Тема 2. Представление данных в памяти ЭВМ. Модели данных

Цель и задачи темы

Изучить терминологию, теоретические вопросы реляционных баз данных, классификацию баз данных, модели данных.

Оглавление

2.1. Определения и понятия теории базы данных

База данных (БД, database) — поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Предметная область — некоторая часть реально существующей системы, функционирующая как самостоятельная единица.

Полная предметная область может представлять собой экономику страны или группы союзных государств, однако на практике для информационных систем наибольшее значение имеет предметная область масштаба отдельного предприятия или корпорации.

Система управления базами данных (СУБД) — комплекс программных и языковых средств, необходимых для создания и модификации базы данных, добавления, модификации,

удаления, поиска и отбора информации, представления информации на экране и в печатном виде, разграничения прав доступа к информации, выполнения других операций с базой.

Реляционная БД — основной тип современных баз данных. Состоит из таблиц, между которыми могут существовать связи по ключевым значениям.

Таблица базы данных (table) — регулярная структура, которая состоит из однотипных строк (записей, records), разбитых на столбцы (поля, fields).

В теории реляционных баз данных синоним таблицы — отношение (relation), в котором строка называется кортежем, а столбец — атрибутом.

В концептуальной модели реляционной БД аналогом таблицы является сущность (entity), с определенным набором свойств — атрибутов, способных принимать определенные значения (набор допустимых значений — домен).

Ключевой элемент таблицы (ключ, regular key) — такое ее поле (простой ключ) или строковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На практике для использования ключей создаются индексы — служебная информация, содержащая упорядоченные сведения о ключевых значениях. В реляционной теории и концептуальной модели понятие «ключ» применяется для атрибутов отношения или сущности.

Первичный ключ (primary key) — главный ключевой элемент, однозначно идентифицирующий строку в таблице. Могут также существовать альтернативный (candidate key) и уникальный (unique key) ключи, служащие также для идентификации строк в таблице.

В реляционной теории первичный ключ — минимальный набор атрибутов, однозначно идентифицирующий кортеж в отношении.

В концептуальной модели первичный ключ — минимальный набор атрибутов сущности, однозначно идентифицирующий экземпляр сущности.

Связь (relation) — функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (parent, родительской) таблице — первичный, второй — внешний ключ — во внешней (child, дочерней) таблице, как правило, первичным не является и образует связь «один ко многим» (1:N). В случае первичного внешнего ключа связь между таблицами имеет тип «один к одному» (1:1). Информация о связях сохраняется в базе данных.

Внешний ключ (foreign key) — ключевой элемент подчиненной (внешней, дочерней) таблицы, значение которого совпадает со значением первичного ключа главной (родительской) таблицы.

Ссылочная целостность данных (referential integrity) — набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

Хранимые процедуры (stored procedures) — программные модули, сохраняемые в базе данных для выполнения определенных операций с информацией базы.

Триггеры (triggers) — хранимые процедуры, обеспечивающие соблюдение условий ссылочной целостности данных в операциях изменения первичных ключей (возможно каскадное изменение данных), удаления записей в главной таблице (каскадное удаление в дочерних таблицах) и добавления записей или изменения данных в дочерних таблицах.

Объект (object) — элемент информационной системы, обладающий определенными свойствами (properties) и определенным образом реагирующий на внешние события (events).

Система — совокупность взаимодействующих между собой и с внешним окружением объектов.

Репликация базы данных — создание копий базы данных (реплик), которые могут обмениваться обновляемыми данными или реплицированными формами, отчетами или другими объектами в результате выполнения процесса синхронизации.

Транзакция — изменение информации в базе в результате выполнения одной операции или их последовательности, которое должно быть выполнено полностью или не выполнено вообще. В СУБД существуют специальные механизмы обеспечения транзакций.

Язык SQL (Structured Query Language) — универсальный язык работы с базами данных, включающий возможности ее создания, модификации структуры, отбора данных по запросам, модификации информации в базе и прочие операции манипулирования базой данных.

Null — значение поля таблицы, показывающее, что информация в данном поле отсутствует. Разрешение на возможность существования значения Null может задаваться для отдельных полей таблицы.

2.2. Классификация баз данных

По технологии обработки данных базы данных подразделяются на централизованные и распределенные.

Централизованная база данных хранится в памяти одной вычислительной системы. Эта вычислительная система может быть мэйнфреймом — тогда доступ к ней организуется с использованием терминалов — или файловым сервером локальной сети ПК.

Распределенная база данных состоит из нескольких, возможно, пересекающихся или даже дублирующих друг друга частей, которые хранятся в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на базы данных с локальным доступом и базы данных с сетевым доступом.

Для всех современных баз данных можно организовать сетевой доступ с многопользовательским режимом работы.

Централизованные базы данных с сетевым доступом могут иметь следующую архитектуру:

- файл-сервер;
- клиент-сервер базы данных;
- «тонкий клиент» — сервер приложений — сервер базы данных (трехуровневая архитектура).

Рис. 1. Схема работы с БД в локальной сети с выделенным файловым сервером

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (файловый сервер). На этот компьютер устанавливается операционная система (ОС) для выделенного сервера (например, Microsoft Windows Server 2003). На нем же хранится совместно используемая централизованная БД в виде одного или группы файлов. Все

другие компьютеры сети выполняют функции рабочих станций (могут работать в ОС Microsoft Windows 2000 Professional или Microsoft Windows 98). Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где и производится обработка информации (рис. 1). При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также локальные БД на рабочих станциях.

Рис. 2. Схема работы с БД в архитектуре «Клиент-сервер»

Клиент-сервер. В этой архитектуре на выделенном сервере, работающем под управлением серверной операционной системы, устанавливается специальное программное обеспечение (ПО) — сервер БД, например Microsoft®SQL Server™или Oracle. СУБД подразделяется на две части: клиентскую и серверную. Основа работы сервера БД — использование языка запросов (SQL). Запрос на языке SQL, передаваемый клиентом (рабочей станцией) серверу БД, порождает поиск и извлечение данных на сервере. Извлеченные данные транспортируются по сети от сервера к клиенту (рис. 2). Тем самым количество передаваемой по сети информации уменьшается во много раз.

Трехуровневая архитектура функционирует в интранет- и интернет-сетях. Клиентская часть («тонкий клиент»), взаимодействующая с пользователем, представляет собой HTML-страницу в Web-браузере либо Windows-приложение, взаимодействующее с Web-сервисами. Вся программная логика вынесена на сервер приложений, который

обеспечивает формирование запросов к базе данных, передаваемых на выполнение серверу баз данных. Сервер приложений может быть Web-сервером или специализированной программой (например, Oracle Forms Server) (рис. 3).

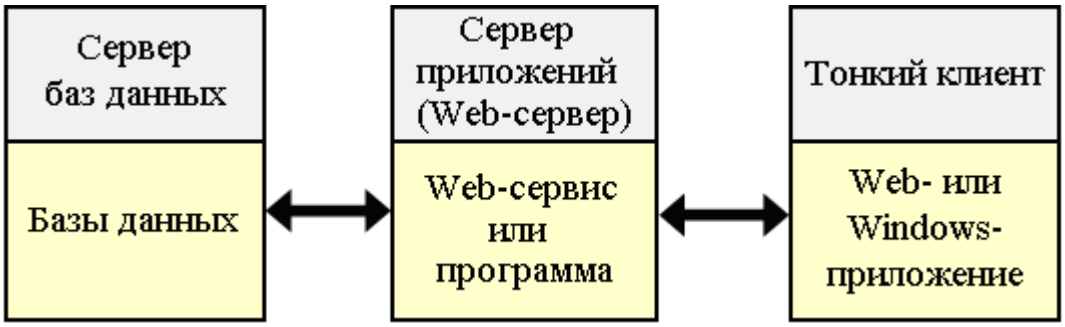

Рис. 3. Схема работы с БД в трехуровневой архитектуре

2.3. Иерархические и сетевые модели данных

В иерархической модели данных имеется один главный объект и остальные — подчиненные — объекты, находящиеся на разных уровнях иерархии. Взаимосвязи объектов образуют иерархическое дерево с одним корневым объектом.

Иерархическая БД состоит из упорядоченного набора нескольких экземпляров одного типа дерева. Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя (рис. 4).

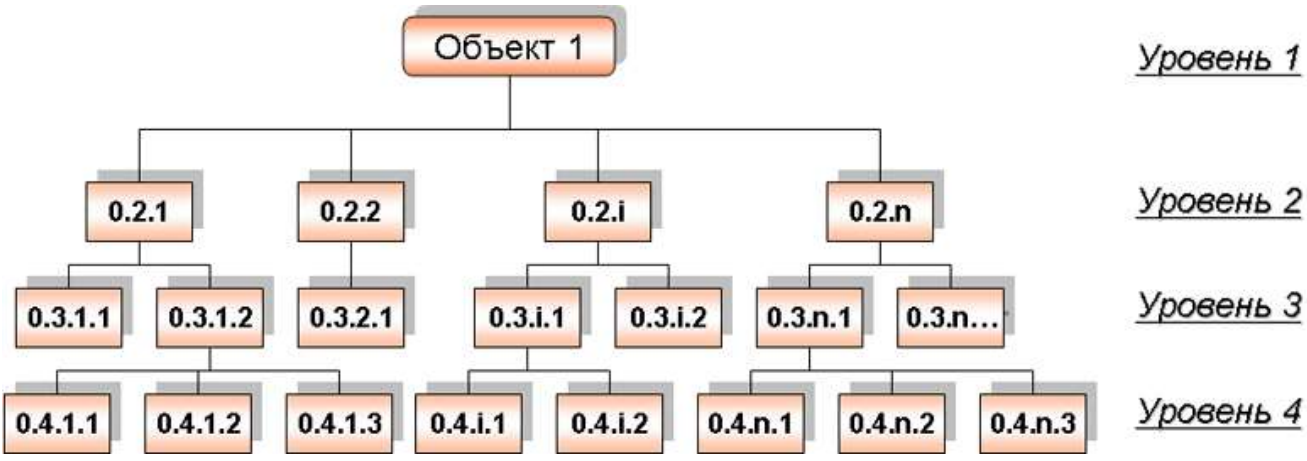

Рис. 4. Схема иерархической модели данных

Типичным представителем (наиболее известным и распространенным) является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г. До сих пор поддерживается много баз данных этой системы.

Сетевые базы данных

Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.

В сетевой модели данных любой объект может быть одновременно и главным, и подчиненным, и может участвовать в образовании любого числа взаимосвязей с другими объектами. Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно — из набора экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи (рис. 5).

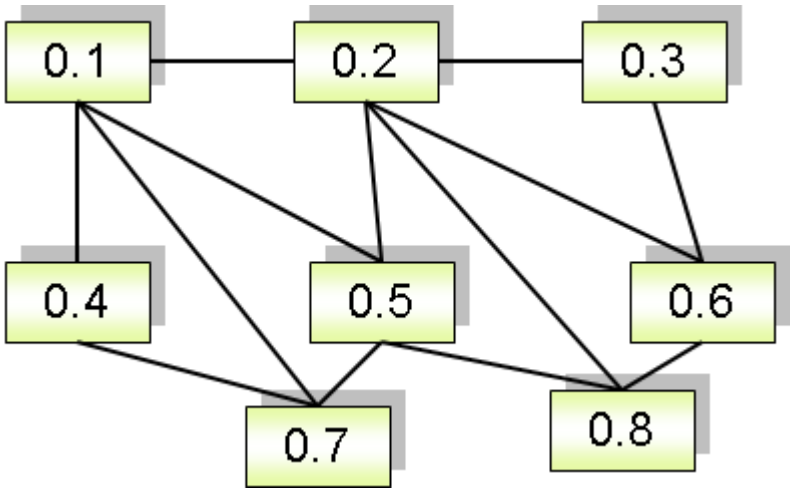

Рис. 5. Схема сетевой модели данных

Типичным представителем является Integrated Database Management System (IDMS) компании Cullinet Software, Inc., предназначенная для использования на машинах основного класса фирмы IBM под управлением большинства

операционных систем. Архитектура системы основана на предложениях Data Base Task Group (DBTG) Комитета по языкам программирования Conference on Data Systems Languages (CODASYL) — организации, ответственной за определение языка программирования Кобол. Отчет DBTG был опубликован в 1971 г., а позже появилось несколько систем, среди которых IDMS.

2.4. Регуляционные базы данных

Реляционные системы далеко не сразу получили широкое распространение. В то время как основные теоретические результаты в этой области были получены еще в 70-х г. и тогда же появились первые прототипы реляционных СУБД, долгое время считалось невозможным добиться эффективной реализации таких систем. Однако постепенное накопление методов и алгоритмов организации реляционных баз данных и управления ими привели к тому, что уже в середине 80-х г. реляционные системы практически вытеснили с мирового рынка ранние СУБД.

Реляционная модель данных основывается на математических принципах, вытекающих непосредственно из теории множеств и логики предикатов. Эти принципы впервые были применены в области моделирования данных в конце 1960-х гг. доктором Е. Ф. Коддом, в то время работавшим в IBM, а впервые опубликованы в 1970 г.

Техническая статья «Реляционная модель данных для больших разделяемых банков данных» доктора Е. Ф. Кодда, опубликованная в 1970 г., является родоначальницей современной теории реляционных БД. Доктор Кодд определил 13 правил реляционной модели (которые называют тринадцатью правилами Кодда).

13 правил Кодда

1. Реляционная СУБД должна быть способна полностью управлять базой данных через ее реляционные возможности.

2. Информационное правило — вся информация в реляционной БД (включая имена таблиц и столбцов) должна определяться строго как значения в таблицах.
3. Гарантированный доступ — любое значение в реляционной БД должно быть гарантированно доступно для использования через комбинацию имени таблицы, значения первичного ключа и имени столбца.
4. Поддержка пустых значений (null value) — СУБД должна уметь работать с пустыми значениями (неизвестными или неиспользованными значениями), в отличие от значений по умолчанию и независимо для любых доменов.
5. Онлайновый реляционный каталог — описание БД и ее содержание должны быть представлены на логическом уровне как таблицы, к которым можно применять запросы, используя язык базы данных.
6. Исчерпывающий язык управления данными — по крайней мере, один из поддерживаемых языков должен иметь четко определенный синтаксис и быть всеобъемлющим. Он должен поддерживать описание структуры данных и манипулирование ими, правила целостности, авторизацию и транзакции.
7. Правило обновления представлений (views) — все представления, теоретически обновляемые, могут быть обновлены через систему.
8. Вставка, обновление и удаление — СУБД поддерживает не только запрос на отбор данных, но и вставку, обновление и удаление.
9. Физическая независимость данных — на программы-приложения и специальные программы логически не влияют изменения физических методов доступа к данным и структур хранилищ данных.
10. Логическая независимость данных — на программы-приложения и специальные программы логически не влияют, в пределах разумного, изменения структур таблиц.
11. Независимость целостности — язык БД должен быть способен определять правила целостности. Они должны

сохраняться в онлайновом справочнике, и не должно существовать способа их обойти.

12. Независимость распределения — на программы-приложения и специальные программы логически не влияет, первый раз используются данные или повторно.
13. Неподрывность — невозможность обойти правила целостности, определенные через язык базы данных, использованием языков низкого уровня.

Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для реляционных баз данных.

Существует много подходов к определению реляционной алгебры, которые различаются наборами операций и способами их интерпретации, но, в принципе, являются более или менее равносильными. Расширенный начальный вариант алгебры, который был предложен Коддом, называется алгеброй Кодда.

В этом варианте набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса — теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

- объединения отношений;
- пересечения отношений;
- взятия разности отношений;
- взятия декартова произведения отношений.

Специальные реляционные операции включают:

- ограничение отношения;
- проекцию отношения;
- соединение отношений;
- деление отношений.

Кроме того, в состав алгебры включается операция присваивания, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и операция переименования атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

- При выполнении операции объединения (UNION) двух отношений с одинаковыми заголовками производится отношение, включающее все кортежи, которые входят хотя бы в одно из отношений — операндов.
- Операция пересечения (INTERSECT) двух отношений с одинаковыми заголовками производит отношение, включающее все кортежи, которые входят в оба отношения-операнда.
- Отношение, являющееся разностью (MINUS) двух отношений с одинаковыми заголовками, включает все кортежи, входящие в отношение — первый операнд, такие, что ни один из них не входит в отношение, которое является вторым операндом.
- При выполнении декартова произведения (TIMES) двух отношений, пересечение заголовков которых пусто, производится отношение, кортежи которого производятся путем объединения кортежей первого и второго операндов.
- Результатом ограничения (WHERE) отношения по некоторому условию является отношение, включающее кортежи отношения-операнда, удовлетворяющее этому условию.
- При выполнении проекции (PROJECT) отношения на заданное подмножество множества его атрибутов производится отношение, кортежи которого являются соответствующими подмножествами кортежей отношения-операнда.
- При соединении (JOIN) двух отношений по некоторому условию образуется результирующее отношение, кортежи которого производятся путем объединения кортежей первого и второго отношений и удовлетворяют этому условию.

- У операции реляционного деления (DIVIDE BY) два операнда — бинарное и унарное отношения. Результирующее отношение состоит из унарных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) включает множество значений второго операнда.
- Операция переименования (RENAME) производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены.
- Операция присваивания (:=) позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

Кодд предложил применение реляционной алгебры в СУРБД, для расчленения данных в связанные наборы. Он организовал свою систему БД вокруг концепции, основанной на наборах данных.

В реляционной модели данные разбиваются на наборы, которые составляют табличную структуру. Эта структура таблиц состоит из индивидуальных элементов данных, называемых полями. Одиночный набор или группа полей известна как запись.

Модель данных, или концептуальное описание предметной области, — самый абстрактный уровень проектирования баз данных.

С точки зрения теории реляционных БД, основные принципы реляционной модели на концептуальном уровне можно сформулировать следующим образом:

- все данные представляются в виде упорядоченной структуры, определенной в виде строк и столбцов и называемой отношением;
- все значения являются скалярами. Это означает, что для любой строки и столбца любого отношения существует одно и только одно значение;

- все операции выполняются над целым отношением, и результатом их выполнения также является целое отношение. Этот принцип называется замыканием.

Формулируя принципы реляционной модели, доктор Кодд выбрал термин «отношение» (relation), потому что, по его мнению, этот термин однозначен (в то время как, например, термин «таблица» имеет множество различных видов — таблица в тексте, электронная таблица и пр.). Весьма распространено следующее заблуждение: реляционная модель названа так потому, что она определяет связи между таблицами. На самом деле, название этой модели происходит от отношений (таблиц базы данных), лежащих в ее основе.

Каждая строка, содержащая данные, называется кортежем, каждый столбец отношения называется атрибутом (на уровне практической работы с современными реляционными БД используются термины «запись» и «поле»).

Элементами описания реляционной модели данных на концептуальном уровне являются сущности, атрибуты, домены и связи.

Сущность — некоторый обособленный объект или событие, информацию о котором необходимо сохранять в базе данных, имеющий определенный набор свойств — атрибутов. Сущности могут быть как физические (реально существующие объекты: например, СТУДЕНТ, атрибуты — номер зачетной книжки, фамилия, его факультет, специальность, номер группы и т. д.), так и абстрактные (например, ЭКЗАМЕН, атрибуты — дисциплина, дата, преподаватель, аудитория и пр.). Для сущностей различают ее тип и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр — конкретными значениями свойств.

Атрибуты сущности бывают:

1. *Идентифицирующие и описательные.* Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно

распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными.

2. *Простые и составные.* Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от особенностей процессов его использования и может быть связано с обеспечением высокой скорости работы с большими базами данных.
3. *Однозначные и многозначные* — могут иметь соответственно одно или много значений для каждого экземпляра сущности.
4. *Основные и производные.* Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст человека вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности — множества значений (или домена), которые может принимать данный атрибут.

Домен — это набор всех допустимых значений, которые может содержать атрибут. Понятие «домен» часто путают с понятием «тип данных». Необходимо различать эти два понятия. Тип данных — это физическая концепция, а домен — логическая. Например, «целое число» — это тип данных, а «возраст» — это домен.

Связи — на концептуальном уровне представляют собой простые ассоциации между сущностями. Например, утверждение «Покупатели приобретают продукты» указывает,

что между сущностями «Покупатели» и «Продукты» существует связь, и такие сущности называются участниками этой связи.

Существует несколько типов связей между двумя сущностями: это связи «один к одному», «один ко многим» и «многие ко многим».

Каждая связь в реляционной модели характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если сущность одного типа оказывается по необходимости связанной с сущностью другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной.

Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи — связь между отделом и сотрудниками, которые в нем работают.

Диаграмма «сущности-связи» (Entity-Relationship diagrams, или E/R diagram) служит для описания схемы базы на концептуальном уровне проектирования. Метод был предложен в 1976 г. Питером Пин Шань Ченом (Peter Pin Shan Chen) [2]. На диаграммах «сущности-связи» сущности изображаются в виде прямоугольников, атрибуты — в виде эллипсов, а связи — в виде ромбов (рис. 6).

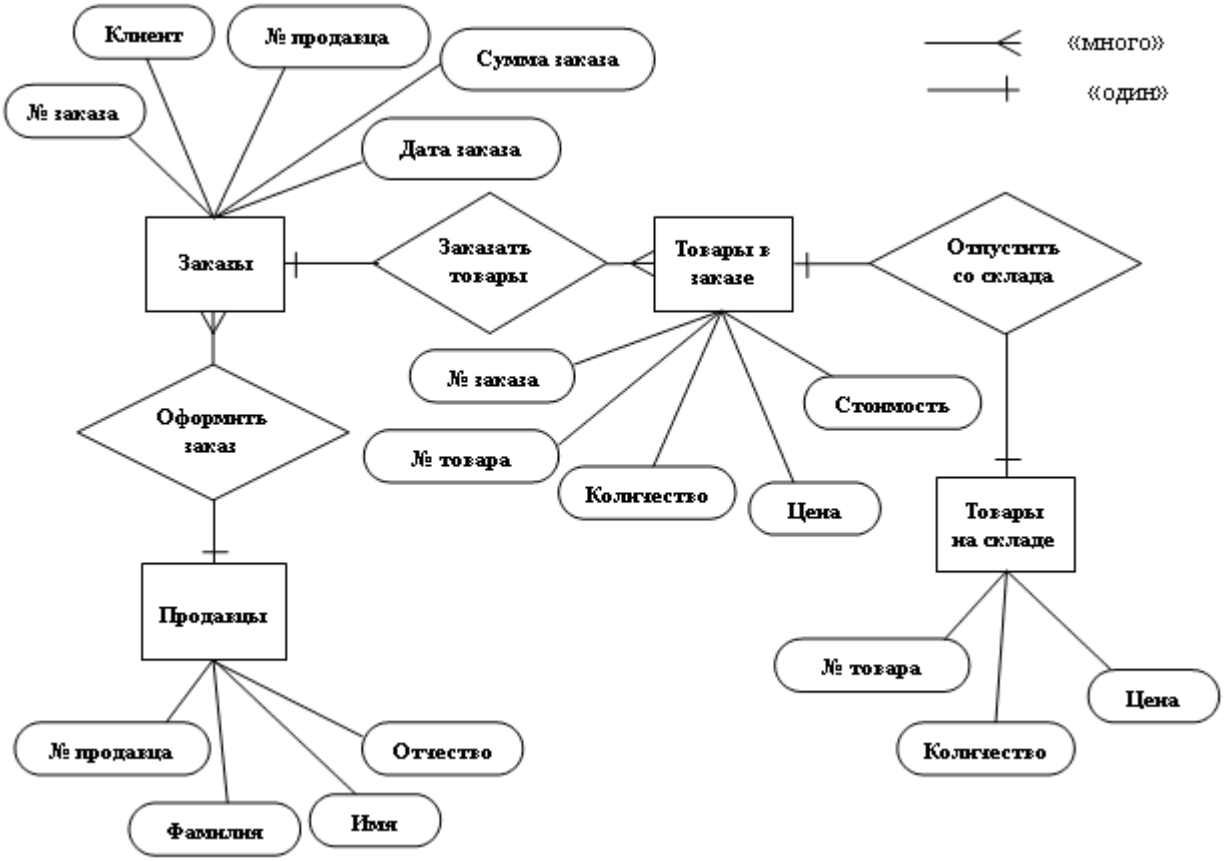

Рис. 6. Диаграмма «сущности-связи»

В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). Кроме того, различные программные средства, реализующие одну и ту же нотацию, могут отличаться своими возможностями. По сути, все варианты диаграмм «сущность-связь» исходят из одной идеи — рисунок всегда нагляднее текстового описания. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями.

Проектирование схемы БД должно решать задачи минимизации дублирования данных, упрощения и ускорения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных. Для решения подобных проблем проводится нормализация отношений.

Однако в технологии работы с хранилищами данных может использоваться обратный прием — денормализация отношений с целью увеличения скорости выполнения запросов к очень большим объемам архивных данных.

В рамках реляционной модели данных Э. Ф. Коддом были разработаны принципы нормализации отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме.

Нормализация — это формальный метод анализа отношений на основе их первичного ключа и существующих связей. Ее задача — это замена одной схемы (или совокупности отношений) БД другой схемой, в которой отношения имеют более простую и регулярную структуру.

При работе с реляционной моделью для создания отношений приемлемого качества достаточно выполнения требований первой нормальной формы.

Первая нормальная форма (1НФ) связана с понятиями простого и сложного атрибутов. Простой атрибут — это атрибут, значения которого атомарны (т. е. неделимы). Сложный атрибут может иметь значение, представляющее собой объединение нескольких значений одного или разных доменов. В первой нормальной форме устраняются повторяющиеся атрибуты или группы атрибутов, т. е. производится выявление неявных сущностей, «замаскированных» под атрибуты.

Отношение приведено к 1НФ, если все его атрибуты — простые, т. е. значение атрибута не должно быть множеством или повторяющейся группой.

Для приведения таблиц к 1НФ необходимо разбить сложные атрибуты на простые, а многозначные атрибуты вынести в отдельные отношения.

Вторая нормальная форма (2НФ) применяется к отношениям с составными ключами (состоящими из двух

и более атрибутов) и связана с понятиями функциональной зависимости.

Если в любой момент времени каждому значению атрибута A соответствует единственное значение атрибута B, то B функционально зависит от A (A→B). Атрибут (группа атрибутов) A называется детерминатором.

Во второй нормальной форме устраняются атрибуты, зависящие только от части уникального ключа. Эта часть уникального ключа определяет отдельную сущность.

Отношение находится во 2НФ, если оно приведено к 1НФ и каждый неключевой атрибут функционально полно зависит от составного первичного ключа.

Третья нормальная форма (3НФ) связана с понятием транзитивной зависимости. Пусть A, B, C — атрибуты некоторого отношения. При этом A→B и B→C, но обратное соответствие отсутствует, т. е. C не зависит от B или B не зависит от A. Тогда говорят, что C транзитивно зависит от A (A→→C).

В третьей нормальной форме устраняются атрибуты, которые зависят от атрибутов, не входящих в уникальный ключ. Эти атрибуты являются основой отдельной сущности.

Отношение находится в 3НФ, если оно находится во 2НФ и не имеет атрибутов, не входящих в первичный ключ и находящихся в транзитивной зависимости от первичного ключа.

Существуют также нормальная форма Бойса-Кодда (НФБК), 4НФ и 5НФ. Однако наибольшее значение имеет 1НФ, так как последующие НФ связаны с понятиями о составных ключах и сложных зависимостях от ключей, а на практике встречаются обычно более простые случаи.

Моделирование структуры базы данных при помощи алгоритма нормализации имеет серьезные недостатки:

1. Методика нормализации предполагает первоначальное размещение всех атрибутов проектируемой предметной области в одном отношении, что является очень неестественной операцией. Интуитивно разработчик сразу проектирует несколько отношений в соответствии с обнаруженными сущностями. Даже если совершить насилие над собой и создать одно или несколько отношений, включив в них все предполагаемые атрибуты, то совершенно неясен смысл полученного отношения.
2. Невозможно сразу определить полный список атрибутов. Пользователи имеют привычку называть разными именами одни и те же вещи или, наоборот, называть одними именами разные вещи.
3. Для проведения процедуры нормализации необходимо выделить зависимости атрибутов, что тоже очень нелегко.

В реальном проектировании структуры базы данных применяется другой метод — так называемое семантическое моделирование. Семантическое моделирование представляет собой моделирование структуры данных, опирающееся на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты диаграмм «сущность-связь» (ERD) с построением концептуальной модели базы данных.

Любой специалист, освоивший общие принципы оптимальной организации реляционных баз данных, в состоянии построить модель, не противоречащую принципам нормализации.

Реляционная БД на физическом уровне состоит из таблиц, между которыми могут существовать связи по ключевым значениям. Одновременно с таблицами и информацией о связях в реляционной базе данных могут присутствовать «хранимые процедуры» и, в частности, «триггеры», обеспечивающие соблюдение условий ссылочной целостности базы.

Соблюдение условий ссылочной целостности в реляционной базе данных

Правило соответствия внешних ключей первичным — основное правило соблюдения условий ссылочной целостности. Для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительской таблице.

Ссылочная целостность может нарушиться в результате операций вставки (добавления), обновления и удаления записей в таблицах. В определении ссылочной целостности участвуют две таблицы — родительская и дочерняя, для каждой из них возможны эти операции, поэтому существует шесть различных вариантов, которые могут привести либо не привести к нарушению ссылочной целостности.

Для родительской таблицы:

- *Вставка.* Возникает новое значение первичного ключа. Существование записей в родительской таблице, на которые нет ссылок из дочерней таблицы, допустимо, операция не нарушает ссылочной целостности.
- *Обновление.* Изменение значения первичного ключа в записи может привести к нарушению ссылочной целостности.
- *Удаление.* При удалении записи удаляется значение первичного ключа. Если есть записи в дочерней таблице, ссылающиеся на ключ удаляемой записи, то значения внешних ключей станут некорректными. Операция может привести к нарушению ссылочной целостности.

Для дочерней таблицы:

- *Вставка.* Нельзя вставить запись в дочернюю таблицу, если для новой записи значение внешнего ключа некорректно. Операция может привести к нарушению ссылочной целостности.
- *Обновление.* При обновлении записи в дочерней таблице можно попытаться некорректно изменить значение внешнего ключа. Операция может привести к нарушению ссылочной целостности.

- *Удаление.* При удалении записи в дочерней таблице ссылочная целостность не нарушается.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

1. Обновление записей в родительской таблице.
2. Удаление записей в родительской таблице.
3. Вставка записей в дочерней таблице.
4. Обновление записей в дочерней таблице.

Основные стратегии поддержания ссылочной целостности

Существуют две основные стратегии поддержания ссылочной целостности.

RESTRICT (ОГРАНИЧИТЬ) — не разрешать выполнение операции, приводящей к нарушению ссылочной целостности.

CASCADE (КАСКАДНОЕ ИЗМЕНЕНИЕ) — разрешить выполнение требуемой операции, но внести при этом необходимые изменения в связанных таблицах так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительской таблице и каскадно выполняется в дочерних таблицах. В реализации этой стратегии имеется одна тонкость, заключающаяся в том, что дочерние таблицы сами могут быть родительскими для некоторых третьих таблиц. При этом может дополнительно потребоваться выполнение какой-либо стратегии и для этой связи и т. д. Если при этом какая-либо из каскадных операций (любого уровня) не может быть выполнена, то необходимо отказаться от первоначальной операции и вернуть базу данных в исходное состояние. Это сложная стратегия, но она не нарушает связей между родительскими и дочерними таблицами.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

Дополнительные стратегии поддержания ссылочной целостности

IGNORE (ИГНОРИРОВАТЬ) — разрешить выполнять операцию без проверки ссылочной целостности. В этом случае в дочерней таблице могут появляться некорректные значения внешних ключей, вся ответственность за целостность базы данных ложится на программиста или пользователя.

SET NULL (ЗАДАТЬ ЗНАЧЕНИЕ NULL) — разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на null-значения. Эта стратегия имеет два недостатка. Во-первых, для нее требуется разрешение на использование null-значений. Во-вторых, записи дочерней таблицы теряют связь с записями родительской таблицы. Установить, с какой записью родительской таблицы были связаны измененные записи дочерней таблицы, после выполнения операции уже нельзя.

SET DEFAULT (ЗАДАТЬ ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ) — разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. Достоинство этой стратегии по сравнению с предыдущей в том, что она позволяет не пользоваться null-значениями. Установить, с какими записями родительской таблицы были связаны измененные записи дочерней таблицы, после выполнения такой операции тоже нельзя.

На рис. 7 представлен пример реляционной базы, содержащей сведения отдела кадров по работникам предприятия, в которой для каждой таблицы показан список ее полей и показаны связи между таблицами по простому ключу — значению поля tabn.

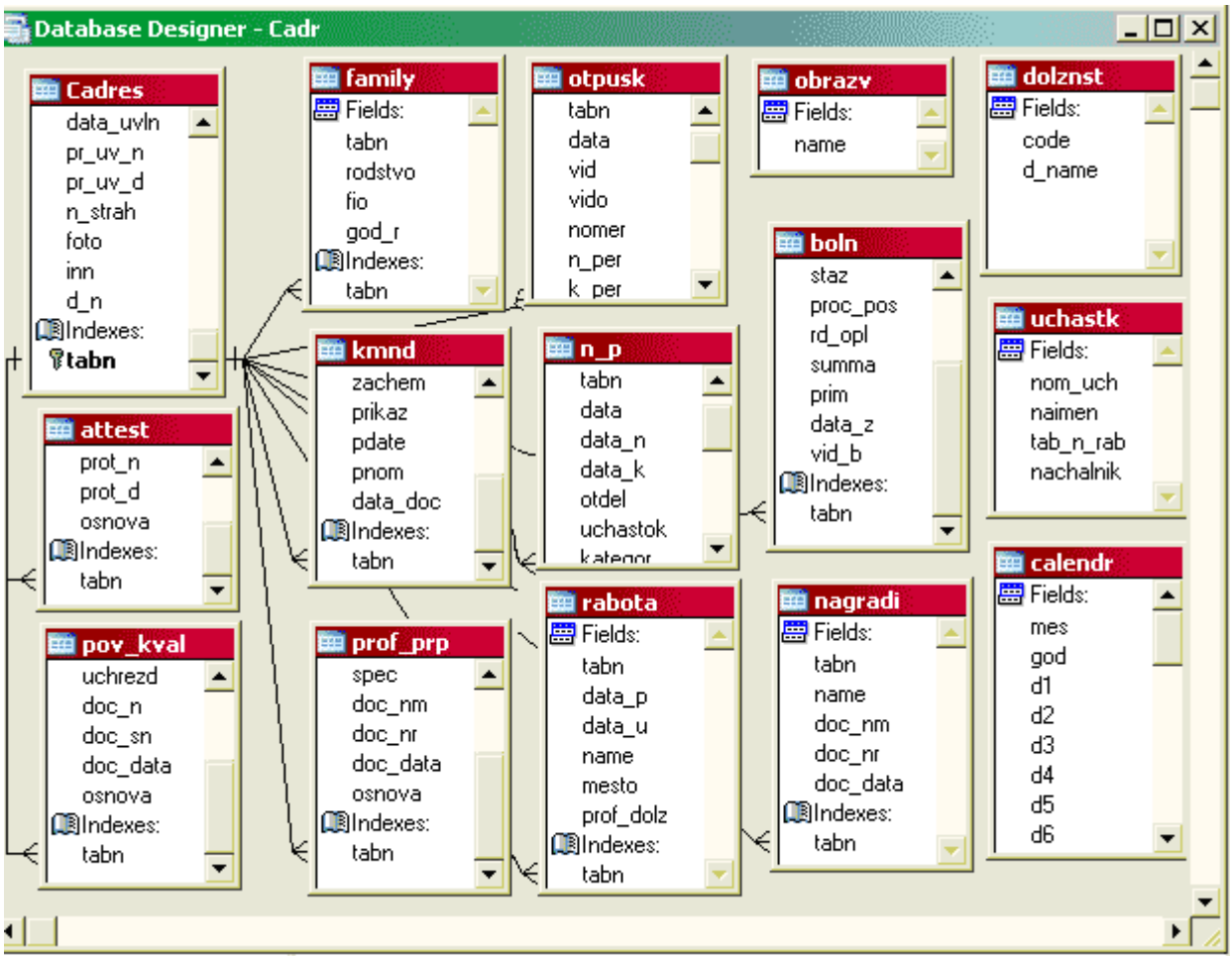

Рис. 7. Схема реляционной базы данных

Начиная с 1980-х г., одновременно с широким распространением персональных компьютеров, большое распространение получили так называемые «настольные» реляционные СУБД (Desktop Databases), такие как dBase, FoxBase (его более поздние версии — FoxPro и Visual FoxPro), Paradox, Access. Наиболее распространенным форматом таблиц подобных реляционных баз стал *.dbf, с которым работали dBase, FoxBase, а также Clipper — система написания программ (в режиме строкового компилятора) для работы с базами данных. В последующем некоторые из них стали полноценными сетевыми СУБД, работающими не только в различных операционных системах в архитектуре «файл-сервер», но и имеющими возможности для работы с серверами баз данных в архитектуре «клиент-сервер», а также разработки и использования html-страниц для работы с базами данных.

Все СУБД для ПК можно подразделить на три вида:

1. Системы управления базами данных в буквальном смысле этого термина, для которых работа с базами возможна только после запуска в работу этой системы без возможности создания автономных программ, работающих с базами. К этим системам относятся: Access, Paradox, dBase.
2. Системы, имеющие как средства для работы с базами данных, так и возможности разработки исполняемых в операционной системе пользовательских программ (приложений), т. е. средства разработчика программ — FoxPro.
3. Системы для разработки пользовательских программ для работы с базами данных — Clipper, Clarion.

Все подобные СУБД имеют в своем составе средства для:

- создания баз данных и модификации их структуры; создания индексных файлов;
- работы с базами в табличном формате или в виде стандартной формы с расположением полей построчно; при этом возможно редактирование данных, добавление записей, удаление записей, работа с данными из нескольких таблиц базы, вычисление сложных выражений для заданных условий и пр.;
- разработки экранных форм, имеющих, кроме редактируемых полей, связанных с базой данных или с переменными памяти, также элементы управления разного вида в виде кнопок; более сложные объекты типа раскрывающихся списков и пр.;
- генерации печатных форм — отчетов сложной структуры с группировкой данных, с получением расчетных значений и итогов по группам и общих итогов (сумма, количество, среднее, максимальное, минимальное, и пр.);
- разработки программных модулей для сложной обработки данных;
- генерации запросов очень сложной структуры — с использованием данных из различных баз, заданием

сложных условий отбора данных, сортировки и группировки данных.

В системах, ориентированных на разработчика, дополнительно возможны разработка меню, справочной системы и проекта, включающего все перечисленные выше компоненты и компилирующегося в исполняемую программу.

Важными факторами, определяющими выбор СУБД, являются:

- Формат базы данных, обеспечивающий возможность обмена информацией с другими приложениями операционной системы. Одним из самых распространенных форматов является dbf-формат, с которым работают dBase, FoxBase, FoxPro, Visual FoxPro, Clipper. Его «понимают» все приложения MS Office. Данные из этих баз можно переносить в Word, Excel, Access. Свои собственные форматы данных имеют Clarion, Paradox, Access.
- Обеспечение секретности и конфиденциальности данных имеют системы, не ориентированные на разработчика программ: Access, Paradox. Однако этот фактор может быть реализован при хранении данных на выделенном сервере, где права различных пользователей легко разграничить.

Все современные СУБД поддерживают режимы работы в локальной сети многих пользователей с одной базой данных. Некоторые имеют «мастеры», «построители» и «генераторы выражений» для ускоренной разработки баз данных, экранных форм, отчетов, стандартных приложений.

Последние версии СУБД, разработанные для работы в ОС Windows 95, относятся к классу RAD-систем (Rapid Application Development) — средства быстрой разработки приложений — и имеют объектно-ориентированный язык программирования. Это такие системы, как Visual FoxPro, MS Access, Visual dBase и др.

Постреляционные базы данных

В настоящее время известны также так называемые постреляционные СУБД, в основе которых лежат модель данных в виде многомерных таблиц (например, в системе Cache фирмы InterSystems Corporation) и широкое использование принципов объектно-ориентированного подхода при организации баз данных и программировании.

Серверы баз данных

В локальных и глобальных компьютерных сетях широко применяются серверы: компьютеры и программные средства для обслуживания клиентов — рабочих станций и/или других серверов.

Примерами серверов могут быть:

- файловый сервер, поддерживающий общее хранилище файлов для всех рабочих станций;
- интернет-сервер, обеспечивающий предоставление информации в глобальной сети Интернет;
- почтовый сервер, обеспечивающий работу с электронной почтой;
- сервер баз данных — СУБД, которая принимает запросы по локальной сети и возвращает информацию, соответствующую запросу.

Термин «сервер баз данных» обычно используют для обозначения всей СУБД, основанной на архитектуре «клиент-сервер», включая и серверную, и клиентскую части. Наиболее распространенными серверами являются в настоящее время Microsoft SQL Server, Oracle, IBM DB2 Universal DataBase, Informix и др. Размер одной базы данных на этих серверах может достигать миллиона терабайт.

2.5. Распределенные базы данных

Основная задача систем управления распределенными базами данных состоит в обеспечении средства интеграции локальных баз данных, располагающихся в некоторых узлах вычислительной сети, с тем, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим базам данных как к единой базе.

Возможны однородные и неоднородные распределенные базы данных. В однородном случае каждая локальная база данных управляется одной и той же СУБД. В неоднородной системе локальные базы данных могут относиться даже к разным моделям данных. Сетевая интеграция неоднородных баз данных — очень сложная проблема. Многие решения известны на теоретическом уровне, но пока не удается справиться с главной проблемой: недостаточной эффективностью интегрированных систем. Более успешно решается промежуточная задача — интеграция неоднородных SQL-ориентированных систем. Этому в большой степени способствует стандартизация языка SQL.

Примером распределенной СУБД может служить System R*. В данной системе разработчики прикладных программ и конечные пользователи остаются в среде языка SQL. Возможность использования SQL основывается на обеспечении System R* прозрачности местоположения данных. Система автоматически обнаруживает текущее местоположение упоминаемых в запросе пользователя объектов данных; одна и та же прикладная программа, включающая предложения SQL, может быть выполнена в разных узлах сети. При этом в каждом узле сети на этапе компиляции запроса выбирается наиболее оптимальный план выполнения запроса в соответствии с расположением данных в распределенной системе.

Выводы

Рассмотрены терминология, теоретические вопросы реляционнных баз данных, классификация баз данных, модели данных.

Вопросы для самопроверки

1.Дать определение базы данных.
2. Перечислить типы баз данных.
3. Дать понятие иерархической и сетевой моделей данных.
4. Пояснить принцип построения реляционной базы данных.
5. Пояснить принцип «сущность-связь».
6. Дать понятие нормализации.
7. Пояснить принцип построения распределенной базы данных.

Библиография

1. Т. С. Карпова. Базы данных: Модели, разработка, реализация. СПб.: Питер, 2001.
2. Грошев Александр Сергеевич www.intut.ru
3. Дейт К. Введение в системы баз данных. М.: Наука, 1980.