

4 Модели и типы данных

Хранимые в базе данные имеют определенную логическую структуру — иными словами, описываются некоторой моделью представления данных (моделью данных), поддерживаемой СУБД. К числу классических относятся следующие модели данных:

- иерархическая,
- сетевая,
- реляционная.

Кроме того, в последние годы появились и стали более активно внедряться на практике следующие модели данных:

- постреляционная,
- многомерная,
- объектно-ориентированная.

Разрабатываются также всевозможные системы, основанные на других моделях данных, расширяющих известные модели. В их числе можно назвать объектно-реляционные, дедуктивно-объектно-ориентированные, семантические, концептуальные и ориентированные модели. Некоторые из этих моделей служат для интеграции баз данных, баз знаний и языков программирования. В некоторых СУБД поддерживается одновременно несколько моделей данных.

§4.1 Иерархическая модель

В иерархической модели связи между данными можно описать с помощью упорядоченного графа (или дерева). Упрощенно представление связей между данными в иерархической модели показано на рисунке 1 (стр. 10 пособия).

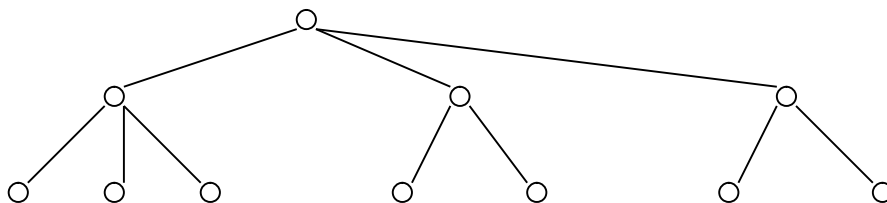


Рисунок 1 — Представление связей в иерархической модели

Для описания структуры (схемы) иерархической БД на некотором языке программирования используется тип данных «дерево».

Тип «дерево» схож с типами данных «структура» языков программирования ПЛ/1 и Си и «запись» языка Паскаль. В них допускается вложенность типов, каждый из которых находится на некотором уровне.

Тип «дерево» является составным. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Простая «запись» состоит из одного типа, например, числового, а составная «запись» объединяет некоторую совокупность типов, например, целое, строку символов и указатель (ссылку). Пример типа «дерево» как совокупности типов показан на рисунке 2 (стр. 11 пособия).



Рисунок 2 — Пример типа «дерево»

Корневым называется тип, который имеет подчиненные типы и сам не является подтипом. *Подчиненный* тип (подтип) является *потомком* по отношению к типу, который выступает для него в роли предка (родителя). Потомки одного и того же типа являются близнецами по отношению друг к другу.

В целом тип «дерево» представляет собой иерархически организованный набор типов «запись».

Иерархическая БД представляет собой упорядоченную совокупность экземпляров данных типа «дерево» (деревьев), содержащих экземпляры типа «запись» (записи). Часто отношения родства между типами переносят на отношения между самими записями. Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание БД. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо.

В иерархических СУБД может использоваться терминология, отличающаяся от приведенной. Так, в системе IMS понятию «запись» соответствует термин «сегмент», а под «записью БД» понимается вся совокупность записей, относящаяся к одному экземпляру типа «дерево».

Данные в базе с приведенной схемой (рисунок 2) могут выглядеть, например, как показано на рисунке 3 (стр. 12 пособия).

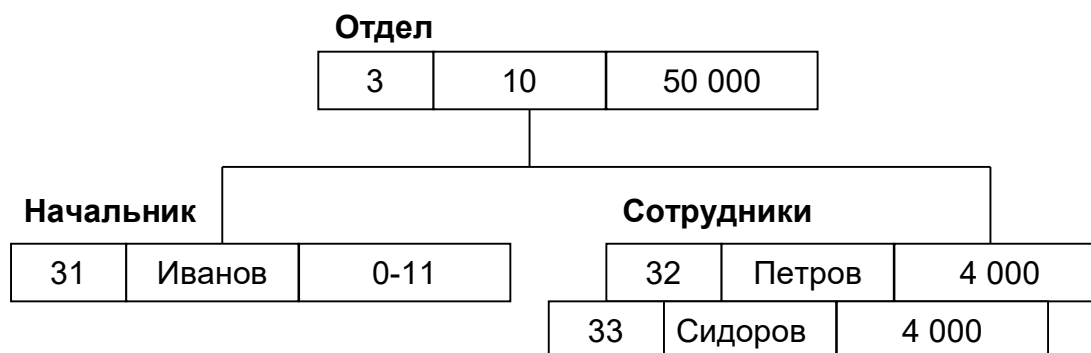


Рисунок 3 — Данные в иерархической базе

Для организации *физического* размещения иерархических данных в памяти ЭВМ могут использоваться следующие группы методов:

- представление линейным списком с последовательным распределением памяти (адресная арифметика, левосписковые структуры);
- представление связными линейными списками (методы, использующие указатели и справочники).

К основным операциям манипулирования иерархически организованными данными относятся следующие:

- поиск указанного экземпляра БД (например, дерева со значением 3 в поле Отд_Номер);
- переход от одного дерева к другому;
- переход от одной записи к другой внутри дерева (например, к следующей записи типа Сотрудники);
- вставка новой записи в указанную позицию;
- удаление текущей записи и т. д.

В соответствии с определением типа «дерево», можно заключить, что между предками и потомками автоматически поддерживается контроль целостности связей. Основное правило контроля целостности формулируется следующим образом: потомок не может существовать без родителя, а у некоторых родителей может не быть потомков. Механизмы поддержания целостности связей между записями различных деревьев отсутствуют.

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения

основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

На иерархической модели данных основано сравнительно ограниченное количество СУБД, в числе которых можно назвать зарубежные системы IMS, PC/Focus, Team-Up и Data Edge, а также отечественные системы Ока, ИНЭС и МИРИС.

§4.2 Сетевая модель

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных (рисунок 4, стр. 13 пособия). Наиболее полно концепция сетевых БД впервые была изложена в Предложениях группы КОДАСИЛ (CODASYL — Conference of Data System Languages) в 1975 году.

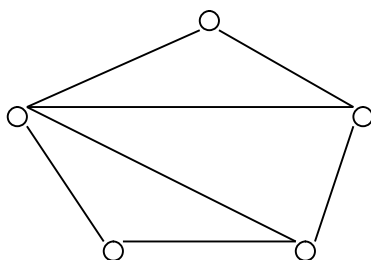


Рисунок 4 — Представление связей в сетевой модели

Для описания схемы сетевой БД используется две группы типов: «запись» и «связь». Тип «связь» определяется для двух типов «запись»: предка и потомка. Переменные типа «связь» являются экземплярами связей.

Сетевая БД состоит из набора записей и набора соответствующих связей. На формирование связи особых ограничений не накладывается. Если в иерархических структурах запись-потомок могла иметь только одну запись-предка, то в сетевой модели данных запись-потомок может иметь произвольное число записей-предков (сводных родителей).

Пример схемы простейшей сетевой БД показан на рисунке 5 (стр. 14 пособия). Типы связей здесь обозначены надписями на соединяющих типы записей линиях.

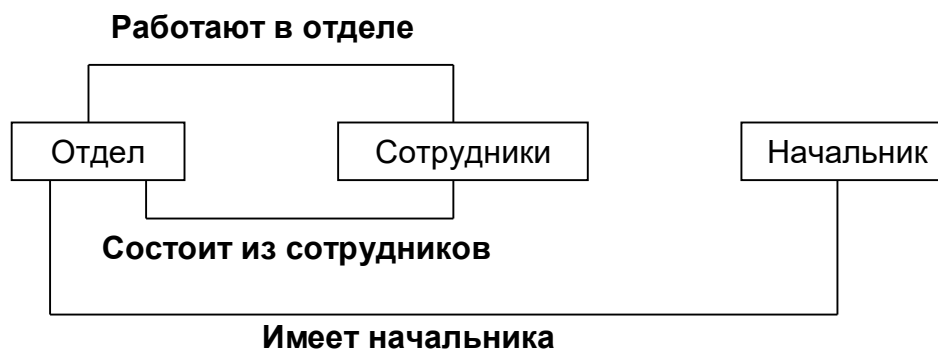


Рисунок 5 — Пример схемы сетевой БД

В различных СУБД сетевого типа для обозначения одинаковых по сути понятий зачастую используются различные термины. Например, такие, как элементы и агрегаты данных, записи, наборы, области и т. д.

Физическое размещение данных в базах сетевого типа может быть организовано практически теми же методами, что и в иерархических базах данных.

К числу важнейших операций манипулирования данными баз сетевого типа можно отнести следующие:

- поиск записи в БД;
- переход от предка к первому потомку;
- переход от потомка к предку;
- создание новой записи;
- удаление текущей записи;
- обновление текущей записи;
- включение записи в связь;
- исключение записи из связи;
- изменение связей и т. д.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Системы на основе сетевой модели не получили широкого распространения на практике. Наиболее известными сетевыми СУБД являются следующие: IDMS, db_VistaIII, СЕТЬ, СЕТОР и КОМПАС.

§4.3 Реляционная модель

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение (relation).

Отношение представляет собой множество элементов, называемых кортежами. Наглядной формой представления отношения является двумерная таблица. Таблица имеет строки (записи) и столбцы (колонки). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам — атрибуты отношения.

С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно: деление одного объекта (явления, сущности, системы и проч.), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удастся описать более сложные логические структуры данных из предметной области, применяют связывание таблиц.



Описанную выше иерархическую модель (рисунки 2 и 3) можно представить в виде трех таблиц реляционной модели. Таблица ОТДЕЛЫ содержит данные о номере отдела (Отд_Номер), количестве сотрудников в отделе (Отд_Размер) и суммарной зарплате сотрудников отдела (Отд_Зарп), причем столбец

Отд_Номер является первичным ключом. В таблице НАЧАЛЬНИКИ содержатся данные о личном номере начальников (Нач_Номер), их фамилии и телефоны, первичным ключом является столбец Нач_Номер. В таблице СОТРУДНИКИ соответственно содержатся номера, фамилии и зарплата сотрудников каждого отдела; первичный ключ — Сотр_Номер. Таблицы НАЧАЛЬНИКИ и СОТРУДНИКИ связаны с таблицей ОТДЕЛЫ по полю Отд_Номер, т.е. это поле является еще и внешним ключом.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Основными недостатками реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Примерами зарубежных реляционных СУБД являются следующие: MySQL (TcX), Paradox и dBASE for Windows (Borland), Visual FoxPro и Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) и Oracle (Oracle).

Заметим, что последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционными. Примером такой системы можно считать продукты Oracle 8.x. Системы предыдущих версий вплоть до Oracle 7.x считаются «чисто» реляционными.

§4.4 Постреляционная модель

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

На рисунке 6 на примере информации о накладных и товарах для сравнения приведено представление одних и тех же данных с помощью реляционной (а) и постреляционной (б) моделей. Таблица НАКЛАДНЫЕ содержит данные о

номерах накладных (ИНВ_N) и номерах покупателей (ПОКУП_N). В таблице НАКЛАДНЫЕ_ТОВАРЫ содержатся данные о каждой из накладных: номер накладной (ИНВ_N), название товара (НАЗВАНИЕ) и количество товара (КОЛИЧ). Таблица НАКЛАДНЫЕ связана с таблицей НАКЛАДНЫЕ_ТОВАРЫ по полю ИНВ_N.

а)

НАКЛАДНЫЕ

ИНВ_N	ПОКУП_N
73	23
74	45
88	23

НАКЛАДНЫЕ_ТОВАРЫ

ИНВ_N	НАЗВАНИЕ	КОЛИЧ
73	Ручка	3
73	Линейка	2
74	Тетрадь	1
74	Карандаш	6
74	Блокнот	2
88	Папка	1

б)

НАКЛАДНЫЕ

ИНВ_N	ПОКУП_N	НАЗВАНИЕ	КОЛИЧ
73	23	Ручка	3
		Линейка	2
74	45	Тетрадь	1
		Карандаш	6
		Блокнот	2
88	23	Папка	1

Рисунок 6 — Структуры данных реляционной и постреляционной моделей

Как видно из рисунка, по сравнению с реляционной моделью в постреляционной модели данные хранятся более эффективно, а при обработке не требуется выполнять операцию соединения данных из двух таблиц.

На длину полей и количество полей в записях таблицы не накладывается требование постоянства. Это означает, что структура данных и таблиц имеет большую гибкость.

Поскольку постреляционная модель допускает хранение в таблицах ненормализованных данных, возникает проблема обеспечения целостности и непротиворечивости данных. Эта проблема решается включением в СУБД механизмов, подобных хранимым процедурам в клиент-серверных системах.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

Рассмотренная постреляционная модель данных поддерживается СУБД uniVers. К числу других СУБД, основанных на постреляционной модели данных, относятся также системы Bubba и Dasdb.

§4.5 Многомерная модель

Многомерный подход к представлению данных в базе появился практически одновременно с реляционным, но реально работающих многомерных СУБД (МСУБД) до настоящего времени было очень мало. С середины 90-х годов интерес к ним стал приобретать массовый характер.

Толчком послужила в 1993 году программная статья одного из основоположников реляционного подхода Э. Кодда. В ней сформулированы 12 основных требований к системам класса OLAP (*OnLine Analytical Processing — оперативная аналитическая обработка*), важнейшие из которых связаны с возможностями концептуального представления и обработки многомерных данных. Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

В развитии концепций ИС можно выделить следующие два направления:

- системы оперативной (транзакционной) обработки;
- системы аналитической обработки (системы поддержки принятия решений).

Реляционные СУБД предназначались для информационных систем оперативной обработки информации и в этой области были весьма эффективны. В системах аналитической обработки они показали себя несколько неповоротливыми и недостаточно гибкими. Более эффективными здесь оказываются многомерные СУБД (МСУБД).

Многомерные СУБД являются узкоспециализированными СУБД, предназначенными для интерактивной аналитической обработки информации. Раскроем основные понятия, используемые в этих СУБД: агрегируемость, историчность и прогнозируемость данных.

Агрегируемость данных означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь-оператор, управляющий, руководитель.

Историчность данных предполагает обеспечение высокого уровня статичности (неизменности) собственно данных и их взаимосвязей, а также обязательность привязки данных ко времени.

Статичность данных позволяет использовать при их обработке специализированные методы загрузки, хранения, индексации и выборки.

Временная привязка данных необходима для частого выполнения запросов, имеющих значения времени и даты в составе выборки. Необходимость упорядочения данных по времени в процессе обработки и представления данных пользователю накладывает требования на механизмы хранения и доступа к информации. Так, для уменьшения времени обработки запросов желательно, чтобы данные всегда были отсортированы в том порядке, в котором они наиболее часто запрашиваются.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными.

По сравнению с реляционной моделью многомерная организация данных обладает более высокой **наглядностью** и **информативностью**. Для иллюстрации на рисунке 7 приведены реляционное (а) и многомерное (б) представления одних и тех же данных об объемах продаж автомобилей.

Если речь идет о многомерной модели с размерностью больше двух, то не обязательно визуально информация представляется в виде многомерных объектов (трех-, четырех- и более мерных гиперкубов). Пользователю и в этих случаях более удобно иметь дело с двумерными таблицами или графиками. Данные при этом представляют собой «вырезки» (точнее, «срезы») из многомерного хранилища данных, выполненные с разной степенью детализации.

а)

МОДЕЛЬ	МЕСЯЦ	ОБЪЕМ
«Жигули»	июнь	12
«Жигули»	июль	24
«Жигули»	август	5
«Москвич»	июнь	2
«Москвич»	июль	18
«Волга»	июль	19

б)

МОДЕЛЬ	Июнь	Июль	Август
«Жигули»	12	24	5
«Москвич»	2	18	нет
«Волга»	нет	19	нет

Рисунок 7 — Реляционное и многомерное представление данных

Рассмотрим основные понятия многомерных моделей данных, к числу которых относятся измерение и ячейка.

Измерение (Dimension) — это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются Дни, Месяцы, Кварталы и Годы. В качестве географических измерений широко употребляются Города, Районы, Регионы и Страны. В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (Cell) или показатель — это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой. В зависимости от того, как формируются значения некоторой ячейки, обычно она может быть переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно) либо формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

В примере на рисунке 7 (б) каждое значение ячейки Объем продаж однозначно определяется комбинацией временного измерения (Месяц продаж) и модели автомобиля. На практике зачастую требуется большее количество измерений. Пример трехмерной модели данных приведен на рисунке 8.



Рисунок 8 — Трехмерная модель данных

Измерения: время (год) — 2000, 2001, 2002; менеджер — Петров, Смирнов, Яковлев; модель — “Волга”, “Жигули”, “Москвич”.
Показатель: объем продаж.

В существующих МСУБД используются два основных варианта (схемы) организации данных: гиперкубическая и поликубическая.

В *поликубической* схеме предполагается, что в БД может быть определено несколько гиперкубов с различной размерностью и с различными измерениями в качестве граней. Примером системы, поддерживающей поликубический вариант БД, является сервер Oracle Express Server.

В случае *гиперкубической* схемы предполагается, что все показатели определяются одним и тем же набором измерений. Это означает, что при наличии нескольких гиперкубов БД все они имеют одинаковую размерность и совпадающие измерения. Очевидно, в некоторых случаях информация в БД может быть избыточной (если требовать обязательное заполнение ячеек).

В случае многомерной модели данных применяется ряд специальных операций. «Срез» (Slice) представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений. Формирование «срезов» выполняется для ограничения используемых пользователем значений, так как все значения гиперкуба практически никогда одновременно не используются. Например, если ограничить значения измерения «модель автомобиля» в гиперкубе (рисунок 8) маркой «Жигули», то получится двумерная таблица продаж этой марки автомобиля различными менеджерами по годам.

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. При организации обработки аналогичных данных на основе реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размерности БД и существенное увеличение затрат оперативной памяти на индексацию.

Недостатком многомерной модели данных является ее громоздкость для простейших задач обычной оперативной обработки информации.

Примерами систем, поддерживающими многомерные модели данных, являются Essbase (Arbor Software), Media Multi-matrix (Speedware), Oracle Express Server (Oracle) и Cache (InterSystems). Некоторые программные продукты, например Media/ MR (Speedware), позволяют одновременно работать с многомерными и с реляционными БД. В СУБД Cache, в которой внутренней моделью данных является многомерная модель, реализованы три способа доступа к данным: прямой (на уровне узлов многомерных массивов), объектный и реляционный.

§4.6 Объектно-ориентированная модель

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями БД и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Стандартизованная объектно-ориентированная модель описана в рекомендациях стандарта ODMG-93 (*Object Database Management Group — группа управления объектно-ориентированными базами данных*). Реализовать в полном объеме рекомендации ODMG-93 пока не удастся. Для иллюстрации ключевых идей рассмотрим несколько упрощенную модель объектно-ориентированной БД.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, строковым — string) или типом, конструируемым пользователем (определяется как class).

Значением свойства типа string является строка символов. Значение свойства типа class есть объект, являющийся экземпляром соответствующего класса. Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.

Пример логической структуры объектно-ориентированной БД библиотечного дела приведен на рисунке 9.

Здесь объект типа БИБЛИОТЕКА является родительским для объектов-экземпляров классов АБОНЕНТ, КАТАЛОГ и ВЫДАЧА. Различные объекты типа КНИГА могут иметь одного или разных родителей. Объекты типа КНИГА, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств удк, название и автор.

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, усиленные объектно-ориентированными механизмами. Ограниченно могут применяться операции, подобные командам SQL (например, для создания БД).

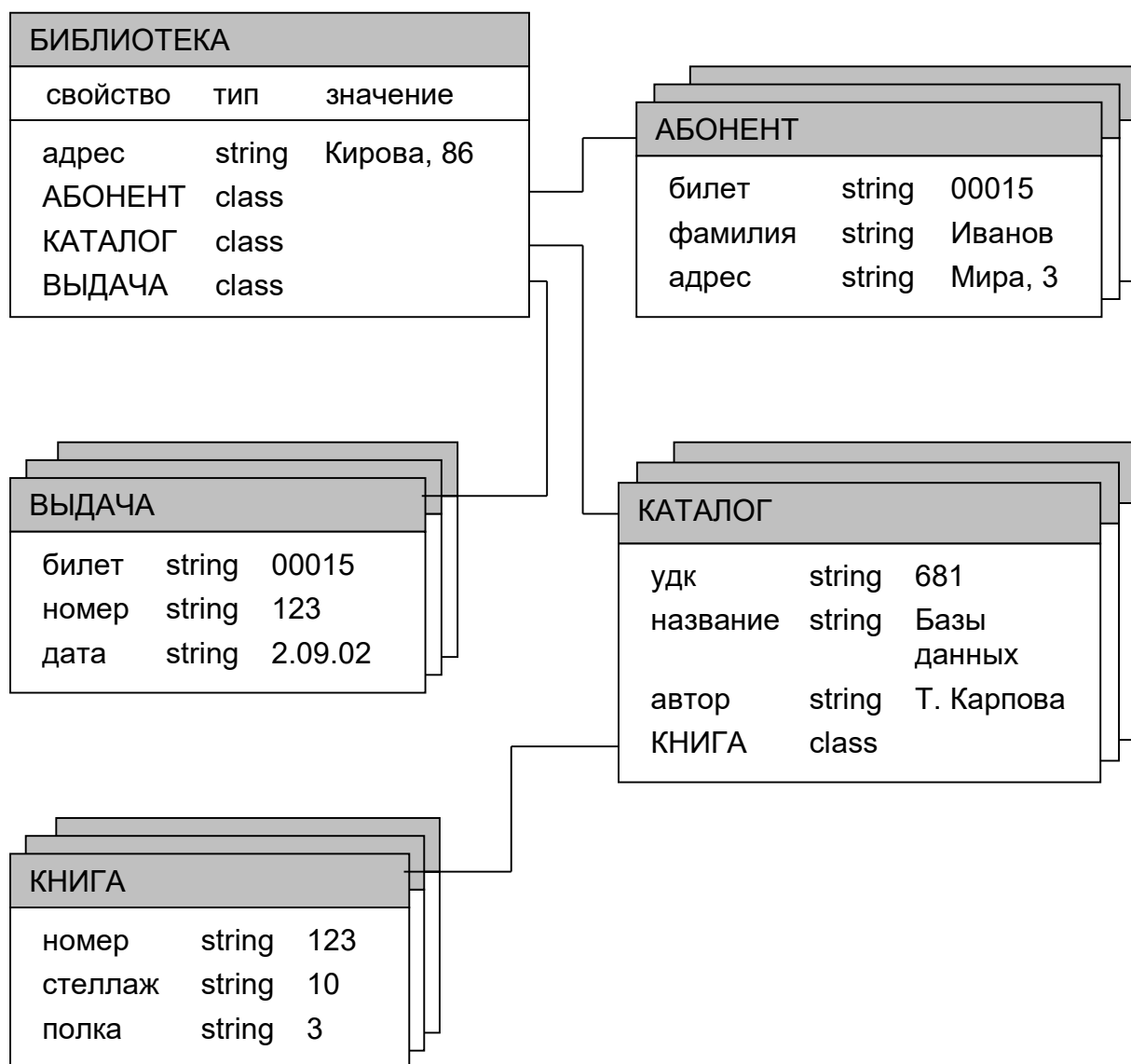


Рисунок 9 — Логическая структура БД «Библиотека»

Создание и модификация БД сопровождается автоматическим формированием и последующей корректировкой индексов (индексных таблиц), содержащих информацию для быстрого поиска данных.

Поиск в объектно-ориентированной БД состоит в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в БД. Определяемый пользователем объект, называемый объектом-целью (свойство объекта имеет тип goal), в общем случае может представлять собой подмножество всей хранимой в БД иерархии объектов. Объект-цель, а также результат выполнения запроса могут храниться в самой базе.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

В 90-е годы существовали экспериментальные прототипы объектно-ориентированных систем управления базами данных. В настоящее время такие системы получили широкое распространение, в частности, к ним относятся следующие СУБД: POET (POET Software), Jasmine (Computer Associates), Versant (Versant Technologies), O2 (Ardent Software), ODB-Jupiter (научно-производственный центр «Интелтек Плюс»), а также Iris, Orion и Postgres.