МИЭТ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

В. М. Илюшечкин

ОСНОВЫ ИСПОЛЬЗОВАНИЯ И ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

УЧЕБНИК ДЛЯ АКАДЕМИЧЕСКОГО БАКАЛАВРИАТА

Рекомендовано Учебно–методическим отделом высшего образования в качестве учебника для студентов высших учебных заведений, обучающихся по инженерно–техническим направлениям и специальностям

Допущено Учебно–методическим объединением вузов по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению «Информатика и вычислительная техника»

Москва ▪ Юрайт ▪ 2015

УДК 004.65(075.8)
ББК 32.973-018.2я73
И43

Настоящая методическая разработка выполнена в рамках инновационной образовательной программы МИЭТ «Современное профессиональное образование для российской инновационной системы в области электроники».

***Автор*:**

Илюшечкин Владимир Михайлович — кандидат технических наук, доцент кафедры информатики и программного обеспечения вычислительных систем факультета микроприборов и технической кибернетики Национального исследовательского университета «МИЭТ».

Илюшечкин, В. М.

И43 Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт ; ИД Юрайт, 2015. — 213 с. — Серия : Бакалавр. Академический курс.

ISBN 978-5-9916-4705-2 (Издательство Юрайт)
ISBN 978-5-9692-1573-3 (ИД Юрайт)

В учебнике содержатся теоретические и практические сведения о современных системах управления базами данных (СУБД), об использовании и проектировании баз данных. Рассматриваются языковые и программные средства СУБД и систем автоматизации проектирования баз данных. Приведены примеры создания инфологических и даталогических моделей, позволяющие студентам научиться проектировать базы данных.

Соответствует актуальным требованиям Федерального государственного образовательного стандарта высшего образования.

Для студентов, обучающихся по направлению «Информатика и вычислительная техника».

УДК 004.65(075.8)
ББК 32.973-018.2я73

ISBN 978-5-9916-4705-2
(Издательство Юрайт)
ISBN 978-5-9692-1573-3
(ИД Юрайт)

Оглавление

Принятые сокращения

БД — база данных
БнД — банк данных
ДЛМ — даталогическая модель базы данных
ИЛМ — инфологическая модель предметной области
ИМ — иерархическая модель данных
НФ — нормальная форма отношения
НФБК — нормальная форма Бойса — Кодда
ОЦ — ограничения целостности
ПО — предметная область
РБД — реляционная база данных
РМ — реляционная модель данных
СМ — сетевая модель данных
СП — сущность-потомок
СР — сущность-родитель
СУБД — система управления базами данных
ФЗ — функциональная зависимость
ФМ — физическая модель базы данных
ЯОД — язык описания данных
3GL — Third-Generation Language (язык третьего поколения)
4GL — Fourth-Generation Language (язык четвертого поколения)
ANSI — American National Standards Institute (Американский институт национальных стандартов)
API — Application Programming Interface (интерфейс прикладного программирования)
ASP — Active Server Pages
BDE — машина баз данных Borland Database Engine
CASE — Computer Aided Software Engineering
CGI — Common Gateway Interface
DDL — Data Definition Language (язык определения данных)
DFD — Data Flow Diagram
DML — Data Manipulation Language (язык манипулирования данными)

DQL — Data Query Language (язык запросов данных)
ER — Entity-Relationship (сущность-связь)
GUID — Globally Unique Identifier (уникальный идентификационный номер)
HTML — Hypertext Markup Language (язык разметки гипертекста)
HTTP — Hypertext Transfer Protocol (сетевой протокол передачи гипертекста)
ICAM — Integrated Computer Aided Manufacturing (интегрированная компьютеризация производства)
IDEF — методология ICAM DEFinition
IE — методология Information Engineering
IIS — Web-сервер Microsoft Internet Information Services
ISO — International Organization for Standardization (Международная организация по стандартам)
MSDE — машина баз данных Microsoft Data Engine
OLE — Object Linking and Embedding (технология связывания и внедрения объектов и протокол разработанные компанией «Майкрософт»)
OLTP — Online Transaction Processing (оперативная обработка транзакций)
PHP — Hypertext Preprocessor (Препроцессор Гипертекста — скриптовый язык программирования)
QBE — Query-By-Example (язык запросов по образцу)
SADT — Structured Analysis and Design Technique (метод структурного анализа и проектирования)
SGML — Standard Generalized Markup Language (стандартный общий язык разметки)
SPARC — Standards Planning and Requirements Committee (подкомитет Американского института национальных стандартов)
SQL — Structured Query Language (структурированный язык запросов)
UML — Unified Modeling Language (унифицированный язык моделирования)
URL — Uniform Resource Locator (определитель местонахождения информационного ресурса)
WWW — World Wide Web (Всемирная паутина)
XML — eXtensible Markup Language (расширяемый язык разметки)

Предисловие

Потребность в информации стала одной из самых насущных в жизни современного цивилизованного человечества. Включая утром радиоприемники и телевизоры, люди с нетерпением ждут новостей о погоде, курсах валют, сообщений о показателях деловой активности и т.д. Приходя на работу, они погружаются в море деловой информации, которую получают через свои служебные компьютеры. Возвращаясь домой, они заходят в супермаркеты за покупками и из кассовых чеков узнают информацию о цене приобретенных товаров.

Источником значительной части информации являются базы данных, в которых содержатся сведения о прогнозах погоды, ежедневных курсах валют, показателях произведенной продукции, ценах и количестве продаваемых товаров, расписаниях движения поездов, отправлении и прибытии самолетов и т.д. Структура баз данных может мало интересовать конечного пользователя, поскольку для него более важны время получения информации, ее актуальность и доступный объем. Создание базы данных, отвечающей всем требованиям пользователей, становится задачей проектировщика, который должен обладать теоретическими знаниями и практическими навыками в области информационных технологий.

Согласно образовательным стандартам высшего профессионального образования подготовка специалиста по направлению «Информатика и вычислительная техника» предусматривает изучение дисциплины «Базы данных», по другим техническим направлениям раздел, посвященный базам данных, включен в общий курс информатики.

Предметом дисциплины «Базы данных» являются база данных как форма организации информационного ядра любой информационной системы, а также языковые и программные средства для работы с базами данных и методы проектирования баз данных. В учебном пособии содержатся основные сведения по этим темам, изложенные в шести главах.

В гл. 1 рассматриваются различные подходы к хранению данных, классификация и архитектура баз данных и систем

управления базами данных, а также дается общее описание процесса проектирования баз данных.

Глава 2 посвящена реляционной модели данных, которая служит основой большинства современных баз данных.

В гл. 3 представлены языки, предназначенные для работы с базами данных. Подробно описывается язык SQL, являющийся общепринятым языком взаимодействия с базами данных. Менее детально изложены возможности языка QBE, относящегося к табличных языкам запросов.

В гл. 4 содержатся сведения о таких широко используемых системах управления базами данных, как Access, Oracle, SQL Server и InterBase, с указанием их технических характеристик и поддерживаемых типов данных.

Главы 5 и 6 посвящены вопросам проектирования баз данных. В гл. 5 излагается метод, основанный на принципах нормализации, и приводится пример использования этого метода для получения базы данных, соответствующей нормальной форме Бойса — Кодда.

В гл. 6 представлен инженерный подход к разработке базы данных, включающий этапы концептуального и даталогического проектирования, поддерживаемые существующими системами автоматизации проектирования баз данных, ряду которых дается краткая характеристика. В качестве примера в этой главе описаны приемы работы с одной из таких систем — ERwin.

Подходы, применяемые при использовании и построении баз данных, универсальны и мало зависят от предметной области, информация о которой хранится в базах данных. Чтобы рассматриваемые в пособии примеры были понятны студентам, выбраны предметные области, связанные с повседневной деятельностью людей.

Глава 1

ОСНОВНЫЕ СВЕДЕНИЯ О ХРАНЕНИИ ДАННЫХ

1.1. Файловые системы хранения данных

Использование компьютерной техники связано с двумя большими областями деятельности человечества [10]. В первой области компьютерная техника применяется для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную. Характерной особенностью данной области является наличие сложных алгоритмов обработки, которые применяются к простым по структуре данным, объем которых сравнительно невелик.

Вторая область включает в себя автоматические или автоматизированные информационные системы, представляющие собой программно-аппаратный комплексы, обеспечивающие надежное хранение информации в памяти компьютера, выполнение специфических для данного приложения преобразований информации и вычислений, предоставление пользователям удобного и легко осваиваемого интерфейса.

Обычно такие системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов и т.д.

Вторая область использования вычислительной техники возникла несколько позже первой. Это связано с тем, что в начальный период развития компьютеров их возможности по хранению информации были ограниченными. Поэтому важным шагом в развитии информационных систем явился переход к использованию централизованных систем управле-

ния файлами. С точки зрения прикладной программы файл — это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные. Правила именования файлов, способ доступа к данным, хранящимся в файле, и структура этих данных зависят от конкретной системы управления файлами и от типа файла. Система управления файлами обеспечивает распределение внешней памяти, отображение имен файлов в соответствующие адреса во внешней памяти и обеспечение доступа к данным.

Для пользователя файл представляется как линейная последовательность записей, с которой можно выполнять ряд стандартных операций:

— создать файл (требуемого типа и размера);
— открыть ранее созданный файл;
— прочитать из файла некоторую запись (текущую, следующую, предыдущую, первую, последнюю);
— записать в файл на место текущей записи новую;
— добавить новую запись в конец файла.

Структура записи файла была известна только программе, которая с ним работала, но не системе управления файлами, и поэтому для того, чтобы извлечь некоторую информацию из файла, необходимо было точно знать структуру записи файла. Каждая программа, работающая с файлом, должна была иметь у себя внутри структуру данных, соответствующую структуре этого файла. При изменении структуры файла требовалось изменять структуру программы, а это требовало новой компиляции, то есть процесса перевода программы в исполняемые машинные команды. Такая ситуации характеризовалась как зависимость программ от данных. Для информационных систем характерным является наличие большого числа различных пользователей (программ), каждый из которых имеет свои специфические алгоритмы обработки информации, хранящейся в одних и тех же файлах. Изменение структуры файла, которое было необходимо для одной программы, требовало исправления и перекомпиляции и дополнительной отладки всех остальных программ, работающих с этим же файлом. Это было первым существенным недостатком файловых систем, который явился толчком к созданию новых систем хранения и управления информацией.

Поскольку файловые системы являются общим хранилищем файлов, принадлежащих разным пользователям, системы управления файлами должны обеспечивать авторизацию доступа к файлам, чтобы по отношению к каждому заре-

гистрированному пользователю данной информационной системы для каждого существующего файла указывались действия, которые разрешены или запрещены конкретному пользователю. Администрирование режимом доступа к файлу в основном выполнялось его создателем — владельцем. Для множества файлов, отражающих информационную модель одной предметной области, такой децентрализованный принцип управления доступом вызывал дополнительные трудности. Отсутствие централизованных методов управления доступом к информации послужило одной из причин разработки систем управления базами данных (СУБД).

Следующей причиной, способствовавшей созданию СУБД, стала необходимость обеспечения эффективной параллельной работы многих пользователей с одними и теми же файлами. В общем случае системы управления файлами обеспечивали режим многопользовательского доступа. Если операционная система поддерживает многопользовательский режим, вполне реальна ситуация, когда не менее двух пользователей одновременно пытаются работать с одним и тем же файлом. Если все пользователи собираются только читать файл, ничего страшного не произойдет. Но если хотя бы один из них будет изменять файл, для корректной работы этих пользователей требуется взаимная синхронизация их действий по отношению к файлу.

Ограничениями, присущими файловым системам, являются разделение и изоляция данных, дублирование данных, зависимость от данных, несовместимость файлов, фиксированные запросы и быстрое увеличение количества приложений [13].

Когда данные изолированы в отдельных файлах, доступ к ним весьма затруднителен. Для извлечения соответствующей поставленным условиям информации программист должен организовать синхронную обработку двух или более файлов.

Из-за децентрализованной работы с данными, проводимой пользователями независимо друг от друга, в файловой системе фактически допускается бесконтрольное дублирование данных, которое нежелательно по следующим причинам:

— дублирование данных сопровождается неэкономным расходованием ресурсов, поскольку на ввод избыточных данных требуется затрачивать дополнительное время и деньги;

— для их хранения необходимо дополнительное место во внешней памяти, что связано с дополнительными накладными расходами;

— дублирование данных может привести к нарушению их целостности.

Физическая структура и способ хранения записей файлов данных жестко зафиксированы в коде приложений. Это значит, что изменить существующую структуру данных достаточно сложно. Кроме того, при модификации структуры данных программы должны быть изменены с целью соответствия новой структуре файла. А таких программ может быть очень много. Следовательно, программист должен прежде всего выявить все программы, нуждающиеся в доработке, а затем их перепроверить и изменить. Ясно, что выполнение всех этих действий требует больших затрат времени и может явиться причиной появления ошибок. Данная особенность файловых систем называется зависимостью программ от данных.

Поскольку структура файлов определяется кодом приложений, она также зависит от языка программирования этого приложения. Например, структура файла, созданного программой на языке Паскаль, может значительно отличаться от структуры файла, создаваемого программой на языке С. Прямая несовместимость таких файлов затрудняет процесс их совместной обработки и может потребовать создания программного обеспечения, предназначенного для преобразования полей файлов в некоторый общий формат, поэтому этот процесс может оказаться весьма длительным и дорогим.

С точки зрения пользователя возможности файловых систем намного превосходят возможности ручных операций с информацией. Соответственно возрастают и требования к реализации новых или модифицированных запросов. Однако файловые системы требуют больших затрат труда программиста, поскольку все необходимые запросы и отчеты должны быть созданы именно им. Во-первых, во многих организациях типы применяемых запросов и отчетов имели фиксированную форму, и не было никаких инструментов создания незапланированных или произвольных запросов как к самой информации, так и к сведениям о том, какие типы информации доступны.

Во-вторых, в других организациях наблюдалось быстрое увеличение количества файлов и приложений. В конечном счете наступал момент, когда программное обеспечение было неспособно адекватно отвечать запросам пользователей,

эффективность его снижалась, а недостаточность документирования имела следствием дополнительное усложнение сопровождения программ. При этом часто игнорировались вопросы поддержки функционирования системы: не предусматривались меры по обеспечению безопасности или целостности данных; средства восстановления в случае сбоя аппаратного или программного обеспечения были крайне ограничены или вообще отсутствовали. Доступ к файлам часто ограничивался узким кругом пользователей, т.е. не предусматривалось их совместное использование даже сотрудниками одного и того же отдела.

1.2. Системы с использованием баз данных

Все перечисленные выше ограничения файловых систем связаны с определением данных внутри приложений вместо независимого хранения от них и отсутствием других инструментов доступа к данным и их обработки помимо приложений.

Существовавшие ограничения заставили разработчиков информационных систем предложить новый подход к управлению информацией. Этот подход был реализован в рамках программных систем, названных впоследствии системами управления базами данных, а сами хранилища информации, которые работали под управлением СУБД, назывались базами или банками данных.

База данных (БД) — это поименованная совокупность взаимосвязанных данных, управляемых специальной системой, называемой СУБД.

СУБД представляет собой совокупность специальных языковых и программных средств, облегчающих пользователям выполнение всех операций, связанных с организацией хранения данных, их корректировкой и доступом к ним. СУБД служит, по существу, посредником между пользователем и БД.

БД и СУБД являются составными частями более сложной системы, именуемой банком данных [6]. *Банк данных (БнД)* — это система, состоящая из баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования

данных. Термин «банк данных» схож с понятием «система баз данных». *Система баз данных* представляет собой совокупность программного обеспечения, данных и аппаратного обеспечения компьютеров, которая реализует набор приложений и моделей данных, и использует СУБД и прикладное программное обеспечение для создания конкретной информационной системы [18]

БнД является сложной человеко-машинной системой,

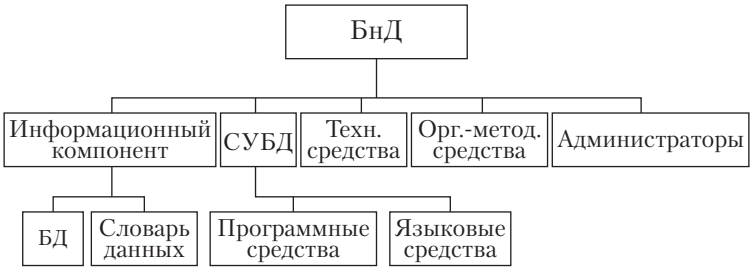

Рис. 1.1. **Компоненты БнД**

состоящей из взаимосвязанных и взаимозависимых компонентов (рис. 1.1).

Информационный компонент БнД содержит базу данных и словарь данных. Словарь данных является хранилищем метаинформации, т.е. информации об информации. Метаинформация включает в себя описание базы данных, информацию о предметной области, представленной в БД, сведения о пользователях БнД, а также некоторую другую информацию.

СУБД предоставляет пользователю программные и языковые средства, обеспечивающие взаимодействие всех частей информационной системы при ее функционировании.

Языковые средства СУБД относятся к языкам четвертого поколения. Языковые средства предназначаются для пользователей разных категорий: конечных пользователей, системных аналитиков, профессиональных программистов. По функциональным возможностям выделяются 10 категорий языков [6]. По форме представления различают аналитические, табличные и графические языковые средства.

Некоторые СУБД предоставляют пользователю несколько языков для достижения одной и той же цели. Например, в системе Delphi [19] для работы с базами данных могут использоваться:

— процедурный язык Object Pascal;
— табличный язык запросов QBE (Query-By-Example);
— язык SQL (Structured Query Language).

В системе FoxPro доступны процедурный язык xBase, табличный язык запросов Relation QBE, а также язык SQL.

Технические средства, используемые в БнД, — это компьютеры различных классов (от больших до персональных), периферийные устройства, коммуникационная (сетевая) аппаратура.

Организационно-методические средства представляют собой различные инструкции, методические и регламентирующие материалы, предназначенные для пользователей, взаимодействующих с БнД.

Администраторы — это специалисты, которые обеспечивают создание, функционирование и развитие БнД. База данных и СУБД являются коллективными ресурсами, которыми следует управлять так же, как и любыми другими ресурсами [13]. Обычно управление данными и базой данных предусматривает управление и контроль за СУБД и помещенными в нее данными. *Администратор данных* отвечает за управление данными, включая планирование базы данных, разработку и сопровождение стандартов, прикладных алгоритмов и деловых процедур, а также за концептуальное и логическое проектирование базы данных. *Администратор базы данных* отвечает за физическую реализацию базы данных, включая физическое проектирование и воплощение проекта, за обеспечение безопасности и целостности данных, за сопровождение операционной системы, а также за обеспечение максимальной производительности приложений и пользователей. По сравнению с администратором данных обязанности администратора базы данных носят более технический характер, и для него необходимо знание конкретной СУБД и системного окружения. В одних организациях между этими ролями не делается различий, а в других важность коллективных ресурсов отражена именно в выделении отдельных групп персонала с указанным кругом обязанностей.

В проектировании больших баз данных участвуют разработчики двух разных типов: *разработчики логической структуры базы данных* и *разработчики физической структуры базы данных*. Разработчик логической структуры базы данных занимается идентификацией данных (т.е. сущностей и их атрибутов), связей между данными и устанавливает ограничения, накладываемые на хранимые данные. Разработчик

логической структуры базы данных должен обладать всесторонним и полным пониманием структуры данных организации и процессов ее функционирования.

Для эффективной работы разработчик логической структуры базы данных должен как можно раньше вовлечь всех предполагаемых пользователей базы данных в процесс создания модели данных. Деятельность разработчика логической структуры базы данных делится на две части

— концептуальное проектирование базы данных, которое совершенно не зависит от таких деталей ее воплощения, как конкретная целевая СУБД, приложения, языки программирования или любые другие физические характеристики;

— даталогическое проектирование базы данных, которое проводится с учетом особенностей выбранной модели данных.

Разработчик физической структуры базы данных получает готовую логическую модель данных и занимается ее физической реализацией, в том числе:

— преобразованием логической модели данных в набор таблиц и ограничений целостности данных;

— выбором конкретных структур хранения и методов доступа к данным, обеспечивающих необходимый уровень производительности при работе с базой данных;

—проектированием любых требуемых мер защиты данных.

Многие этапы физического проектирования базы данных в значительной степени зависят от выбранной целевой СУБД, а потому может существовать несколько различных способов воплощения требуемой схемы. Следовательно, разработчик физической структуры базы данных должен разбираться в функциональных возможностях целевой СУБД и понимать достоинства и недостатки каждого возможного варианта реализации. Разработчик физической структуры базы данных должен уметь выбрать наиболее подходящую стратегию хранения данных с учетом всех существующих особенностей их использования.

После создания базы данных начинается разработка приложений, предоставляющих пользователям необходимые им функциональные возможности. Именно эту работу и выполняют *прикладные программисты*. Обычно прикладные программисты работают на основе спецификаций, созданных системными аналитиками. Как правило, каждая программа содержит некоторые операторы, требующие от СУБД выполнения определенных действий с базой данных — например таких, как извлечение, вставка, обновление или удаление дан-

ных. Эти программы могут создаваться на различных языках программирования третьего или четвертого поколения.

Конечные пользователи являются клиентами базы данных; она проектируется, создается и поддерживается для того, чтобы обслуживать их информационные потребности. Пользователей можно классифицировать по способу использования ими системы на рядовых и опытных.

Рядовые пользователи обычно и не подозревают о наличии СУБД. Они обращаются к базе данных с помощью специальных приложений, позволяющих в максимальной степени упростить выполняемые ими операции. Такие пользователи инициируют выполнение операций базы данных, вводя простейшие команды или выбирая команды меню. Это значит, что таким пользователям не нужно ничего знать о базе данных или СУБД. Например, чтобы узнать цену товара, кассир в супермаркете использует сканер для считывания нанесенного на него штрих-кода. В результате этого простейшего действия специальная программа не только считывает штрих-код, но и выбирает на основе его значения цену товара из базы данных, а также уменьшает значение в другом поле базы данных, обозначающем остаток таких товаров на складе, после чего выбивает цену и общую стоимость на кассовом аппарате.

Опытные пользователи знакомы со структурой базы данных и возможностями СУБД. Для выполнения требуемых операций они могут использовать такой язык запросов высокого уровня, как SQL. А некоторые опытные пользователи могут даже создавать собственные прикладные программы.

Использование СУБД для доступа к данным дает ряд преимуществ, к которым относятся [13]:

— контроль за избыточностью данных;
— непротиворечивость данных;
— больше полезной информации при том же объеме хранимых данных;
— совместное использование данных;
— поддержка целостности данных;
— повышенная безопасность;
— применение стандартов;
— повышение эффективности с ростом масштабов системы;
— возможность нахождения компромисса при противоречивых требованиях;
— повышение доступности данных и их готовности к работе;

— улучшение показателей производительности;
— упрощение сопровождения системы за счет независимости от данных;
— улучшенное управление параллельной работой;
— развитые службы резервного копирования и восстановления.

Контроль за избыточностью данных. Как отмечалось выше, традиционные файловые системы неэкономно расходуют внешнюю память, сохраняя одни и те же данные в нескольких файлах. При использовании базы данных, наоборот, предпринимается попытка исключить избыточность данных за счет интеграции файлов, что позволяет отказаться от хранения нескольких копий одного и того же элемента информации. Однако полностью избыточность информации в базах данных не исключается, а лишь ограничивается ее степень. В одних случаях ключевые элементы данных необходимо дублировать для моделирования связей, а в других случаях некоторые данные требуется дублировать из соображений повышения производительности системы.

Непротиворечивость данных. Устранение избыточности данных или контроль над ней позволяет уменьшить риск возникновения противоречивых состояний. Если элемент данных хранится в базе только в одном экземпляре, то для изменения его значения потребуется выполнить только одну операцию обновления, причем новое значение станет доступным сразу всем пользователям базы данных. А если этот элемент данных с ведома системы хранится в базе данных в нескольких экземплярах, то такая система сможет следить за тем, чтобы копии не противоречили друг другу. К сожалению, во многих современных СУБД такой способ обеспечения непротиворечивости данных не поддерживается автоматически.

Больше полезной информации при том же объеме хранимых данных. Благодаря объединению рабочих данных предприятия на основе тех же данных можно получать дополнительную информацию. Например, в файловой системе сотрудникам отдела кадров недоступны сведения об окладах работников предприятия. Аналогично, сотрудники расчетного отдела бухгалтерии не имеют полных сведений семейном положении работников. При объединении этих файлов в общей базе сотрудники обоих отделов смогут получать больше информации.

Совместное использование данных. Файлы обычно принадлежат отдельным лицам или целым отделам, которые

используют их в своей работе. В то же время база данных принадлежит всему предприятию в целом и может совместно использоваться всеми зарегистрированными пользователями. При такой организации работы большее количество пользователей может работать с большим объемом данных. Более того, при этом можно создавать новые приложения на основе уже существующей в базе данных информации и добавлять в нее только те данные, которые в настоящий момент еще не хранятся в ней, а не определять заново требования ко всем данным, необходимым новому приложению. Новые приложения могут также использовать такие предоставляемые типичными СУБД функциональные возможности, как определение структур данных и управление доступом к данным, организация параллельной обработки и обеспечение средств копирования/восстановления, исключив необходимость реализации этих функций со своей стороны.

Поддержка целостности данных. Целостность базы данных означает корректность и непротиворечивость хранимых в ней данных. Целостность обычно описывается с помощью ограничений, т.е. правил поддержки непротиворечивости, которые не должны нарушаться в базе данных. Ограничения можно применять к элементам данных внутри одной записи или к связям между записями. Например, ограничение целостности может гласить, что оклад сотрудника не должен превышать 20 тыс. руб. в месяц или же что в записи с данными о сотруднике номер подразделения, в котором он работает, должен соответствовать реально существующему в компании. Таким образом, интеграция данных позволяет администратору баз данных задавать требования по поддержке целостности данных, а системе управления базами данных применять их.

Повышенная безопасность. Безопасность базы данных заключается в защите базы данных от несанкционированного доступа со стороны пользователей. Без привлечения соответствующих мер безопасности объединенные данные становятся более уязвимыми, чем данные в файловой системе. Однако объединение позволяет администратору баз данных определить требуемую систему безопасности базы данных, а СУБД привести ее в действие. Система обеспечения безопасности может быть выражена в форме имен и паролей для идентификации пользователей, которые зарегистрированы в этой базе данных. Доступ к данным со стороны зарегистрированного пользователя может быть ограничен только

некоторыми операциями (извлечением, вставкой, обновлением и удалением).

Применение стандартов. Объединение данных позволяет администратору баз данных определять и применять необходимые стандарты. Например, стандарты предприятия, государственные и международные стандарты могут регламентировать формат данных при обмене ими между системами, соглашения об именах, форму представления документации, процедуры обновления и правила доступа.

Повышение эффективности с увеличением масштабов системы. Комбинируя все рабочие данные предприятия в одной базе данных и создавая приложения, которые работают с одним источником данных, можно добиться существенной экономии средств. В этом случае бюджет, который обычно выделялся каждому отделу для разработки и поддержки их собственных файловых систем, можно объединить с бюджетами других подразделений (с более низкой общей стоимостью), что позволит добиться повышения эффективности при росте масштабов производства. Теперь консолидированный бюджет можно будет использовать для приобретения оборудования в той конфигурации, которая в большей степени отвечает потребностям предприятия.

Возможность нахождения компромисса для противоречивых требований. Потребности одних пользователей или подразделений предприятия могут противоречить потребностям других пользователей. Но поскольку база данных контролируется администратором баз данных, он может принимать решения о проектировании и способе использования базы данных, при которых имеющиеся ресурсы всего предприятия в целом будут использоваться наилучшим образом. Эти решения обеспечивают оптимальную производительность для самых важных приложений, причем чаще всего за счет менее критичных.

Повышение доступности данных и их готовности к работе. Данные, необходимость в которых не ограничивается рамками отдельных подразделений, в результате объединения становятся непосредственно доступными конечным пользователям. Потенциально это повышает функциональность системы, что, например, может быть использовано для более качественного обслуживания конечных пользователей или клиентов предприятия. Во многих СУБД предусмотрены языки запросов или инструменты для создания отчетов, которые позволяют пользователям вводить не предусмотренные

заранее запросы и почти немедленно получать требуемую информацию на своих терминалах, не прибегая к помощи программиста, который для извлечения этой информации из базы данных должен был бы создать специальное программное обеспечение.

Улучшение показателей производительности. Поскольку в СУБД предусмотрено много стандартных функций, которые программист обычно должен самостоятельно реализовать в приложениях для файловых систем, то на базовом уровне СУБД обеспечивает все низкоуровневые процедуры работы с файлами, которую обычно выполняют приложения. Наличие этих процедур позволяет программисту сконцентрироваться на разработке более специальных, необходимых пользователям функций, не заботясь о подробностях их воплощения на более низком уровне. Во многих СУБД предусмотрена также среда разработки с инструментами, упрощающими создание приложений баз данных. Результатом является повышение производительности работы программистов и сокращение времени разработки новых приложений.

Упрощение сопровождения системы за счет независимости от данных. В файловых системах описания данных и логика доступа к данным встроены в каждое приложение, поэтому программы становятся зависимыми от данных. Для изменения структуры данных или для изменения способа хранения данных на диске может потребоваться существенно преобразовать все программы, на которые эти изменения способны оказать влияние. В СУБД подход иной: описания данных отделены от приложений, а потому приложения защищены от изменений в описаниях данных. Эта особенность называется независимостью от данных. Наличие независимости программ от данных значительно упрощает обслуживание и сопровождение приложений, работающих с базой данных.

Улучшенное управление параллельной работой. В некоторых файловых системах при одновременном доступе к одному и тому же файлу двух пользователей может возникнуть конфликт двух запросов, результатом которого будет потеря информации или утрата ее целостности. В свою очередь, во многих СУБД предусмотрена возможность параллельного доступа к базе данных и гарантируется отсутствие подобных проблем.

Развитые службы резервного копирования и восстановления. Ответственность за обеспечение защиты данных от