например, по степени формальной релевантности, и будут выданы только несколько первых.

Язык запросов для поиска данных, как правило, искусственный, имеющий строгий синтаксис и ограниченный словарь, при поиске информации же предпочтительнее естественный язык, хотя и с некоторыми исключениями, а в настоящее время «естественный язык» сводится к списку ключевых слов. При поиске данных запрос обычно является полной спецификацией того, что нужно найти и в каком виде показать, при информационном поиске — неполной, кроме того, многие действия совершаются информационно-поисковой системой по умолчанию.

2.3. БД оперативной и ретроспективной информации. Хранилища данных

С точки зрения основных особенностей ПрО и решаемых задач можно выделить два основных класса БД – оперативной и ретроспективной информации.

БД оперативной информации являются основой так называемых ***OLTP-приложений*** (Оп-Line Transactions Processing). Типичными примерами OLTP-приложений являются системы складского учета, системы заказов билетов, банковские системы, выполняющие операции по переводу денег, и т. п. Основная функция подобных систем заключается в одновременном выполнении большого количеств коротких *транзакций* – завершенных блоков операций манипулирования данными, например: "снять некоторую сумму денег со счета *A* и добавить эту сумму на счет *B",* "продать пассажиру билет на заданный поезд на заданное место на определенную дату". Завершенность транзакции означает, что при возникновении ошибки транзакция должна целиком откатиться и вернуть БД к состоянию, которое было до начала транзакции (не должно быть ситуации, когда деньги сняты со счета *A*, но не поступили на счет *B*).

Основные особенности OLTP-приложений:

1. В единицу времени одновременно выполняется большое число транзакций (к системе может быть подключено несколько тысяч пользователей, работающих в одно и то же время).

2. Практически все запросы к базе данных, которые должны выполняться в реальном времени, состоят из команд вставки, обновления, удаления.

3. Запросы на выборку в основном предназначены для предоставления пользователям возможности выбора из различных справочников, и большая часть этих запросов известна заранее еще на этапе проектирования.

Таким образом, критическими для OLTP-приложений является скорость и надежность выполнения коротких операций обновления данных.

БД ретроспективной информации входят в состав документальных ИС, ориентированных на задачи информационного поиска, а также в *OLAP-приложения* (Оп-Line Analitical Processing, оперативная аналитическая обработка данных). Это обобщенный термин, характеризующий принципы построения *систем поддержки принятия решений* (DSS, Decision Support System), а также *хранилищ данных* (data warehouse) и *систем интеллектуального анализа данных* (data mining). Такие системы предназначены для установления зависимостей между данными (например, можно попытаться определить, как связан объем продаж товаров с характеристиками потенциальных покупателей) или для проведения анализа, отвечающего на вопросы "что если...".

БД ретроспективной информации характеризуются следующими особенностями:

1. Добавление в БД новых данных происходит относительно редко крупными блоками.
2. Данные из БД обычно никогда не удаляются.
3. Запросы к данным являются нерегламентированными и, как правило, достаточно сложными. Очень часто новый запрос формулируется аналитиком для уточнения результата, полученного при выполнении предыдущего запроса.
4. Скорость выполнения запросов важна, но не критична.

Для OLAP-приложений характерно, что перед загрузкой данные проходят различные процедуры "очистки", связанные с тем, что в одну базу могут поступать данные из многих источников, имеющих различные форматы представления для одних и тех же данных, данные могут быть некорректны, ошибочны и т.п..

Хранилища данных

Огромное количество информации, накопленной в оперативных базах данных, позволяет, например, поставить задачу применения систем поддержки принятия решений. Однако системы оперативной обработки чаще всего проектируются без учета какой-либо поддержки подобных требований, поэтому преобразование обычных систем OLTP в системы поддержки принятия решений оказывается чрезвычайно сложной задачей. Как правило, типичная организация имеет множество различных систем операционной обработки с перекрывающимися, а иногда и противоречивыми определениями, например с разными типами, выбранными для представления одних и тех же данных. Основной задачей является преобразование накопленных архивов данных в источник новых знаний, причем таким образом, чтобы пользователю было предоставлено единое интегрированное и консолидированное

представление о данных организации. Концепция хранилища данных была задумана как технология, способная удовлетворить требования систем поддержки принятия решений и базирующаяся на информации, поступающей из нескольких различных источников оперативных данных.

Концепция хранилища данных первоначально была предложена как решение, обеспечивающее доступ к данным, накопленным в нереляционных системах. Предполагалось, что такое хранилище информации позволит организациям использовать свои архивы данных для эффективного решения деловых задач. Однако из-за чрезвычайной сложности и невысокой производительности подобных систем, созданных на начальных этапах, первые попытки создания хранилищ информации в целом оказались неудачными. С тех пор к концепции хранилищ информации возвращались вновь и вновь, но только в последние годы технология хранилищ данных стала рассматриваться как ценное и жизнеспособное решение.

Хранилище данных - предметно-ориентированный, интегрированный, привязанный ко времени и неизменяемый набор данных, предназначенный для поддержки принятия решений.

В приведенном определении указанные характеристики данных рассматриваются следующим образом. ***(слайд 6)***

- *Предметная ориентированность.* Хранилище данных организовано вокруг основных предметов (или субъектов) организации (например, клиенты, товары и сбыт), а не вокруг прикладных областей деятельности (выставление счета клиенту, контроль запасов и продажа товаров). Это свойство отражает необходимость хранения данных, предназначенных для поддержки принятия решений, а не обычных оперативно-прикладных данных.
- *Интегрированность*. Смысл этой характеристики состоит в том, что оперативно-прикладные данные обычно поступают из разных источников, которые часто имеют несогласованное представление одних и тех же данных, например, используют разный формат. Для предоставления пользователю единого обобщенного представления данных необходимо создать интегрированный источник, обеспечивающий согласованность хранимой информации.
- *Привязка ко времени*. Данные в хранилище точны и действительны только в том случае, если они привязаны к некоторому моменту или промежутку времени: хранимая информация фактически представляет собой набор снимков состояния

данных.

• *Неизменяемость*. Это означает, что данные не обновляются в оперативном режиме, а лишь регулярно пополняются за счет информации из оперативных систем обработки. При этом новые данные никогда не заменяют, а лишь дополняют прежние. Таким образом, база данных хранилища постоянно пополняется новыми данными, последовательно интегрируемыми с уже накопленной информацией.

Конечной целью создания хранилища данных является интеграция корпоративных данных в едином репозитории, обращаясь к которому пользователи могут выполнять запросы, подготавливать отчеты и проводить анализ данных. Подводя итог, можно сказать, что технология хранилищ данных — это технология управления данными и их анализа.

Сравнение систем OLTP и хранилищ данных

СУБД, созданная для поддержки оперативной обработки транзакций (OLTP), обычно рассматривается как непригодная для организации хранилищ данных, поскольку к этим двум типам систем предъявляются совершенно разные требования. Например, системы OLTP проектируются с целью обеспечения максимально интенсивной обработки фиксированных транзакций, тогда как хранилища данных — прежде всего для обработки единичных *произвольных запросов*. На слайде ***(слайд 7)*** для сравнения приведены основные характеристики типичных систем OLTP и хранилищ данных.

Проблемы разработки и сопровождения хранилищ данных

Перечислим потенциальные проблемы, связанные с разработкой и сопровождением хранилищ данных ***(слайд 8)***.

- *Недооценка ресурсов, необходимых для загрузки данных*: многие разработчики склонны недооценивать время, необходимое для извлечения, очистки и загрузки данных в хранилище.
- *Скрытые проблемы источников данных*: проблемы, связанные с источниками данных, поставляющими информацию в хранилище, могут быть обнаружены только спустя несколько лет после начала их эксплуатации.
- *Отсутствие требуемых данных в имеющихся архивах*: в хранилищах данных часто возникает потребность получить сведения, которые не учитывались в оперативных системах, служащих источниками данных. В таком случае организация должна решить, стоит ей модифицировать существующие системы OLTP или же создать новую систему по сбору недостающих данных

- *Повышение требований конечных пользователей*
- *Унификация данных*: создание крупномасштабного хранилища данных может быть связано с решением серьезной задачи унификации данных, но унификация способна уменьшить ценность собранной информации
- *Высокие требования к ресурсам*: может потребоваться огромный объем дискового пространства.
- *Владение данными*: создание хранилища данных может потребовать изменения статуса конечных пользователей в отношении прав владения данными
- *Сложное сопровождение*: любая реорганизация деловых процессов или источников данных может отразиться на работе хранилища данных
- *Долговременный характер проектов*
- *Сложности интеграции*

Локальные и распределенные БД

В общем случае режимы работы с БД можно классифицировать по следующим признакам:

- многозадачность - однопользовательский или многопользовательский;
- правило обслуживания запросов – последовательное или параллельное;
- схема размещение данных – централизованная или распределенная БД.

Следует отметить, что общая тенденция развития технологий обработки данных вполне соответствует этапам развития средств вычислительной техники и информационных технологий, и в первую очередь – сетевых. В этом смысле следует выделить два класса: системы распределенной обработки данных и системы распределенных баз данных.

Системы распределенной обработки данных в основном отражают структуру и свойства многопользовательских операционных систем с базой данных, размещенной на большом центральном компьютере (мэйнфрейме). Еще до недавнего времени это был единственно возможный вариант вычислительной среды для реализации больших баз данных. Клиентские места в этом случае реализовались либо в виде терминалов или мини-ЭВМ, обеспечивающих в основном ввод-вывод данных и не имеющих собственных вычислительных ресурсов для функционально-ориентированной обработки получаемых данных.

Развитие сетевых технологий в сочетании с широким распространением персональных ЭВМ и внедрением стандартов открытых систем привело к появлению

систем баз данных размещенных в сети разнотипных компьютеров. Такие системы распределенных баз данных обеспечивают обработку распределенных запросов, когда при обработке одного запроса используются ресурсы базы, размещенные на различных ЭВМ сети. Система распределенных баз данных состоит из узлов, каждый из которых является СУБД, а узлы взаимодействуют между собой так, что база данных любого узла будет доступна пользователю, так как если бы она была локальной. Архитектура распределенной БД приведена на слайде ***(слайд 9)***.

Соотношение основных требований и свойств СУБД: система компромиссов ***(слайд 10)***

В общем случае можно сказать, что основные задачи обработки данных, решаемые на основе концепций баз данных, сводятся к следующим вопросам:

1). Каким образом сложные нелинейные структуры данных представить в виде линейных – наиболее соответствующих принципу последовательного представления (хранения) в машинной памяти.

2). Каким образом организовать данные, чтобы была возможность эффективного внесения, удаления и редактирования данных.

3). Как организовать данные, чтобы использование пространства памяти (плотность данных) было достаточно рациональным, а скорость доступа к записям данных высокой.

4). Каким образом организовать данные, чтобы поиск был эффективным и позволял отыскивать записи по нескольким ключам.

Создание базы данных - это по существу попытка найти компромисс сразу по нескольким направлениям и сочетаниям нескольких взаимообратных факторов (с точки зрения их влияния на показатель общей эффективности системы), в том числе, следующих ***(слайд 11)***:

1) Эффективность – простота;
2) Скорость выборки – стоимость (сложность) аппаратных средств;
3) Скорость выборки – сложность процедур доступа;
4) Плотность данных – время доступа и сложность процедур;
5) Независимость данных – производительность;
6) Гибкость средств поиска – избыточность данных или
7) Гибкость поиска – скорость поиска;
8) Сложность процедур доступа – простота обслуживания.

4.1. Основные понятия реляционной модели данных

Реляционная модель является удобной и наиболее привычной формой представления данных в виде таблицы. Такой способ представления

1) понятен пользователю-непрограммисту;

2) позволяет легко изменять схему – присоединять новые элементы данных и записи без изменения соответствующих подсхем;

3) обеспечивает необходимую гибкость при обработке непредвиденных запросов.

В математических дисциплинах понятию «таблица» соответствует понятие «отношение» (relation). Отсюда и произошло название модели – реляционная. Т.е., применительно к базам данных понятия «реляционная БД» и «табличная БД» по существу являются синонимами.

Функциональным назначением реляционной модели является обеспечение согласованности структур данных, операций манипулирования данными, целостность данных ***(слайд 2).***

Одним из основных преимуществ реляционной модели ПрО является ее однородность. Все данные рассматриваются как хранимые в таблицах, в которых каждая строка имеет один и тот же формат. Каждая строка в таблице представляет некоторый объект реального мира или соотношение между объектами. Пользователь должен сам (для себя) решить вопрос, обладают ли выделенные сущности однородностью (общностью), чем и решается проблема пригодности модели для предполагаемого применения.

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие: *домен, отношение, кортеж, кардинальность, атрибуты, степень, первичный ключ.* Соотношение этих понятий иллюстрируется ***Слайдом 3.***

Домен – это совокупность значений, из которой берутся значения соответствующих атрибутов определенного отношения. С точки зрения программирования домен – это тип данных, определяемый системой (стандартный) или пользователем ***(слайд 4)***.

Отношение. Отношение имеет две части - заголовок и тело. Нестрого говоря, заголовок - это атрибуты, а тело - это картежи. Заголовок для базового отношения, т.е. значение базовой переменной-отношения, очевидно, вполне конкретен и известен системе, поскольку он задается как часть определения соответствующей базовой переменной-отношения. Т.к. результат обязательно должен иметь вполне определенный тип отношения, поэтому, если рассматривать свойство реляционной замкнутости более строго, каждая реляционная операция должна быть определена таким образом, чтобы

выдавать результат с надлежащим типом отношения (в частности, с соответствующим набором имен атрибутов или заголовком) ***(слайд 5, 6)***.

Первичный ключ – это столбец или некоторое подмножество столбцов, которые уникально, т.е. единственным образом определяют строки. Первичный ключ, который включает более одного столбца, называется множественным, или комбинированным, или составным. Правило целостности объектов утверждает, что первичный ключ не может быть полностью или частично пустым, т.е. иметь значение null.

Остальные ключи, которые можно также использовать в качестве первичных, называются потенциальными или *альтернативными* ключами.

Модель предъявляет к таблицам следующие требования:

1. данные в ячейках таблицы должны быть структурно неделимыми;
2. данные в одном столбце должны быть одного типа;
3. каждый столбец должен быть уникальным (недопустимо дублирование столбцов);
4. столбцы размещаются в произвольном порядке;
5. строки размещаются в таблице также в произвольном порядке;
6. столбцы имеют уникальные наименования.

4.2. Основы реляционной алгебры

С точки зрения внешнего представления (абстрагирования на логическом уровне) объектов реального мира *модель данных* – это основные понятия и способы, используемые при анализе и описании предметной области.

Среди многих попыток представить обработку данных на формальном абстрактном уровне реляционная модель, предложенная Э.Ф. Коддом, стала по существу первой работоспособной *моделью данных*, поскольку помимо средств описания объектов имела эффективный инструментарий преобразований этих описаний - операции реляционной алгебры.

Реляционная алгебра может быть задана множеством *T* реляционных таблиц (отношений) и множеством *O* алгебраических операций над ними. Все операции из множества *O* переводят операнды-отношения из *T* в отношения, принадлежащие *T*.

К числу операций реляционной алгебры (в том виде, в котором она была определена Э.Ф. Коддом) относят следующие: ***(слайд 7)***

- бинарные операции:

Y - объединение

I - пересечение

\ - разность

× - декартово произведение

J - соединение

/ - деление

- унарные операции:

S - выборка

Pr - проекция.

Этот набор операций можно классифицировать следующим образом:

- теоретико-множественные операции (аналогичные одноименным операциям в теории множеств): объединение, пересечение, разность, декартово произведение;

- специальные реляционные операции: выборка, проекция, соединение, деление.

Рассмотрим подробнее операции реляционной алгебры.

Объединение возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим **(слайд 8).**

Пересечение возвращает отношение, содержащее все кортежи, которые принадлежат одновременно двум заданным отношениям ***(слайд 9).***

Разность возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух заданных отношений и не принадлежат второму ***(слайд 10).***

Произведение - возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям ***(слайд 11).***

Выборка. Пусть задано логическое выражение – условие отбора F с атрибутами некоторой реляционной таблицы R. Тогда результатом выборки $S_F(R)$ будет подмножество множества записей R, для которых F имеет значение *истина* ***(слайд 12).***

Проекция. Пусть реляционная таблица R имеет n атрибутов. Зададим подмножество атрибутов a_1, a_2, …, a_m ($m \leq n$). Тогда результатом проекции $Pr_{a_1,a_2,\ldots,a_m}(R)$ будет реляционная таблица, имеющая m атрибутов из n атрибутов таблицы R. Следует отметить, что в результате операции количество кортежей может уменьшиться, т.к. по определению отношение не может содержать одинаковые кортежи ***(слайд 13).***

Соединение. Рассмотрим вариант так называемого естественного соединения двух реляционных таблиц по равенству значений заданных атрибутов. Будем считать, что в отношениях сравниваются одноименные атрибуты. Тогда естественным соединением отношений R и P по равенству атрибутов a_1, a_2, …, a_m (каждый из которых присутствует в обоих отношениях) будет отношение, полученное после выборки из декартового

произведения отношений *R* и *P* только тех кортежей, на которых значения заданных в операции атрибутов совпадают. При этом значения одноименных атрибутов в результирующем кортеже появляются один раз, а не дважды ***(слайд 14)***.

Деление. Отношение, полученное в результате деления *R/P,* содержит в качестве атрибутов те, и только те атрибуты делимого *R,* которые отсутствуют в делителе *P*, а в качестве кортежей в результат деления включаются те кортежи делителя, которые при декартовом умножении частного на делитель *P* содержаться в делимом *R* ***(слайд 15)***.

Результат выполнения любой операции над отношением также является отношением, поэтому результат одной операции может использоваться в качестве исходных данных для другой. Другими словами, можно записывать вложенные реляционные выражения, т.е. выражения, в которых операторы сами представлены реляционными выражениями, причем произвольной сложности. Эта особенность называется свойством *реляционной замкнутости*.

Реляционная алгебра имеет набор правил вывода типов (отношений), позволяющих вывести тип (отношение) на выходе произвольной реляционной операции, зная типы (отношения) на входе этой операции. Задав такие правила для всех операций, можно гарантировать, что для реляционного выражения любой сложности будет вычисляться результат, имеющий вполне определенный тип (отношение) и, в частности, известный набор имен атрибутов.

Рассмотренные восемь операторов Кодда не являются минимальным набором, так как не все из них примитивны, т.е. часть из них можно определить через другие операторы. Действительно, операции соединения, пересечения и деления можно определить через остальные пять. Эти пять операций (выборка, проекция, произведение, объединение и разность) можно рассматривать как примитивные в том смысле, что ни одна из них не выражается через другие. Они образуют минимальный набор, но, тем не менее, необязательно единственно возможный. Кроме того, остальные три операции (в особенности операция соединения) на практике используются настолько часто, что, несмотря на то, что они не являются примитивными, имеет смысл обеспечить их непосредственную поддержку.

Предшествующее рассмотрение алгебры представлено в контексте только операций выборки данных. Однако, как отмечается в классических введениях к реляционной алгебре, ее основная цель - обеспечить запись реляционных выражений, позволяющих определять:

- области выборки, т.е. тех данных, которые должны быть доставлены в результате выполнения операции выборки;
- области обновления, т.е. данных, которые должны быть вставлены, изменены или удалены в результате выполнения операции обновления;
- правила поддержки целостности данных, т.е. некоторых особых требований, которым должна удовлетворять база данных;
- производные переменные-отношения, т.е. те данные, которые должны быть включены в представления базы данных;
- требования устойчивости, т.е. данные, которые должны быть включены в контролируемую область для некоторых операций управления параллельным доступом к информации;
- ограничения защиты, т.е. данные, для которых осуществляется тот или иной тип контроля доступа.

В целом, выражения реляционной алгебры служат для символического высокоуровневого представления намерений пользователя (например, в отношении некоторого определенного запроса). И именно потому, что подобные выражения являются символическими и высокоуровневыми, ими можно манипулировать в соответствии с различными высокоуровневыми правилами преобразования, в том числе и для оптимизации процедур выполнения запросов на данные.

5.1. Документальные информационные системы, основанные на концепции БД

Элементом информационного массива АИС является документ. Документом при этом можно считать как всю совокупность записей (обычно текстов), хранящихся в БД, так и отдельные фрагменты текста - заголовки разделов, абзацы и т. д. при этом большую проблему представляет учет контекста употребления слова, зависящий больше не от синтаксических, а от семантических критериев. При поиске может использоваться сходство внутренней структуры отдельных документов, представленной, например, средствами формального языка типа SGML (Standard Generalized Markup Language). Помимо текста, для поиска могут использоваться содержащиеся в статьях уравнения, таблицы и графики, хотя пока такие средства поиска разрабатываются только для узких специализированных применений (поиск по структурным химическим формулам, элементам электронных схем и т. д.). В более широком смысле, объектами текста могут являться заголовки или абзацы, примечания, ссылки, названия таблиц и т. п. В некоторых системах применяются свернутые (сигнатурные) образы документов.

Организация данных в БД документальной информации построены на тех же принципах, что и БД фактографических систем. Однако есть и существенные различия, которые обусловлены в первую очередь информационной природой элементов данных:

1. Запись базы данных – документ, который задается как набор в общем случае *необязательных* полей, для каждого из которых определены имя и тип. Допустимо большинство стандартных типов (так называемые «форматные» поля, задающие числовые, символьные и другие величины), а также текстовые. Текстовые поля имеют переменную длину и композиционную структуру, не имеющую прямых аналогов среди стандартных типов языков программирования: текстовое поле состоит из параграфов; параграф - из предложений; предложение - из слов. При этом идентифицируемым (адресуемым) элементом данных с точки зрения хранения будет *поле*, а с точки зрения поиска (атомарным семантически значимым) – *слово*. Вследствие этого поисковые структуры строятся в виде инвертированных файлов.

2. Семантическая природа текстовых полей, представляющих смысл в основном на естественном языке, определяет необходимость учитывать важнейшие свойства используемых терминов: синонимию, полисемию, омонимию, контекстную обусловленность смысла отдельного слова и возможность выразить один смысл многими способами. Вследствие этого поисковые *индексы* могут быть отличны от соответствующих словоформ поля (***слайд 2***).

Для рассмотрения особенностей реализации поиска информации важно понимать тот простой факт, что поиск – это процесс, сводящийся к *отбору* через соотнесение отыскиваемого с объектами, хранящимися в массиве. Причем определяющими для понимания методологической основы автоматизации информационного поиска являются два следующих фактора:

1) сравниваются не сами объекты (они, как отмечалось, не очень удобны для сравнения), а описания - так называемые «поисковые образы»;

2) сам процесс является сложным (составным, не одноактным) и обычно реализуется последовательностью разнотипных операций.

Первый фактор имеет коммуникативную природу, что предполагает решение на уровне лингвистических средств. Второй – технологическую, предполагающую, что задача реализации процесса поиска сводится к задаче построения структур данных и алгоритмов обработки.

Таким образом, до начала поиска необходимо *выразить* информационную потребность, т. е. *специфицировать* «образец» той информации, которая необходима для

решения задачи в сфере ОД, а по окончании – определить степень решения задачи ОД (а не только соответствие слов запроса словам найденных документов) (***слайд 3***).

Для задач содержательного (семантического) поиска, реализуемого в дискретной вычислительной среде, такой образец обычно представляется набором атрибутов, отражающих свойства объекта в форме, скорее фактографической, чем аналитической.

Атрибут, как *поисковый признак,* задается парой <*имя, значение*> и может быть представлен в *позиционной* или *ключевой форме*.

Первая аналогична табличному способу представления данных о свойствах некоторого множества объектов: отдельному *i*-ому атрибуту соответствует *i*-я колонка, каждая ячейка которой содержит значение этого атрибута, свойственное отдельному объекту. Характерными чертами такого способа являются: 1) атомарность, т. е. отдельный атрибут отдельного объекта имеет строго одно значение; 2) предопределенность набора атрибутов – все существенные атрибуты объектов, информация о которых должна быть внесена в такую форму, должны быть определены на шаге, предшествующем построению таблицы и внесению в неё значений атрибутов.

Вторая - *ключевая форма* - имеет вид всем знакомого математического выражения: *имя атрибута = значение*. Такой способ не имеет указанных выше ограничений (позиционность параметров), однако порождает процедурную избыточность: необходимо предварительно определить процедуры разбора выражений, специфицирующих значения атрибутов. Для документальных систем, поисковые образы в которых представлены набором дескрипторов, атрибут задается неявно предикатом *«поисковый образ имеет в составе дескриптор»,* а сам дескриптор является значением атрибута.

Безусловно, в качестве поискового образа (ПО) может выступать и полное, «аналитическое» описание. Однако такое решение также имеет недостатки: 1) технические возможности для создания полнотекстовых баз данных появились сравнительно недавно, причем содержание, по которому возможен поиск, представлено, в основном, в форме текстов; 2) свободная лексика, авторская точка зрения и стиль изложения, свойственные первичным документам, затрудняют для пользователя нахождение общего с автором лексического пространства.

Поэтому в классе поисковых задач «общность» представления предметной области достигается другим путем – построением поискового образа на основе свойства концентрации информации, в частности, снижением детальности понятий и их связей, а также нормализацией лексики. Например, при создании вторичного документа (реферата, списка ключевых слов, классификационного кода) содержание первичного редуцируется до уровня перечисления основных понятий, в той или иной степени однозначно

характеризующих его содержание, но в контексте именно той предметной области, для которой создается база данных. В свою очередь, для обозначения таких характеристических понятий используется ограниченная (нормализованная) лексика, снижающая влияние свойств синонимии и полисемии. Представление информационного содержимого конкретных документов в виде поисковых образов обеспечивает, с одной стороны, очень эффективную вычислительную процедуру (когда отбор производится по условию простого сопоставления отдельных терминов запроса с терминами документов), а с другой стороны – дает пользователю возможность получать достаточно хорошие, семантически полные и точные ответы на запросы, выражаемые упрощенным «телеграфным» стилем, где семантические отношения редуцированы до уровня отношения «совместной встречаемости».

Такое решение позволяет обеспечить полноту поиска, однако приводит к снижению точности отождествления реальной потребности с содержанием отдельного документа, что, в свою очередь, может быть компенсировано процедурной избыточностью – обычно последующим перебором уже самим пользователем найденных системой потенциально полезных первоисточников, количество которых будет уже вполне приемлемым. Таким образом, назначение ПО и, соответственно, принципы его построения, определяются именно задачами информационно-поисковой деятельности: используя операции упорядочения и выборки, сократить множество «перебираемых» объектов и, тем самым, объемы ресурсов, используемых при этом.

7.1. Модели многоуровневой архитектуры систем баз данных

В области проектирования и разработки систем баз данных используются различные средства моделирования, причем даже в рамках одной конкретной системы необходим целый комплекс моделей разного назначения.

Опубликованный в 1975 году отчет ANSI/X3/SPARC зафиксировал не только широкое признание концепций многоуровневой архитектуры систем баз данных, но и необходимость явного выделения специального *концептуального* уровня представления базы данных, единого для всех ее приложений и независимого от них. Кроме этого уровня предусматривались еще два уровня: внутренний уровень, который должен обеспечивать поддержку представления хранимой базы данных, и внешний, поддерживающий представления базы данных “с точки зрения” приложений. На каждом архитектурном уровне предполагалось использование той или иной модели данных. Кроме того, на внешнем (прикладном, пользовательском) уровне таких моделей может быть несколько.

Модели, а также схемы, специфицируемые на их основе, называются, соответственно, внешней, концептуальной и внутренней.

Как очевидно конечной целью проектирования является построение конкретной базы данных, в той или иной степени воплощающей представление проектировщика о предметной области и задачах, решаемых пользователями с использованием созданной базы. Рассматривая базу данных как *конкретную реализацию модели*, мы по существу устанавливаем порядок процесса, отделяя этап определения принципов (то, какой база *должна* быть) от этапа воплощения этих принципов при реализации базы данных в конкретной среде СУБД, ОС и языках программирования. И, как показывает практика, между реализациями баз данных и принципами их построения всегда есть расхождения. Различия являются следствием разных причин, но чаще всего - это явный или неявный отказ от некоторых принципиальных ограничений, налагаемых, например, моделью данных или базовыми (встроенными) алгоритмами обработки, в пользу частного решения, которое, по мнению проектировщика, будет более эффективно, например, для понимания или обработки данных.

Важность отделения проектирования на абстрактном уровне от физической реализации состоит в том что, объявляя принципы, мы *конструктивно* ограничиваем область применения. Во-первых, размерность и сложность задачи *должна быть* сокращена до такого уровня, чтобы реализация стала возможной в данных конкретных условиях – ресурсах среды, профессионализме проектировщика, подготовленности пользователя и т.д. Во-вторых, поскольку база данных по определению предназначена для *многофункционального* использования *различными* пользователями, и в тоже время - для обслуживания запросов, *не предвиденных* при проектировании, такое явное объявление принципов позволит не вводить в заблуждение пользователя и не создавать приложения для решения задач, которые в силу своего принципиального отличия от тех, которые рассматривались при проектировании, обусловят неэффективную обработку данных[1]. В-третьих, проектирование – это процесс своеобразного согласования точек зрения двух основных субъектов: пользователя и проектировщика базы данных. Для пользователя характерны требования высокой степени общности и широты представления (и не громоздкость детальных описаний), позволяющих ему получить достаточно сведений без

[1] Применяемые формальные языки представления предметной области не позволяют описывать *все* отношения, которые проектировщик считает важными. С другой стороны, многие проекты (и, в частности, рассматриваемые ***примеры***) воспринимаются как достаточно простые, а проектные решения кажутся очевидными. Кроме того, опытный программист всегда может предложить некоторый эмпирический и, возможно, действительно эффективный способ для целевого представления и обработки нужной информации. Однако это означает отказ от единого формализма, что при увеличении количества данных и связей значительно усложняет проблемы управления базой и в частности – понимание пользователем организации и методов доступа.

затраты значительных временных или интеллектуальных ресурсов. Для администратора, выполняющего проектирование и оптимизацию системы баз данных, необходима высокая степень детализации и формализации, обеспечивающих обоснованность технических решений, а также возможность автоматизации проектирования.

7.2. Типология моделей

Основные отличия любых методов представления информации заключаются в том, каким способом фиксируется семантика предметной области. Но, следует особо отметить, что для всех уровней и для любого метода представления предметной области (но для нас важно, что *в контексте создания и использования машинных баз данных*) в основе отображения (т.е., собственно формирования представления) лежит *кодирование* понятий и отношений между понятиями. Многоуровневая система моделей представления информации иллюстрируется ***слайдами 2, 3, 4 (Типология моделей)***.

Ключевым этапом при разработке любой информационной системы является проведение системного анализа: формализация предметной области и представление системы как совокупности компонент. Системный анализ позволяет, с одной стороны лучше понять «что надо делать» и «кому надо делать» (аналитику, разработчику, руководителю, пользователю), а с другой - отслеживать во времени изменения рассматриваемой модели и обновлять проект.

Декомпозиция, как основа системного анализа, может быть функциональной (построение иерархий функций) или объектной.

Однако в большинстве систем, если говорить, например, о базах данных, типы данных являются более статичным элементом, чем способы их обработки. Поэтому получили интенсивное развитие такие методы системного анализа, как диаграммы массивов данных (Data Flow Diagram). Развитие реляционных баз данных в свою очередь стимулировало развитие методик построения моделей данных, и в частности, ER-диаграмм (Entity Relationship Diagram). Но и функциональная декомпозиция и диаграммы данных дают только некоторый срез исследуемой предметной области, но не позволяют получить представление системы в целом.

Различаются и методы отображения, используемые на этапе построения даталогических моделей, отражающих способ идентификации элементов и связей, но, что особенно важно, в контексте их будущего представления в одномерном пространстве памяти вычислительной машины. Модели подразделяются на фактографические - ориентированные на представление хорошо структурированной информации, и документальные - представляющие наиболее распространенный способ отражения

слабоструктурированной информации. Если в первом случае говорят о реляционной, иерархической или сетевой моделях данных, то во втором — о семантических сетях и документальных моделях. Хотя, разделение на фактографические и документальные в этой группе моделей является достаточно условным. Документ, как последовательность полей может быть представлен, в том числе, и реляционной моделью. И в этом случае выбор специализированного решения чаще всего обуславливается требованием общей эффективности.

При проектировании информационных систем свойства объектов (их характеристики) задаются атрибутами. Именно значения атрибутов позволяют выделить в предметной области как различные объекты (типы объектов), так и среди объектов одного типа — их различные экземпляры. Представление атрибутов удобнее всего моделируется теоретико-множественными отношениями. Отношение наглядно представляется как таблица, где каждая строка — кортеж отношения, а каждый столбец (домен) представляет множество значений атрибута. Список имен атрибутов отношения образует схему отношения, а совокупность схем отношений, используемых для представления БД, в свою очередь образует схему базы данных.

Представление схем БД в виде схем отношений упрощает процедуру проектирования БД. Этим объясняется создание систем, в которых проектирование БД ведется в терминах реляционной модели данных, а работа с БД поддерживается СУБД одного из описанных в данном пособии типов.

Модель данных должна, так или иначе, дать основу для описания данных и манипулирования данными, а также дать средства анализа и синтеза структур данных. Любая модель, построенная более или менее аккуратно с точки зрения математики, сама создает объекты для исследования и начинает жить как бы параллельно с практикой.

Реляционная модель данных в качестве основы отображения непосредственно использует понятие отношения. Она ближе всего находится к так называемой концептуальной модели предметной среды и часто лежит в основе последней.

В отличие от теоретико-графовых моделей в реляционной модели связи между отношениями реализуются неявным образом, для чего используются *ключи отношений*. Например, отношения иерархического типа реализуется механизмом первичных / внешних ключей, когда в подчиненном отношении должен присутствовать набор атрибутов, связывающих это отношение с основным. Такой набор атрибутов в основном отношении будет называться первичным ключом, а в подчиненном – вторичным.

Прогресс в области разработки языков программирования, связанный, в первую очередь с типизацией данных и появлением объектно-ориентированных языков, позволил подойти к анализу сложных систем с точки зрения иерархических представлений - классам объектов со свойствами инкапсуляции, наследования и полиморфизма, схемы которых отображают не только данные и их взаимосвязи, но и методы обработки данных.

В этом смысле объектно-ориентированный подход является гибридным методом и позволяет получить более естественную формализацию системы в целом. В итоге это позволяет снизить существующий барьер между аналитиками и разработчиками (проектировщиками и программистами), повысить надежность системы и упростить сопровождение, в частности, интеграцию с другими системами.

7.3. Этапы проектирования и объекты моделирования

Проектирование базы данных - это упорядоченный формализованный процесс создания *системы взаимосвязанных описаний,* т.е. таких моделей предметной области, которые связывают (фиксируют) хранимые в базе данные с объектами предметной области, описываемыми этими данными. Прикладное назначение таких описаний состоит в том, чтобы пользователь, практически не имеющий представления об организации данных в БД (физическом размещении в памяти данных и механизмах их поиска), обращая запрос к базе, имел бы практическую возможность получить адекватную информацию о состоянии объекта предметной области. ***(Слайд 5 - Стадии и объекты)***

Проектирование начинается с анализа предметной области и выявления функциональных и других требований к проектируемой системе. Подробнее этот процесс будет рассмотрен ниже, а здесь отметим, что проектирование обычно выполняется человеком (группой людей) – системным аналитиком (а на практике чаще администратором базы данных), которым может быть как специально выделенный сотрудник, так и будущий пользователь базы данных, достаточно хорошо знакомый с машинной обработкой данных.

Объединяя отдельные представления о содержимом базы данных, полученные в результате опроса пользователей, и свои представления о данных, которые могут потребоваться для решения практических задач, системный аналитик сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, выполненное с использованием естественного языка, математических выражений, таблиц,

графов и других средств, понятных всем людям, работающим над проектированием базы данных, называют *инфологической моделью.*

Такая человеко-ориентированная модель практически полностью независима от физических параметров среды хранения данных, которой может быть как память человека, так и ЭВМ. Поэтому инфологическая модель не изменяется до тех пор, пока какие-то изменения в реальном мире (той его части, которая отнесена к предметной области) не потребуют изменения в модели соответствующего фрагмента описания, чтобы эта модель продолжала адекватно отражать предметную область.

Остальные модели являются машинно-ориентированными. С их помощью СУБД дает возможность программам и пользователям осуществлять доступ к хранимым данным лишь по их именам, не заботясь о физическом расположении этих данных.

Так как доступ к данным осуществляется с помощью конкретной СУБД, то модели должны быть представлены на языке описания данных этой СУБД. Такое описание, создаваемое по инфологической модели данных, называют *даталогической моделью* данных.

Для размещения и поиска данных на внешних запоминающих устройствах СУБД использует *физическую модель* данных.

Представленная трехуровневая архитектура (инфологический, даталогический и физический уровни) позволяет обеспечить независимость хранимых данных от использующих их программ. Хранимые данные могут быть переписаны на другие носители или может быть реорганизована их физическая структура, в том числе дополнена полями для новых приложений, но это повлечет лишь изменение физической и, возможно, даталогической модели данных. Главное, такие изменения физической и даталогической моделей не будут замечены пользователями системы (окажутся "прозрачными" для них). Кроме того, независимость данных обеспечивает возможность создания новых приложений для решения новых задач без разрушения существующих.

Приведенная цитата (***Слайд 6***) по-прежнему актуальна, хотя книга издана более 20 лет назад. Действительно, средства проектирования непрерывно развиваются, но и задачи, решение которых пользователь предполагает автоматизировать с помощью систем баз данных, существенно усложнились и для эффективного применения средств формализации и автоматизации необходимо понимать природу системы моделей.

С точки зрения объектов моделирования необходимо различать модели предметной области и модели базы данных. Эти модели взаимосвязаны, поскольку представляют собой образы одного и того же оригинала – некоторого множества предметов реального мира, информацию о которых мы предполагаем хранить и обрабатывать с помощью проектируемой БД. Характер взаимосвязей (и, соответственно, отличий) проявляется и в процессе проектирования системы баз данных. Модель предметной области скорее ассоциируется с неформальным[2] уровнем семантического моделирования, а модель базы данных – с формализованным уровнем системы (и в частности, СУБД).

Разнообразие моделей связано также и с различием используемых парадигм моделирования, по существу определяющих способ *представления* взаимосвязи объектов на уровне *структур данных*. С этой точки зрения, различаются реляционные, сетевые, иерархические, объектные, объектно-реляционные, документальные и другие виды моделей. Соответственно различаются и описываемые их средствами схемы баз данных.

12.1. Основные понятия реляционной модели данных

Как было показано в предыдущей лекции, для определения реляционной модели данных необходимо объявить структуру данных, способ манипулирования ими и ограничения целостности (***слайд 2***).

12.1.1. Структурный компонент реляционной модели

С точки зрения структуры данных реляционная модель является удобной и наиболее привычной формой представления данных в виде таблицы. Понятию «таблица» соответствует понятие «отношение» (relation). Отсюда и произошло название модели – реляционная. Т.е., применительно к базам данных понятия «реляционная БД» и «табличная БД» по существу являются синонимами. В отличие от иерархической и сетевой модели, такой способ представления

1) понятен пользователю-непрограммисту;

2) позволяет легко изменять схему – присоединять новые элементы данных и записи без изменения соответствующих подсхем;

3) обеспечивает необходимую гибкость при обработке непредвиденных запросов.

К тому же любая сетевая или иерархическая схема может быть представлена двумерными отношениями.

---

[2] Правильнее было бы говорить о *неформализованности*, связанной с невозможностью обоснованного *однозначного* выбора (из реально существующих) объектов средств, используемых для моделирования.

Одним из основных преимуществ реляционной модели является ее однородность. Все данные рассматриваются как хранимые в таблицах, в которых каждая строка имеет один и тот же формат. Каждая строка в таблице представляет некоторый объект реального мира или соотношение между объектами. Пользователь модели сам должен для себя решить вопрос, обладают ли соответствующие сущности реального мира однородностью. Этим самым решается проблема пригодности модели для предполагаемого применения.

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие: *домен, отношение, кортеж, кардинальность, атрибуты, степень, первичный ключ.* Соотношение понятий иллюстрируется на слайде (***слайд 3***).

Домен – это совокупность значений, из которой берутся значения соответствующих атрибутов определенного отношения. С точки зрения программирования домен – это тип данных, определяемый системой (стандартный) или пользователем.

Первичный ключ – это столбец или некоторое подмножество столбцов, которые уникально, т.е. единственным образом определяют строки. Первичный ключ, который включает более одного столбца, называется множественным, или комбинированным, или составным. Правило целостности объектов утверждает, что первичный ключ не может быть полностью или частично пустым, т.е. иметь значение null.

Остальные ключи, которые можно также использовать в качестве первичных, называются потенциальными или *альтернативными* ключами.

Сформулируем правила назначения первичных ключей сущностей:

1). Первичный ключ должен *однозначно идентифицировать* любой экземпляр сущности.

2). По возможности первичный ключ должен быть *наиболее компактным* из всех потенциальных ключей, наилучший тип данных для первичного ключа — целочисленный.

3). Первичный ключ может быть составным, но увеличение количества столбцов, входящих в него, противоречит требованию компактности. Требование компактности также не удастся выполнить, если, например, в качестве первичного ключа выбрать атрибут строкового типа данных большой длины.

4). Значения первичного ключа *не должны подвергаться частым модификациям.* Идеально, если бизнес-логика предметной области такова, что эти значения вообще не предполагается изменять.

5). *Правила модификации первичного ключа должны контролироваться внутренней бизнес-логикой* предметной области, а не решениями, которые принимаются

над ней. Например, в базе данных, разрабатываемой для нужд деканата, для сущности СТУДЕНТ не стоит выбирать в качестве первичного ключа серию и номер паспорта студента. Хотя эти данные в принципе и обладают свойствами обязательности и уникальности, но их изменение может быть инициировано студеном, а не администрацией факультета.

5). Если среди информации, собранной о сущности, не удается выделить данные, которые удовлетворяют перечисленным выше требованиям, то рекомендуется рассмотреть возможность создания *суррогатного первичного ключа,* который, не неся никакой семантической нагрузки, просто служит идентификатором конкретного экземпляра сущности. Обычно в качестве суррогатного первичного ключа выбираются всевозможные коды или идентификаторы. Суррогатный ключ чаще всего скрыт на внешнем уровне моделирования реляционной базы данных.

Внешний ключ – это столбец или подмножество одной таблицы, который может служить в качестве первичного ключа для другой таблицы. *Внешний ключ* таблицы является ссылкой на первичный ключ другой таблицы. Правило ссылочной целостности гласит, что внешний ключ может быть либо пустым, либо соответствовать значению первичного ключа, на который он ссылается. Внешние ключи являются неотъемлемой частью реляционной модели, поскольку реализуют связи между таблицами базы данных.

Внешний ключ, как и первичный ключ, тоже может представлять собой комбинацию столбцов. На практике внешний ключ всегда будет составным (состоящим из нескольких столбцов), если он ссылается на составной первичный ключ в другой таблице. Очевидно, что количество столбцов и их типы данных в первичном и внешнем ключах совпадают.

Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей.

Понятия реляционной модели представляют специальную терминологию, введенную авторами теоретических основ, однако они имеют и более привычные аналоги (но не во всем эквиваленты!), соответствие которых приведено в следующей таблице (***слайд 4***).

12.1.2. Управляющий компонент реляционной модели

Множество допустимых операций над данными, представленными в виде совокупности отношений задается реляционной алгеброй. Кроме реляционных операций манипулирования данными в состав управляющего компонента должны входить

определение данных; определение представлений; условия целостности; идентификация прав доступа; границы транзакций (начало, завершение и отмена).

12.1.3. Целостность данных (*слайд 5*)

Целостность на уровне доменов

В реляционной теории принято считать, что все значения атрибутов отношения атомарны. Это следует из трактовки понятия домена. Домен можно рассматривать как подмножество значений некоторого типа данных, имеющих определенный смысл. Реляционная модель требует, чтобы типы используемых данных были простыми (скалярными), т. е. не обладающими внутренней структурой.

Домен имеет уникальное имя в пределах базы данных, определен на простом типе данных или на другом домене. Собственно для реляционной модели данных тип используемых данных не важен. Требование того, чтобы тип данных был *простым,* нужно понимать так, что *в реляционных операциях не должна учитываться внутренняя структура данных.*

Основное назначение доменов состоит в том, что они *ограничивают сравнения.* Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип. Таким образом, понятие домена помогает правильно *моделировать* предметную область.

Целостность на уровне отношений

Потенциальные ключи служат единственным *средством адресации па уровне кортежей* в отношении. Только знание значения потенциального ключа кортежа позволяет точно указать этот кортеж.

С точки зрения семантического моделирования данных, потенциальные ключи служат *средством идентификации* объектов предметной области — экземпляров сущностей, данные о которых хранятся в отношении. Поскольку эти экземпляры должны быть различимы по определению, их идентификаторы не могут содержать неизвестные значения.

Обычно для ситуации наличия неизвестных или неполных данных используются типы данных, пополненные так называемым *NULL-знпчением.*

NULL-значение — это некий указатель на то, что значение неизвестно. Проблема использования NULL-значения в теории реляционных баз данных окончательно не решена. Практически все реализации современных реляционных СУБД позволяют

использовать NULL-значения, несмотря на их недостаточную теоретическую обоснованность.

Правило целостности отношений гласит: каждое отношение должно иметь по крайней мере один потенциальный ключ, входящие в состав которого атрибуты не могут принимать NULL-значений. Этот потенциальный ключ лучше всего объявлять первичным ключом таблицы, соответствующей данному отношению.

Следует отметить, что большинство СУБД вполне позволяют создавать таблицы и без первичных ключей. Однако нарушение правила целостности отношений на практике сразу дает о себе знать. Например, для СУБД MS SQL-сервер станет невозможным доступ к данным по технологии OLE DB Provider.

13.1. Постановка задачи

Задача проектирования БД для предметной области состоит в том, чтобы обеспечить поддержку не только любых ныне используемых, но и будущих приложений. Таким образом, БД создают основу для обработки неформализованных, изменяющихся и неизвестных запросов и создания приложений, для которых невозможно заранее определить требования к данным. Это позволяет в дальнейшем строить на основе предметных БД достаточно стабильные информационные системы, т.е. системы, в которых большинство изменений можно осуществить без переписывания старых приложений.

С другой стороны, основывая проектирование БД на реализации текущих и видимых задач, можно существенно ускорить создание информационной системы, структура которой учитывает наиболее часто встречающиеся пути доступа к данным. Однако по мере количества таких информационных систем быстро увеличивается число прикладных БД и, соответственно, резко возрастает уровень дублирования данных и повышается стоимость их ведения.

Желание достичь одновременно гибкости и эффективности приводит к тому, что в общем случае предметный подход используется для построения первоначальной информационной структуры, а прикладной – для ее совершенствования с целью повышения эффективности обработки данных.

При проектировании информационной системы необходимо провести анализ целей этой системы и выявить требования к ней отдельных пользователей. Сбор данных начинается с выявления и изучения объектов информационной среды и процессов, в которых эти объекты участвуют. Объекты (сущности) группируются по типу и по мощности связей между ними (студент – сессия, преподаватель – дисциплина и т.д.).

Дальнейшая задача 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. Такой проект БД можно создать, используя методологию нормализации отношений.

Рассмотрим следующую задачу: пусть необходимо обеспечить сбор и обработку данных по результатам сдачи экзаменов и зачетов студентами факультета. Организация данных должна обеспечивать (***слайд 2***):

выполнение текущего учебного плана;

формирование ведомостей по отдельным дисциплинам для групп студентов;

формирование листов зачетных книжек студентов;

формирование сводной ведомости курса;

расчет среднего балла по дисциплинам и т.п.

13.2. Восходящее проектирование (универсальное отношение)

Проектирование БД на основе описания предметной области в виде сводной таблицы (технология восходящего проектирования) предполагает выявление необходимого набора атрибутов – характеристических свойств объектов таким образом, чтобы каждый из этих атрибутов имел в базе данных уникальное имя, и представление этих атрибутов в виде двумерной таблицы. Перечислим набор атрибутов, необходимый для обеспечения перечисленных в постановке задачи функций, в виде универсального отношения (***слайд 3***):

Сессия (*ФИО студента, № зачетной книжки, Дисциплина, Семестр, Форма отчетности, Количество часов, Оценка, Дата сдачи, ФИО преподавателя, Должность преподавателя, Кафедра*).

Применим правила нормализации к универсальному отношению «Сессия» (***слайд 4***).

1. *Определение первичного ключа таблицы*.

Предположим, что каждый студент сдает один раз экзамен (зачет) по дисциплине учебного плана и получает оценку. Дисциплина учебного плана однозначно характеризуется наименованием, номером семестра, за который отчитывается студент, и формой отчетности (т.к. учебный план предусматривает сдачу и экзамена, и зачета по

одной и той же дисциплине в рамках одного семестра). Тогда в качестве первичного ключа отношения «Сессия» можно использовать следующий набор атрибутов:

№ зачетной книжки, *Дисциплина, Семестр, Форма отчетности*.

2. *Выявление атрибутов, функционально зависящих от части составного ключа*.

Каждый из атрибутов - *ФИО преподавателя*, *Должность преподавателя, Кафедра* и *Количество часов* - функционально зависит только от атрибутов *Дисциплина*, *Семестр* и *Форма отчетности*, т.е. этот атрибут вместе с совокупностью атрибутов первичного ключа составит вторую таблицу:

Учебный план (*Дисциплина, Семестр, Форма отчетности, Количество часов, ФИО преподавателя, Должность преподавателя, Кафедра*).

Из исходной таблицы при этом удаляются атрибуты *ФИО преподавателя*, *Должность преподавателя, Кафедра* и *Количество часов*:

Сводная ведомость (*ФИО студента, № зачетной книжки, Дисциплина, Семестр, Форма отчетности, Оценка*).

Составной первичный ключ, повторяющийся в обеих таблицах, приводит к избыточности при дублировании информации сразу трех столбцов, поэтому кажется целесообразным ввести дополнительный атрибут - № (порядковый номер) – в таблицу «Учебный план» и использовать именно его в качестве первичного ключа. Тогда таблицы примут следующий вид:

Учебный план *(№ Уч. план, Дисциплина, Семестр, Форма отчетности, Кол-во часов, ФИО преподавателя, Должность преподавателя, Кафедра*).

Сводная ведомость (*ФИО студента, № зачетной книжки, № Уч. план, Оценка*).

В таблице «Сводная ведомость» осталась еще одна частичная функциональная зависимость: *№ зачетной книжки →ФИО студента.* Ликвидировав эту зависимость, получим еще одну таблицу – «Студенты»:

Студенты (*№ зачетной книжки, ФИО студента*)

Ликвидация ФЗ приведет к изменению таблицы «Результаты сессии»:

Сводная ведомость (*№ зачетной книжки, № Уч. план, Оценка*)

3. *Выявление транзитивных зависимостей*

В таблице «Учебный план» выявляются следующие транзитивные зависимости:

№ Уч. план → ФИО преподавателя

ФИО преподавателя → Должность преподавателя

ФИО преподавателя → Кафедра

Применив второй шаг процедуры нормализации, получим таблицу «Преподаватели»:

Кадровый состав (*ФИО преподавателя, Должность преподавателя, Кафедра*)

Следует иметь в виду, что в этом случае в рамках ПрО считается, что атрибут *ФИО преподавателя* однозначно (уникально) характеризует конкретного преподавателя.

Итак, получили следующую декомпозицию (***слайд 5***):

Учебный план *(№ Уч. план, Дисциплина, Семестр, Форма отчетности, Кол-во часов, ФИО преподавателя*)

Студенты (*№ зачетной книжки, ФИО студента*)

Кадровый состав (*ФИО преподавателя, Должность преподавателя, Кафедра*)

Сводная ведомость (*№ зачетной книжки, № Уч. план, Оценка*)

Такой декомпозиции на самом деле достаточно для того, чтобы преобразовать исходную таблицу к совокупности нормализованных таблиц (все полученные таблицы приведены к ЗНФ и к НФБК).

13.2. Нисходящее проектирование

13.2.1. Построение инфологической модели

Представим предметную область как взаимодействие двух сущностей - «Дисциплина учебного плана» и «Студент»: каждый студент сдает экзамен или зачет по некоторой дисциплине учебного плана и получает оценку, которая должна быть зафиксирована в модели данных (***слайд 6***).

«Дисциплина учебного плана» с точки зрения решаемой задачи должна быть представлена группой свойств, позволяющих характеризовать дисциплину в рамках каждого отдельного семестра: наименование дисциплины, семестр, количество часов, форма отчетности (экзамен или зачет) и данные о преподавателе, читающем дисциплину. Необходимость задания таких свойств обусловлена, с одной стороны, задачей организации хранения результатов сдачи экзаменов и зачетов (наименование дисциплины, семестр и форма отчетности), и с другой стороны – задачей формирования листов зачетных книжек (количество часов и данные о преподавателе). Отдельный экземпляр такой сущности однозначно идентифицируется тройкой свойств – наименование дисциплины, семестр и форма отчетности.

Сущность «Студент» для обеспечения выполнения объявленных функций должна характеризоваться следующими свойствами: фамилия, имя, отчество и номер группы. Однако следует отметить, что даже набор значений всех этих свойств не может однозначно характеризовать экземпляр сущности, т.к. можно предполагать наличие в одной группе полных однофамильцев. Таким образом, для идентификации отдельного

экземпляра сущности необходимо ввести дополнительное (ключевое) свойство – идентификационный номер студента (например, номер зачетной книжки).

Определим для сущности «Студент» еще два дополнительных свойства, которые не будут непосредственно обеспечивать решение поставленной задачи, но могут служить для реализации дополнительных (сервисных) функций (например, организации почтовой или телефонной связи): домашний адрес и номер телефона. Свойство «Домашний адрес», являясь по сути составным, будет на самом деле рассматриваться в контексте решаемых задач как простое, а свойство «Номер телефона» - как условное.

Взаимодействие сущностей реализуется связью «Сводная ведомость», т.е. Студент сдает экзамен (зачет) по Дисциплине учебного плана. Мощность связи – «многие ко многим» (M:M). Для идентификации связи отдельных экземпляров сущностей в этом случае необходимо наличие у связи следующих дополнительных свойств: оценка и дата сдачи экзамена (зачета).

ER-диаграмма рассматриваемой задачи представлена на слайде (***слайд***).

Построенная ER-диаграмма находится в *первой нормальной форме*, т.к. сущности не имеют повторяющихся групп свойств. Однако при рассмотрении свойств сущности «Дисциплина учебного плана» можно заметить, что свойство «Преподаватель» зависит только от части ключевых свойств – а именно от свойств «Наименование дисциплины» и, возможно, «Форма отчетности». Следовательно, для того, чтобы привести ER-диаграмму ко второй нормальной форме, необходимо выделить свойство «Преподаватель» в отдельную сущность (***слайд 7***).

Новая сущность «Преподаватель» характеризуется группой основных свойств - фамилия, имя, отчество, и группой дополнительных свойств – кафедра, должность, домашний адрес и телефон. Так же, как и для сущности «Студент», для сущности «Преподаватель» необходимо ввести дополнительное (ключевое) свойство – идентификационный номер преподавателя.

Взаимодействие новой сущности с сущностью «Дисциплина учебного плана» осуществляется посредством новой связи «Читает». Мощность связи – «Многие к одному» (M:1), т.е. несколько дисциплин учебного плана может читать один преподаватель.

Измененная ER-диаграмма представлена на слайде. Новый вариант ER-диаграммы находится *в третьей нормальной форме*, т.к. сущности не имеют свойств, зависящих от не ключевых.

13.2.2. Построение реляционной схемы

Следующий этап проектирования – построение даталогической модели. В рассматриваемом случае задача этого этапа – преобразование ER-диаграммы в реляционную схему.

Реляционный подход, в основе которого лежит принцип разделения данных и связей, обеспечивает с одной стороны независимость данных, а с другой – более простые способы хранения и обновления.

Первые шаги преобразования состоят в превращении каждой сущности в отношение (таблицу). Связь типа M:M, которую называют «сущность-связь», тоже превращается в отдельное отношение. Каждое свойство становится атрибутом - столбцом соответствующей таблицы.

После реализации этих шагов получаем реляционную схему, изображенную на слайде (***слайд 8***), где представлены таблицы «Студенты», «Сводная ведомость», «Учебный план» и «Кадровый состав», отображающие соответственно сущности «Студент», «Сводная ведомость», «Дисциплина учебного плана» и «Преподаватель».

Далее необходимо преобразовать связи во внешние ключи. Связь «многие ко многим», реализуемая отношением «Сводная ведомость», должна содержать уникальные идентификаторы сущностей – участников связи. При этом, если для однозначной идентификации студента достаточно добавить в таблицу столбец *ID_Студент*, то однозначная идентификация дисциплины потребует добавления в таблицу столбцов *Наименование, Семестр* и *Форма_отчетности*. Хранение всей этой информации явно приведет к *избыточности* данных и их потенциальной *противоречивости* (например, если при переносе дисциплины на другой семестр обновить только строку таблицы «Учебный план», то содержимое таблицы «Сводная ведомость» станет не актуальным).

Для ликвидации избыточности и потенциальной противоречивости данных добавим в таблицу «Учебный план» столбец *ID_План,* содержимое которого будет однозначно идентифицировать каждую строку таблицы. Теперь этот новый столбец станет первичным ключом, и одноименный столбец должен быть добавлен в таблицу «Сводная ведомость».

Связь «Читает» предполагает добавление в таблицу «Учебный план» столбца *ID_Преподаватель*. Реляционная схема со связями представлена на слайде (***слайд 9***).

13.2.3. Нормализация таблиц

Все построенные таблицы находятся в *первой нормальной форме,* т.к. каждый столбец таблицы неделим и в рамках одной таблицы нет столбцов с одинаковыми по смыслу значениями.

Таблица «Сводная ведомость» через столбцы *ID_Студент* и *ID_План* связывает информацию о студенте с информацией о конкретной дисциплине и фиксирует оценку, полученную студентом. Оценка и дата сдачи экзамена (зачета) однозначно зависят от содержимого столбцов *ID_Студент* и *ID_План,* которые представляют собой составной первичный ключ. Таким образом, все таблицы имеют первичные ключи, которые однозначно определяют строки и не избыточны, и можно говорить о том, что таблицы находятся *во второй нормальной форме*.

Рассмотрим подробнее таблицу «Учебный_план», которая содержит перечень дисциплин текущего учебного плана. Первичным ключом таблицы служит столбец *ID_План*, который однозначно характеризует каждую дисциплину учебного плана с точностью до семестра, т.е. для дисциплин, протяженность изучения которых более одного семестра, в таблице будет отведено столько строк, сколько семестров длится изучение дисциплины. Тогда хранение наименований дисциплин в таблице «Учебный_план» становится избыточным: например, если изучение английского языка длится 6 семестров, то наименование «Английский язык» будет повторено в 6 записях и есть вероятность сделать 6 различных ошибок при вводе одного и того же наименования.

Чтобы избежать этого, проведем декомпозицию отношения «Учебный план», выделив наименования дисциплин в отдельное отношение. В результате получим дополнительную таблицу «Дисциплины» со столбцами *ID_Дисциплина* и *Наименование*, а столбец *Наименование* в таблице «Учебный_план» заменим столбцом *ID_Дисциплина,* сформировав тем самым вторичный ключ, связывающий новую таблицу с таблицей «Учебный_план» (***слайд 10***).

13.2.4. Физическая модель

Теперь можно говорить о базе данных «Сессия», реляционная схема которой представлена следующими пятью таблицами:

«Студенты» - содержит по одной строке для каждого из студентов;

«Учебный_план» - содержит по одной строке для отдельной дисциплины отдельного семестра;

«Дисциплины» - содержит по одной строке для наименования дисциплины;

«Сводная_ведомость» - содержит по одной строке для каждого результата сдачи отдельным студентом отдельной дисциплины;

«Кадровый_состав» - содержит по одной строке для каждого из преподавателей.

На слайде в графической форме изображены перечисленные таблицы, их столбцы, первичные и внешние ключи. Задание первичных и внешних ключей сопровождается построением дополнительных структур – индексов, обеспечивающих быстрый доступ к данным через значение ключа.

Все таблицы базы данных «Сессия» находятся в третьей нормальной форме:

каждый столбец таблицы неделим и в рамках одной таблицы нет столбцов с одинаковыми по смыслу значениями (1НФ);

первичные ключи однозначно определяют запись и не избыточны, все поля каждой из таблиц зависят от ее первичного ключа (2НФ);

значение любого поля, не входящего в первичный ключ, не зависит от значения другого поля, тоже не входящего в первичный ключ (3НФ).

Следующий этап проектирования – определение доменов (типов) данных, хранящихся в столбцах таблиц. Параллельно с заданием типа необходимо сформулировать ограничения целостности, связанные с типом - перечень допустимых значений типа.

Исходя из особенностей данных и их функционального назначения, требуется задать способ представления и границы возможных изменений для каждого из столбцов таблиц. При этом необходимо ответить на вопрос, данные каких типов должны храниться в столбцах и какова их максимальная длина (например, если в столбце предполагается хранить процентные значения, то достаточно будет целого типа данных длиной 1 байт, так как диапазон возможных значений от 0 до 255; если для данных столбца выбирается тип «строка символов», то желательно указать максимальный размер данных столбца и т.п.).

Далее, в каждой таблице должны быть выделены столбцы, которые *обязательно должны быть заполнены* при создании отдельной строки таблицы. Задание такого ограничения целостности не позволит, например, ввести в таблицу «Студенты» строку, в которой не указан номер группы. Если подобные ограничения целостности не будут заданы, в таблице могут появиться строки, которые не будут учтены при выполнении функций по обработке данных: появление в таблице «Студенты» строки без номера группы приведет к ошибке при формировании ведомости.

Следующий важный момент - задание для столбцов значений по умолчанию. Значение по умолчанию впоследствии будет автоматически вводиться в указанный столбец для каждой строки таблицы. Например, в столбец *Дата_сдачи* таблицы «Сводная

ведомость» при заполнении очередной строки может автоматически заноситься текущая дата.

На слайде (***слайд 11***) представлены таблицы базы данных «Сессия» с типами данных столбцов и предлагаемыми ограничениями целостности.