КОНСПЕКТ ЗАНЯТИЯ № 1 ВТОРОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ»

3. МОДЕЛИ ДАННЫХ И МОДЕЛИ БАЗЫ ДАННЫХ

Каждая система БД реализует ту или иную модель данных. *Модель данных* определяет правила порождения допустимых для системы видов структур данных, возможные операции над такими структурами, классы представимых средствами системы ограничений целостности данных. Таким образом, модель данных задает границы множества всех конкретных БД, которые могут быть созданы средствами этой системы.

Описание выбранной предметной области в терминах модели данных позволяет получить модель БД. Обычно выделяют три уровня моделей БД [1,4].

Инфологическая модель отражает информацию о предметной области без ориентации на конкретную СУБД (или даже на тип предполагаемой к использованию СУБД). В связи с этим, некоторые авторы [1] говорят о существовании инфологической модели предметной области, а не БД.

Даталогическая модель БД – модель логического уровня, представляющая собой отображение логических связей между элементами данных независимо от их содержания и среды хранения. Эта модель строится в терминах информационных единиц, допустимых в той СУБД, в среде которой будет создаваться БД. Этап создания данной модели называется даталогическим или логическим проектированием.

Физическая модель БД строится с учетом возможностей по организации и хранению данных, предоставляемых СУБД и используемой программно-аппаратной платформой. Она, в частности, определяет используемые запоминающие устройства и способы организации данных в среде хранения.

При проектировании БД, первой строится инфологическая модель, после чего – даталогическая, и только после нее – физическая. Более подробно эти этапы будут рассмотрены в следующих разделах пособия.

Но вернемся к рассмотрению моделей данных. Разные авторы приводят несколько различающиеся перечни существующих моделей данных. Например, в [6] предлагается такой список моделей данных и периодов времени, когда в их разработке были получены основные результаты:

- иерархическая (*англ.* hierarchical), конец 1960-х и 1970-е;
- сетевая (*англ.* network), 1970-е;
- реляционная (*англ.* relational), 1970-е и начало 1980-х;
- сущность-связь (*англ.* entity-relationship), 1970-е;
- расширенная реляционная (*англ.* extended relational), 1980-е;
- семантическая (*англ.* semantic), конец 1970-х и 1980-е;
- объектно-ориентированная (*англ.* object-oriented), конец 1980-х – начало 1990-х;
- объектно-реляционная (*англ.* object-relational), конец 1980-х – начало 1990-х;
- полуструктурированная (*англ.* semi-structured), конец 1990-х до настоящего времени.

Первыми появились модели данных, основанные на теории графов – иерархическая и сетевая. Они будут более подробно рассматриваться ниже. Следующей появилась разработанная Эдгаром Коддом (Edgar Codd) реляционная модель данных, основанная на математической теории множеств. На сегодняшний день она является самой распространенной, и поэтому будет рассматриваться наиболее подробно. Вопросам, связанным с реляционной моделью и логическим проектированием реляционных баз данных, посвящены четвертая и пятая главы пособия.

Модель «сущность-связь» была предложена Питером Ченом (Peter Chen) в 1976 году, в качестве унифицированного способа описания предметной области. Как самостоятельная модель данных (в соответствии с приведенным выше определением), она развития не получила, но стала основой для создания инфологических моделей БД.

Этап инфологического проектирования будет рассмотрен в шестой главе.

Семантическая модель, также как и модель «сущность-связь», используется для построения инфологических моделей. Только в этом случае, пользовательские данные представляются в виде набора семантических объектов. *Семантический объект* [7] – это именованная совокупность атрибутов, которая в достаточной степени описывает отдельный феномен (объект, явление и т. п.).

Объектно-ориентированная и объектно-реляционная модели данных появились в результате распространения объектно-ориентированного подхода в программировании. Объектная модель данных предлагает рассматривать БД как множество объектов, обладающих свойствами инкапсуляции, наследования и т. д. В 1989 году был опубликован «Манифест систем объектно-ориентированных баз данных», а в 1991 году был образован консорциум ODMG (*от англ.* Object Data Management Group), который занялся разработкой стандартов. В 2000 году была опубликована версия стандарта The Object Data Standard: ODMG 3.0, а в 2001 году группа прекратила свою деятельность. Примерно в то же время велась активная работа по адаптации реляционной модели к требованиям объектно-ориентированного подхода к разработке ПО, что привело к появлению объектно-реляционной модели данных. Позднее, объектные расширения были введены в стандарт языка SQL.

К полуструктурированным относят данные, в которых можно выделить некоторую структуру, но она недостаточно строгая по сравнению с реляционными структурами данных (или структурами других традиционных моделей данных) [6]. Наиболее ярким примером полуструктурированных данных являются XML-документы (*от англ.* eXtensible Markup Language – расширяемый язык разметки). Действительный (*англ.* valid) XML-документ должен соответствовать определенному формату описания (схеме), где задана структура документа, допустимые названия элементов, атрибутов и т. д. Формат XML широко используется для обмена данными между приложениями и его поддержка обеспечивается многими СУБД.

3.1. ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Исторически первыми появились СУБД, реализующие иерархическую модель данных: первая коммерческая СУБД IBM IMS относится к этому типу. В иерархической системе данные организованы в наборы древовидных структур (иерархий). Основными информационными единицами являются поле, сегмент (запись), связь, база данных.

Поле данных (или просто «поле», в некоторых изданиях, аналогично сетевой модели, также называется «элемент») – минимальная именованная единица данных, доступная пользователю с помощью СУБД.

Сегмент или *запись* составляет основную единицу обработки БД: записи запоминаются, извлекаются, удаляются. Определяют тип и экземпляр записи (сегмента). *Тип записи* – это именованная совокупность полей данных с указанием их типов, *экземпляр записи* (или просто запись) – это некоторая совокупность значений элементов в последовательности, соответствующей определению типа. Иными словами, тип записи задает все множество подобных объектов, а экземпляр – конкретный объект из этого множества. Например, тип объекта (тип сегмента) – стол. Тогда экземпляр – тот конкретный стол, за которым вы работаете, описываемый его набором характеристик: цвет, размеры, материал и т. д.

Для того, чтобы можно было однозначно различать записи, каждый тип записи должен иметь ключ. *Ключ* – это набор полей, однозначно идентифицирующий экземпляр записи. Например, в записи, описывающей человека, таким ключом может быть номер паспорта.

Связь (англ. link*)* – иерархическое отношение между записями двух типов; некоторые авторы по аналогии с сетевой моделью пользуются термином «групповое отношение». Связи при графическом изображении обозначаются дугами ориентированного графа, типы записей – вершинами. Такое изображение структуры БД называется диаграммой Бахмана (автор – американский исследователь Чарльз Бахман (Charles Bachman), один из пионеров в области баз данных, получивший за свои работы в 1973 году премию Тьюринга). Дуги графа,

обозначающие логические связи, являются направленными. Запись, из которой выходит стрелка, является логически исходной; запись, в которую приходит стрелка – логически подчиненной.

Тип связи определяется ее именем и задает свойства, общие для всех экземпляров связи данного типа. Экземпляр связи задается логически исходной записью («владельцем») и множеством (возможно пустым) подчиненных записей. Таким образом, каждой подчиненной записи в иерархической модели может соответствовать только одна исходная; одной исходной записи может соответствовать несколько подчиненных.

В иерархической модели сегменты и связи между ними создают древовидные структуры (деревья). В каждом дереве существует только одна запись, которая не связана ни с какой исходной записью, она называется корневой. Таким образом, дерево – совокупность корневой записи и множества подчиненных записей.

Схема иерархической БД представляет собой совокупность отдельных деревьев, каждое дерево в рамках модели называется *физической базой данных*. Каждая физическая БД удовлетворяет следующим иерархическим ограничениям [4]:

- в каждой физической БД существует один корневой сегмент, то есть сегмент, у которого нет логически исходного (родительского) типа сегмента;
- каждый логически исходный сегмент может быть связан с произвольным числом логически подчиненных сегментов;
- каждый логически подчиненный сегмент может быть связан только с одним логически исходным (родительским) сегментом.

Набор всех экземпляров записей, подчиненных одному экземпляру корневой записи, называется *физической записью*.

Экземпляры-потомки одного типа, связанные с одним экземпляром записи-предка называются *близнецами*. В примере на рис. 3.1, б записи-близнецы типа «Нач.отдела» – Петров и Сидорова, типа «Служащий» – Николаев и Васильев.

а)

б)

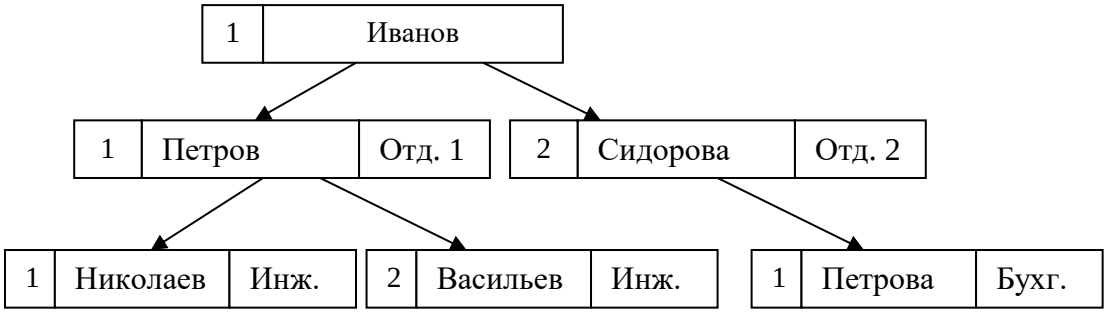

Рис. 3.1. а) пример структуры дерева; б) пример отдельной физической записи

Чтобы записи могли быть однозначно идентифицированы, применяется следующая схема. Корневая запись каждого дерева обязательно содержит ключ с уникальными значениями. Ключи некорневых записей должны быть уникальны только среди «близнецов». Каждая запись идентифицируется полным составным ключом, под которым понимается совокупность ключей записей, начиная от корневой и далее вниз по иерархическому пути до искомой записи.

Для связей в иерархической модели данных обеспечивается *автоматический режим включения* и *фиксированное членство*. Это означает, что для запоминания любой некорневой записи, в БД должна существовать соответствующая ей исходная запись. Подчиненная запись жестко закрепляется за исходной, а при удалении исходной записи, автоматически удаляются все подчиненные ей.

Основной единицей работы в иерархической модели является запись, над ней могут производиться следующие операции.

Операция *добавить* (INSERT) позволяет занести в БД новую запись (с учетом ограничений, связанных с требованиями относительно

уникальности ключа в рамках всей БД для корневой записи и среди «близнецов» – для подчиненной).

Операция *обновить* (UPDATE) дает возможность изменить значения данных предварительно извлеченной записи. Ключевые данные записи не могут подвергаться обновлению: в подобных случаях, старая запись должна быть удалена, а новая, содержащая измененные данные, – запомнена.

Операция *удалить* (DELETE) служит для исключения из БД некоторой записи и всех подчиненных ей.

Операция *извлечь* (GET) имеет несколько модификаций: извлечь по значению ключа, извлечь следующую запись (в порядке левостороннего обхода дерева), извлечь следующую, удовлетворяющую дополнительному условию и т. д.

Надо отметить, что в любом случае, обработка начинается с корневой записи, и доступ к некорневым записям осуществляется по иерархическому пути.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Диго С.М. Базы данных. Проектирование и создание: Учебно-методический комплекс. – М.: Изд. центр ЕАОИ, 2008. – 171 с.

4. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2002 – 304 с.

6. N. Sharma, R. F. Chong и др. Database fundamentals / [Электронный ресурс] URL: https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Database+fundamentals

7. Крёнке Д. Теория и практика построения баз данных. 8-е изд.– СПб.: Питер, 2003. (Серия «Классика computer science»). – 800 с.