Ранние подходы к организации систем управления база данных (СУБД)

Цель: сформировать представление о понятии модели данных. Раскрыть суть классических и современных моделей представления данных, описать их достоинства и недостатки

Понятие модели данных

В классической теории баз данных модель данных есть формальная теория представлния и обработки данных в системе управления базами данных.

Длительное время термин «модель данных» использовался без формального определения. Одним из первых специалистов, который достаточно точно определил это понятие, был Э. Кодд. Хотя понятие модели данных было введено Коддом, наиболее распространненая трактовка модели данных принадлежит Кристоферу Дейту. Согласно его пределению модель данных включает три аспекта:

- ✓ **аспект структуры –** методы описания типов и логических структур данных в базе данных;
- ✓ **аспект манипуляции –** методы манипулирования данными;
- ✓ **аспект целостности –** методы описания и поддержки целостности базы данных.

Аспект структуры определяет, что логически представляет собой база данных.

Аспект манипуляции содержит спецификацию одного или нескольких языков, предназначенных для написания запросов к базе данных. Эти языки могут быть абстрактными, не обладающие точно проработанным синтаксисом. Основное назначение манипуляционной части модели данных – обеспечить эталонный язык БД, уровень выразительности которого должен поддерживаться в реализациях СУБД, соответствующих данной модели. Аспект манипуляции определяет способы перехода между состояниями базы данных – способы модификации данных и способы извлечения данных из базы данных.

Аспект целостности определяет средства описаний корректных состояний базы данных. В целостной части модели данных описываются механизмы ограничений целостности, которые обязательно должны поддерживаться во всех реализациях СУБД, соответствующие данной модели.

В теории баз данных модель данных является ядром любой базы данных. Модель данных описывает информационные объекты предметной области, взаимосвязи между ними и позволяет:

- ✓ определить границу между логическим и физическим аспектами управления базой данных (независимость данных);
- ✓ обеспечить конечным пользователям и программистам возможность и средства общего понимания смысла данных (коммуникабельность);

- ✓ определить языковые понятия высокого уровня, обеспечивающие возможность выполнения однотипных операций над большими совокупностями записей (в общем случае разнотипных данных) как единую операцию (обработка множеств).

В модели данных описывается некоторый набор понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими базы данных, если они основываются на этой модели. Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык.

К числу классических моделей данных относятся следующие:

- ✓ системы, основанные на инвертированных списках;
- ✓ иерархическая;
- ✓ сетевая;
- ✓ реляционная.

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

- ✓ многомерная;
- ✓ объектно-ориентированная.

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

Далее коротко будут рассмотрены ранние (дореляционные) СУБД, так как:

- эти системы исторически предшествовали реляционным, и для правильного понимания причин повсеместного перехода к реляционным системам нужно знать хотя бы что-нибудь про их предшественников;
- внутренняя организация реляционных систем во многом основана на использовании методов ранних систем;
- эти системы активно использовались в течение многих лет, дольше, чем используется какая-либо из реляционных СУБД. Некоторые из ранних систем используются даже в наше время, накоплены громадные базы данных, поэтому актуальна проблема использование этих систем совместно с современными системами.

Наиболее общих характеристики ранних систем:

- не основывались на каких-либо абстрактных моделях. Понятие модели данных появилось в области БД только с возникновением реляционного подхода.
- доступ к БД производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в БД, используя языки программирования. Интерактивный доступ к БД поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом.

- доступ к данным на уровне записей заставляли пользователя самого производить всю оптимизацию доступа к БД, без какой-либо поддержки системы.

Основные особенности систем, основанных на инвертированных списках

К числу наиболее известных и типичных представителей таких систем относятся Datacom/DB компании Applied Data Research, Inc. (ADR), ориентированная на использование на машинах основного класса фирмы IBM, и Adabas компании Software AG.

Организация доступа к данным на основе инвертированных списков используется практически во всех современных реляционных СУБД, но в этих системах пользователи не имеют непосредственного доступа к **инвертированным спискам (индексам).**

База данных, организованная с помощью инвертированных списков, похожа на реляционную БД, но с тем отличием, что хранимые таблицы и пути доступа к ним видны пользователям. При этом:

- строки таблиц упорядочены системой в некоторой физической последовательности.
- физическая упорядоченность строк всех таблиц может определяться и для всей БД (так делается, например, в Datacom/DB).
- для каждой таблицы можно определить произвольное число ключей поиска, для которых строятся индексы. Эти индексы автоматически поддерживаются системой, но явно видны пользователям.

Поддерживаются два класса операторов.

1. Операторы, устанавливающие адрес записи, среди которых:
   - прямые поисковые операторы (например, найти первую запись таблицы по некоторому пути доступа);
   - операторы, находящие запись в терминах относительной позиции от предыдущей записи по некоторому пути доступа.
2. Типичные операторы над адресуемыми записями
   - LOCATE FIRST - найти первую запись таблицы T в физическом порядке; возвращает адрес записи;
   - RETRIVE - выбрать запись с указанным адресом;
   - UPDATE - обновить запись с указанным адресом;
   - DELETE - удалить запись с указанным адресом и т.п.

Общие правила определения целостности БД отсутствуют. В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу.

Теоретико-графовые модели данных

1. Иерархические системы

Типичным представителем (наиболее известным и распространенным) является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г. До сих пор существуют БД, которые поддерживаются этой СУБД.

В основе иерархических СУБД лежит довольно простая модель данных, которую можно представить себе **в виде дерева (графа)**. Дерево состоит из **узлов (вершин)**, каждая из которых кроме одной, имеет единственную родительскую вершину и несколько или ни одной дочерних. Вершина, не имеющая родительской, называется **корнем дерева**. Вершины, не имеющие дочерних, называются **листьями**. Остальные вершины являются ветвями.

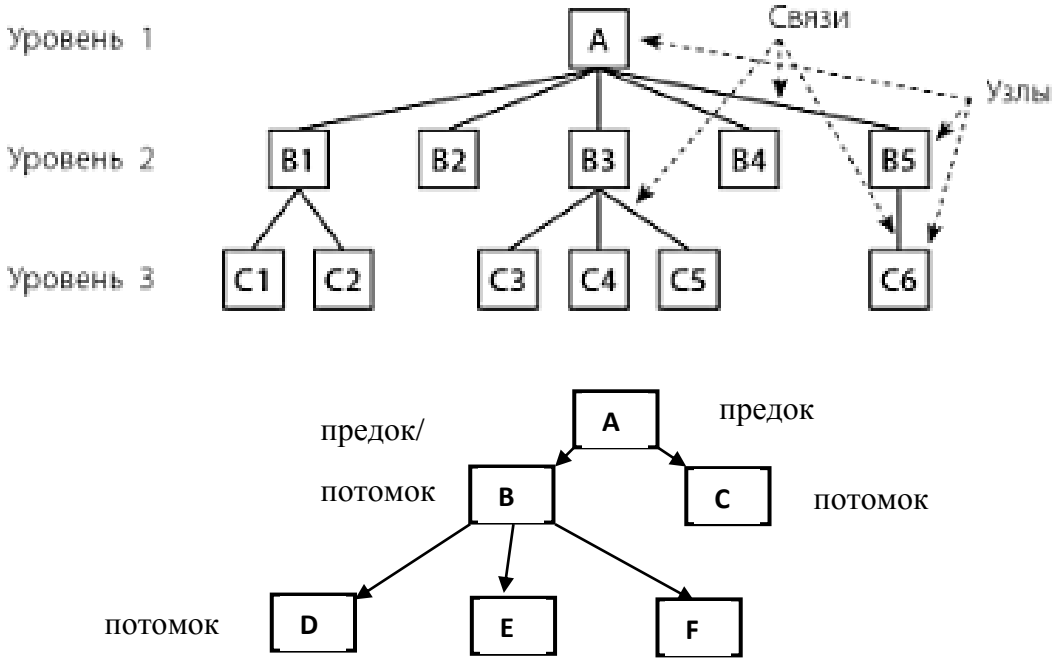

Рисунок 1 Схемы иерархической модели данных

Все экземпляры потомка с общим предком называются **близнецами.** Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

Иерархические базы данных наиболее пригодны для моделирования структур, по своей природе являющихся иерархическими. В качестве примеров можно привести воинские подразделения или сложные механизмы, состоящие из более простых узлов, которые в свою очередь тоже можно подвергнуть декомпозиции. Тем не менее существует значительное количество структур, не сводящихся к простой иерархии.

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными.

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

Иерархические СУБД быстро прошли пик популярности, которая обусловливалась их ранним появлением на рынке. Затем их недостатки сделали их неконкурентоспособными, и в настоящее время иерархическая модель представляет исключительно исторический интерес.

Иерархической базой данных является файловая система, состоящая из корневой директории, в которой имеется иерархия поддиректорий и файлов.

2. Сетевые модели данных

Типичным представителем является Integrated Database Management System (IDMS) появилась в 70-х годах.

Сетевой подход к организации данных является расширением иерархического. Сетевая БД состоит из набора записей и набора связей между этими записями. На формирование связи особых ограничений не накладывается. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.

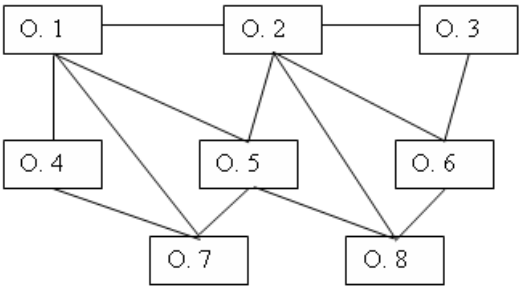

Рисунок 2 Схема сетевой модели

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.

Реляционная модель данных

Реляционная модель относится к теоретико-множественным моделям данных и на сегодняшний день является самой популярной как среди пользователей, так и среди профессиональных разработчиков. Реляционная модель появилась в следствие стремления сделать базу данных как можно более гибкой. Данная модель представила простой и эффективный механизм поддержания связей данных. Реляционная система управления базами

данных представляет собой одну из наиболее удачных технологий обработки данных. Большая часть данных в мире хранится в реляционной форме.

Реляционная модель единственная из всех обеспечивает единообразие представления данных. В реляционной модели данных информационные объекты представляются в виде таблиц. Каждая таблица базы данных представляется в виде совокупности строк и столбцов. Строки (записи) соответствуют экземпляру объекта, конкретному событию или явлению. Столбцы (поля) соответствуют атрибутам, т.е. признакам, характеристикам или параметрам объекта, события или какого-либо явления.

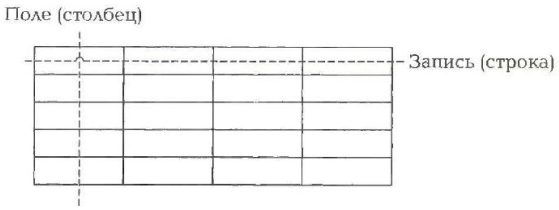

Рисунок 3 Схема таблицы базы данных

В каждой таблице базы данных необходимо наличие хотя бы одного уникального ключа.

Уникальный ключ – это поле или набор полей, однозначно идентифицирующий каждый экземпляр объекта (запись). Уникальных ключей в таблице может быть несколько. Один их уникальных ключей определяют как **первичный ключ.** Так же как и для уникальных ключей, в таблице не допускается наличие двух и более записей с одинаковыми значениями первичного ключа. Он должен быть минимально достаточным, а значит, не содержать полей, удаление которых не отразится на его уникальности.

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

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 реализации на ЭВМ. Именно простота и понятность для пользователя стали основными причинами их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми. Основным недостатком реляционной модели является сложность описания иерархических и сетевых связей.

Последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называются объектно-реляционными. Примером такой продукции можно считать продукты Oracle 8.x.

Постреляционная модель данных

Классическая реляционная модель данных предполагает неделимость данных, хранящихся в зависимых таблицах, и не допускает избыточности данных. Постреляционная модель данных представляет собой расширенную реляционную модель данных, где нет ограничения неделимости данных. Эта модель допускает составные (многозначные) поля, т.е. значение поля может состоять из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

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

Недостатком постреляционной модели данных является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных. Эта проблема решается включением в СУБД соответствующих механизмов.

Многомерная модель данных

В развитии информационных систем можно выделить следующие два направления: системы оперативной обработки и системы аналитической обработки (системы поддержки принятия решений). Реляционные СУБД предназначаются для информационных систем оперативной обработки информации и в этой области являются наиболее эффективными. Однако в системах аналитической обработки они показали себя недостаточно гибкими. Здесь более эффективными являются многомерные системы управления данными. Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия ращения. По сравнению с реляционной моделью данных многомерная организация данных обладает более высокой наглядностью и информативностью.

В многомерных СУБД данные организованы в виде упорядоченных многомерных массивов – гиперкубов. Они обеспечивают более быструю реакцию на запросы данных за счет того, обращения поступают относительно небольшими блоками данных, необходимых для конкретной группы пользователей.

Технология многомерных баз данных – достаточно новая технология. Несмотря на то, что многомерный подход к представлению данных в базе появился практически одновременно с реляционным, реально работающих СУБД сначала было очень мало. Только с середины 1990-х гг. интерес к ним стал приобретать массовый характер.

Многомерность модели данных означает не визуальное представление, а логическую организацию структуры информации при описании и в операциях манипулирования данными. На рисунке ниже приведены реляционное (*а*) и многомерное (*б*) представление одних и тех же данных (в таблицах хранятся данные об объемах продажи мебели).

Модель	Месяц	Объем
Шкаф	Июнь	12
Шкаф	Июль	24
Шкаф	Август	5
Диван	Июнь	2
Диван	Июль	18
Стол	Июль	19

а

Модель	Июнь	Июль	Август
Шкаф	12	24	5
Диван	2	18	
Стол		19	

б

Рисунок 4 Реляционное и многомерное представление данных

Далее рассмотрены основные понятия многомерных моделей данных – измерение, мера и срез.

Измерение – это множество однотипных данных, образующих одну из граней гиперкуба. Другими словами, измерения являются осями многомерной системы координат. Примерами наиболее часто используемых временных измерений являются «Дни», «Месяцы», «Кварталы» и «Года». 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.

Мера (ячейка, показатель) – это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определении как числовой. Это могут быть объемы продаж в количественном иди денежном выражении, товарные остатки на складе, издержки и т.п.

Февраль

Январь

	Склад 1	Склад 2	Склад 3	Склад 4
Мебель	1 000	200	100	200
Аксессуары	500	50	25	50
Ткань	500	50	25	50

Рисунок 5 Пример гиперкуба

В качестве мер в трехмерном кубе, изображенном выше, использованы суммы продаж, а в качестве измерений – время, товар, склад. Измерения представлены на определенных уровнях группировки: товары группируются по видам, а данные о времени совершения операций – по месяцам.

Куб совсем не обязательно должен быть трехмерным. Он может быть и двух- и многомерным – в зависимости от решаемой задачи.

Однако куб сам по себе для анализа не пригоден. Даже трехмерный куб сложно отобразить на экране компьютера так, чтобы были видны значения

интересующих мер. Для визуализации данных, хранящихся в кубе, применяются, как правило, привычные двумерные табличные представления, имеющие сложные иерархические заголовки строк и столбцов. Эта операция называется «разрезанием» куба. Термин это опять же образный.

Аналитик как бы берет и разрезает куб по интересующим его направлениям. Пользователь получает естественную, интуитивно понятную модель данных. Это дает возможность получить сводные или детальные сведения о бизнес-процессе и осуществлять прочие манипуляции, которые придут в голову пользователю в процессе анализа. Этим способом аналитик получает двумерный срез куба и с ним работает. Примерно так же лесорубы считают годовые кольца на спиле. Соответственно, «неразрезанными», как правило, остаются только два измерения – по числу измерений таблицы.

Срез представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений. Формирование «срезов» выполняется для ограничения используемых значений, как как все значения гиперкуба практически никогда одновременно не используются. На рисунке ниже показа срез куба для одной меры – «Количество» и двух «неразрезанных» измерений – «Склад» и «Время».

	Склад 1	Склад 2	Склад 3	Склад 4
Январь	2 000	300	150	300
Февраль	3 000	1 000	200	1 000

Рисунок 6 Двумерный срез куба для одной меры

На рисунке ниже представлено лишь одно «неразрезанное» измерение – «Склад», но здесь отображаются значения двух мер – «Количество», «Сумма».

	Склад 1	Склад 2	Склад 3	Склад 4
Количество	200	100	50	300
Сумма	3 000	1 000	200	1 000

Рисунок 7 Двумерный срез куба для нескольких мер

Двумерное представление куба возможно и тогда, когда «неразрезанными» остаются и более двух измерений. При этом на осях среза (строках и столбцах) будут размещены два и более измерений «разрезаемого» куба, как показано ниже.

	Январь				Февраль			
	Склад 1	Склад 2	Склад 3	Склад 4	Склад 1	Склад 2	Склад 3	Склад 4
Количество	200	100	50	300	200	100	50	300
Сумма	2000	1000	2000	3000	2000	10000	200	1000

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

Основные преимущества многомерных СУБД следующие:

- ✓ общая простота системы, что позволяет осуществлять быстрое встраивание технологий многомерных СУБД в приложения. Системы, основанные на основе многомерных баз данных, требуют меньше специальных навыков по разработке и администрированию;
- ✓ относительно низкая общая стоимость владения, а также быстрый возврат инвестиций;
- ✓ в случае использования многомерных СУБД поиск и выборка данных осуществляется значительно быстрее, чем при многомерном концептуальном взгляде на реляционную базу данных, так как многомерная база данных обеспечивает оптимизированный доступ к запрашиваемым ячейкам;
- ✓ многомерные СУБД легко справляются с задачами включения в информационную модель разнообразных строенных функций, тогда как объективно существующие ограничения языка SQL делают выполнение этих задач на основе реляционных СУБД достаточно сложными, а иногда и невозможными.

При хранении данных в многомерных структурах возникает потенциальная проблема «разбухания» за счет хранения пустых значений. Ведь если в многомерном массиве зарезервировано место под все возможные комбинации меток измерений, а реально заполнена лишь малая часть (например, ряд товаров продается только в небольшом числе регионов), то большая часть куба будет пустовать, хотя место будет занято. Современные системы умеют справляться с этой проблемой.

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

Объектно-ориентированная модель данных

Объектно-ориентированные системы управления базами данных являются результатом совмещения возможностей и особенностей баз данных и возможностей объектно-ориентированных языков программирования. Объектно-ориентированные СУБД позволяют работать с объектами баз данных так же, как с объектами с объектно-ориентированных языках программирования. Они расширяют языки программирования, прозрачно вводя долговременные данные, управление параллельной работой с данными, восстановление данных, ассоциированные запросы и другие возможности.

Появление объектно-ориентированных СУБД было вызвано потребностями программистов на объектно-ориентированных языках в средствах для хранения объектов, не помещавшихся в оперативной памяти

компьютера. Также была важна задача сохранения состояния объектов между повторными запусками прикладной программы. Поэтому большинство объектно-ориентированных систем управления базами данных представляют собой библиотеку, процедуры управления данными которой включаются в прикладную программу.

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

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

Пример логической структуры объектно-ориентированной БД «Колледж» представлена на рисунке ниже.

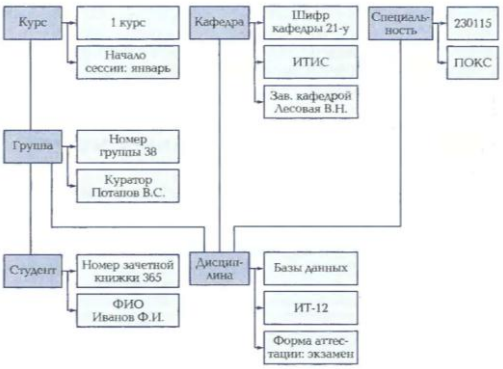

Рисунок 8 Логическая структура объектно-ориентированной базы данных

Логическая структура объектно-ориентированной базы данных внешне похожа на структуру иерархической базы данных. Основное отличие между ними состоит в методах манипулирования данными. Для выполнения действий над данными в рассматриваемой модели базы данных применяются логические операции, усиленные объектно-ориентированным механизмом.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность 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 данных являются высокая понятийная сложность и неудобство обработки.

Реакцией производителей реляционных СУБД на возрастающую популярность объектных технологий стало появление объектно-реляционных баз данных, так называемых универсальных серверов. Объектно-реляционная СУБД реляционная СУБД, поддерживающая некоторые технологии, реализующие объектно-ориентированный подход: объекты, классы, классы и наследование реализованы в структуре баз данных и языке запросов.

Объектно-реляционными СУБД являются, например, широко известные Oracle Database, Informix, DB2.

Основными преимуществами объектно-реляционной модели данных являются возможность повторного и совместного использования компонентов.

Очевидным недостатком подхода с использованием объектно-реляционной СУБД являются сложность и связанные с ней повышенные расходы. Простота и ясность, присущие реляционной модели, теряются при использовании подобных типов расширения.

Вопросы и задания:

1. Дайте определение понятию «модель данных».
2. Перечислите аспекты модели данных.
3. Перечислите классические и современные модели представления данных.
4. Укажите достоинства и недостатки иерархической модели данных.
5. Укажите достоинства и недостатки систем, основанных на инвертированных списках.
6. Поясните, как организуется физическое размещение данных в БД иерархической структуры.
7. Охарактеризуйте сетевую модель данных.
8. Охарактеризуйте реляционную модель данных.
9. Назовите особенности постреляционной модели данных.
10. Поясните, где находит применение многомерные модели данных.
11. Укажите достоинства и недостатки многомерных моделей данных.
12. Охарактеризуйте многомерную модель данных.
13. Назовите и объясните смысл операций, выполняемых над данных в случае многомерной модели данных.
14. Приведите примеры многомерных таблиц.
15. Укажите достоинства и недостатки объектно-ориентированной модели представления данных.
16. Раскройте смысл понятий, используемых в многомерных СУБД: «измерение», «ячейка», «срез».
17. Заполните таблицу, представленную ниже, данными.

Название модели данных	Тип модели данных	Основные понятия модели данных	Период возникновения	Направление концепции информационных систем	Примеры	Достоинства	Недостатки

Преподаватель С.В. Банцевич