Содержание:

Введение.

Трехуровневая архитектура ANSI-SPARC.

-Первая попытка создания стандартной терминологии и общей архитектуры СУБД была предпринята в 1971 году группой, называемой DBTG, которая признала необходимость использования двухуровневого подхода, построенного на основе использования системного представления - схемы, и пользовательских представлений -. подсхем. Сходные терминология и архитектура были предложены в 1975 году Комитетом планирования стандартов и норм Национального Института Стандартизации США (ANSI/SPARC), который признал необходимость использования трехуровневого подхода. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления. Ниже перечислено несколько причин, по которым желательно выполнять такое разделение.

-Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое собственное представление о них, должен иметь возможность изменять свое представление о данных, причем это изменение не должно оказывать влияния на других пользователей.

-Пользователи не должны непосредственно иметь дело с такими подробностями физического хранения данных в базе, как индексирование и хеширование.

-Внутренняя структура базы данных не должна зависеть от таких изменений физических аспектов хранения информации, как переключение на новое устройство хранения.

-АБД должен иметь возможность изменять концептуальную или глобальную структуру базы данных без какого-либо влияния на всех пользователей.

-Администратор базы данных (АБД)- должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские

представления.

Внешний уровень и Концептуальный уровень.

-Уровень, на котором воспринимают данные пользователи, называется внешним уровнем, тогда как СУБД и операционная система воспринимают данные на внутреннем уровне. Именно на внутреннем уровне данные реально сохраняются с использованием всех тех структур и файловой организации. Концептуальный уровень представления данных предназначен для отображения внешнего уровня на внутренний и обеспечения необходимой независимости друг от друга.

-Внешний уровень - представление базы данных с точки зрения пользователей. Этот уровень описывает ту часть базы данных, которая относится к каждому пользователю.

-Внешний уровень состоит из нескольких различных внешних представлений базы данных. Внешнее представление содержит только те сущности, атрибуты и связи "реального мира", которые интересны пользователю.

Помимо этого, различные представления могут по-разному отображать одни и те же данные. Некоторые представления могут включать производные или вычисляемые данные, которые не хранятся в базе данных как таковые, а создаются по мере надобности. Представления могут также включать комбинированные или производные данные из нескольких объектов.

-Концептуальный уровень - обобщающее представление базы данных. Этот уровень описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними.

-Промежуточный концептуальный уровень содержит логическую структуру всей базы данных. Фактически, это полное представление требований к данным со стороны организации, которое не зависит от любых соображений относительно способа их хранения. На концептуальном уровне представлены следующие компоненты:

-все сущности, их атрибуты и связи;

-накладываемые на данные ограничения;

-семантическая информация о данных;

-информация о мерах обеспечения безопасности и поддержки целостности данных.

-**Концептуальный уровень поддерживает каждое внешнее представление**, в том смысле, что любые доступные пользователю данные должны содержаться (или могут быть вычислены) на этом уровне. Однако этот уровень не содержит никаких сведений о методах хранения данных.

Внутренний уровень.

-**Внутренний уровень** - физическое представление базы данных в компьютере. Этот уровень описывает, как информация хранится в базе данных.

-**Внутренний уровень описывает**- физическую реализацию базы данных и предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. Он содержит описание структур данных и организации отдельных файлов, используемых для хранения данных в запоминающих устройствах. На этом уровне осуществляется взаимодействие СУБД с методами доступа операционной системы с целью размещения данных на запоминающих устройствах, создания индексов, извлечения данных и т.д. **На внутреннем уровне хранится следующая информация:**

-распределение дискового пространства для хранения данных и индексов;

-описание подробностей сохранения записей;

-сведения о размещении записей;

-сведения о сжатии данных и выбранных методах их шифрования.

-Ниже внутреннего уровня находится физический уровень, который контролируется операционной системой, но под руководством СУБД.

-Физический уровень доступа к данным ниже СУБД состоит только из известных операционной системе элементов.

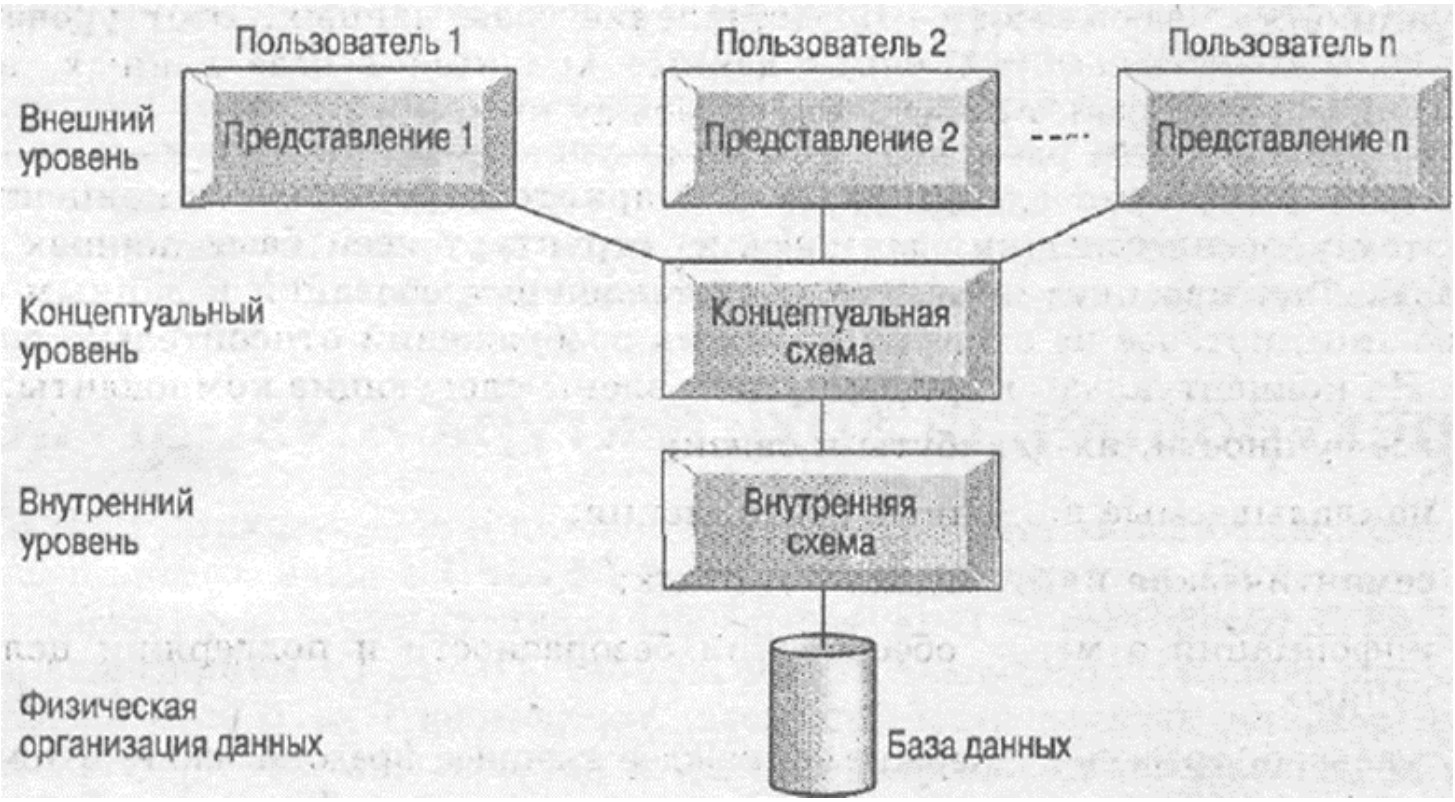

рисунок 1. Трехуровневая архитектура ANSI-SPARC.

Схемы, отображения и экземпляры.

-**Общее описание базы данных называется схемой базы данных.** Существует три различных типа схем базы данных, которые определяются а соответствии с уровнями абстракции трехуровневой архитектуры, как показано на рисунке 1. На самом высоком уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных. На концептуальном уровне описание базы данных называют концептуальной схемой, а на самом низком уровне абстракции — внутренней схемой.

-**Концептуальная схема** - описывает все элементы данных и связи между ними, с указанием необходимых ограничений поддержки целостности данных. Для каждой базы данных имеется только одна концептуальная схема. На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных.

-Она содержит определения хранимых записей, методы представления, описания полей данных, сведения об индексах и выбранных схемах хеширования. Для каждой базы данных существует только одна внутренняя схема.

-СУБД отвечает за установление соответствия между этими тремя типами схем, а также за проверку их непротиворечивости. Иначе говоря, СУБД должна убедиться в том, что каждую внешнюю схему можно вывести на основе концептуальной схемы. Для установления соответствия между любыми внешней и внутренней схемами СУБД должна использовать информацию из концептуальной схемы. Концептуальная схема связана с внутренней схемой посредством концептуально внутреннего отображения. Оно позволяет СУБД найти фактическую запись или набор записей на физическом устройстве хранения, которые образуют логическую запись в концептуальной схеме, с учетом любых ограничений, установленных для выполняемых над данной логической записью операций, также позволяет обнаружить любые различия в именах объектов, именах атрибутов, порядке следования атрибутов, их типах данных и т.д. Наконец, каждая внешняя схема связана с концептуальной схемой с помощью внешне концептуального отображения. С его помощью СУБД может отображать имена пользовательского представления на соответствующую часть концептуальной схемы.

-Эти внешние представления сливаются воедино в одном концептуальном представлении. СУБД поддерживает внешне концептуальное отображение. Затем концептуальный уровень отображается на внутренний уровень, который содержит физическое описание структуры записи концептуального представления. На этом уровне определение структуры формулируется на языке высокого уровня. Эта структура содержит указатель, который позволяет физически связать все записи в единую цепочку. Порядок полей на внутреннем уровне отличается от порядка атрибутов, принятого на концептуальном уровне. Таков механизм, с помощью которого СУБД осуществляет концептуально внутреннее отображение.

-**Важно различать описание базы данных и саму базу данных. Описанием базы данных является схема базы данных.** Схема создается в процессе проектирования базы данных, причем предполагается, что она изменяется достаточно редко. Однако содержащаяся в базе данных информация может меняться часто. Совокупность информации, хранящейся в базе данных в любой определенный момент времени, называется состоянием базы данных. Следовательно, одной и той же схеме базы данных может соответствовать множество ее различных состояний. Схема базы данных иногда называется содержанием базы данных, а ее состояние — детализацией.

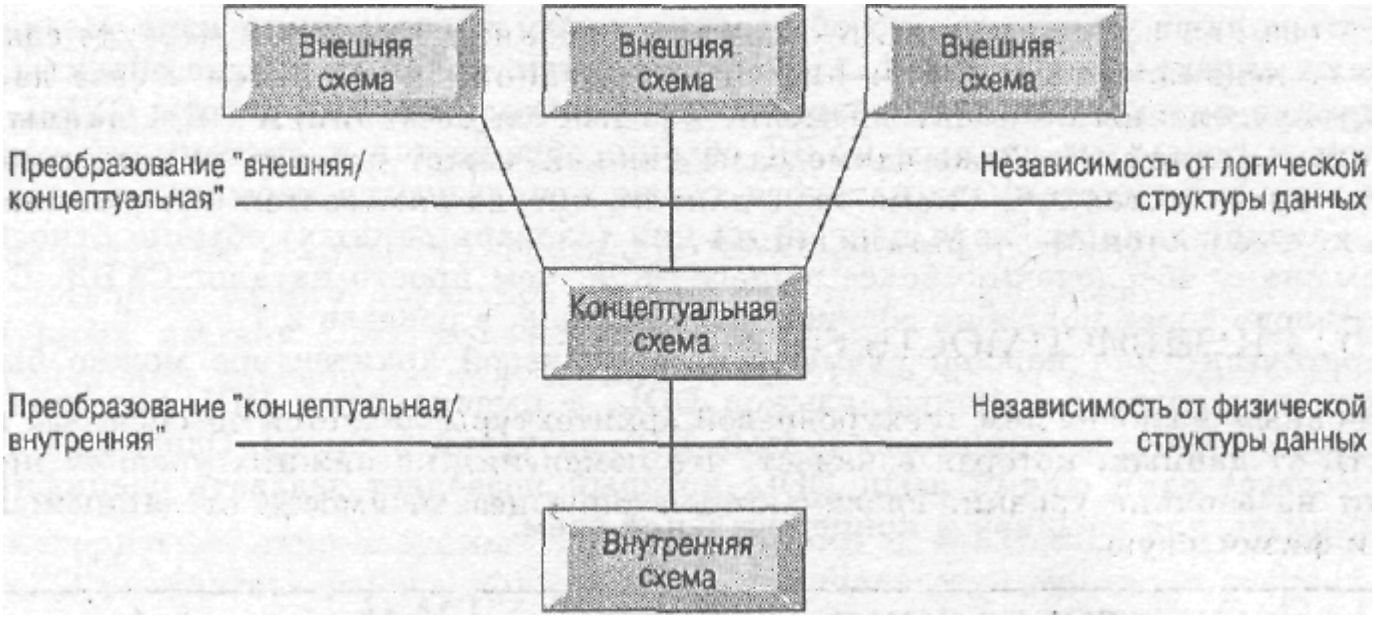

рисунок 2. Реализация независимости от данных в трехуровневой архитектуре ANSI-SPARC.

Независимость от данных.

-Основным назначением трехуровневой архитектуры является обеспечение независимости от данных, которая означает, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

-Логическая независимость от данных - логическая независимость от данных означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему.

-Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без необходимости внесения изменений в уже существующие внешние схемы или переписывания прикладных программ. Ясно, что пользователи, для которых эти изменения предназначались, должны знать о них, но очень важно, чтобы другие пользователи даже не подозревали об этом.

-**Физическая независимость от данных** - физическая независимость от данных означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему.

-Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов

или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы.

-Принятое в архитектуре ANSI-SPARC двухэтапное отображение может сказываться на эффективности работы, но при этом оно обеспечивает более высокую независимость от данных. Для повышения эффективности в модели ANSI-SPARC допускается использование прямого отображения внешних схем на внутренние, без обращения к концептуальной схеме. Однако это снижает уровень независимости от данных, поскольку при каждом изменении внутренней схемы потребуется внесение определенных изменений во внешнюю схему и все зависящие от нее прикладные программы.

Заключение.

Язык определения данных (DDL) и Язык управления данными (DML).

-**Язык определения данных – DDL** - Описательный язык, который позволяет АБД или пользователю описать и поименовать сущности, необходимые для работы некоторого приложения, а также связи, имеющиеся между различными сущностями.

-Схема базы данных состоит из набора определений, выраженных на специальном языке определения данных — DDL. Язык DDL используется как для определения новой схемы, так и для модификации уже существующей. Этот язык нельзя использовать для управления данными.

-Результатом компиляции DDL-операторов является набор таблиц, хранимый в особых файлах, называемых системным каталогом. В системном каталоге интегрированы метаданные — т.е. данные, которые описывают объекты базы данных, а также позволяют упростить способ доступа к ним и управления ими.

-**Язык управления данными DML** - Язык, содержащий набор операторов для поддержки основных операций манипулирования содержащимися в базе данными.

К операциям управления данными относятся:

-вставка в базу данных новых сведений;

-модификация сведений, хранимых в базе данных;

-извлечение сведений, содержащихся в базе данных;

-удаление сведений из базы данных.

Таким образом, Языки DML отличаются базовыми конструкциями извлечения данных. Следует различать два типа языков DML: процедурный и непроцедурный. Основное отличие между ними заключается в том, что процедурные языки указывают то, как можно получить результат оператора языка DML, тогда как непроцедурные языки описывают то, какой результат будет получен. Как правило, в процедурных языках записи рассматриваются по отдельности, тогда как непроцедурные языки оперируют с целыми наборами записей.

-**Процедурный язык DML** - Язык, который позволяет сообщить системе о том, какие данные необходимы, и точно указать, как их можно извлечь.

С помощью процедурного языка DML пользователь, а точнее — программист, указывает на то, какие данные ему необходимы и как их можно получить. Это значит, что пользователь должен определить все операции доступа к данным, которые должны быть выполнены для получения требуемой информации. Обычно такой процедурный язык DML позволяет извлечь запись, обработать ее и, в зависимости от полученных результатов.

-**Непроцедурный язык DM**L - Язык, который позволяет указать лишь то, какие данные требуются, но не то, как их следует извлекать.

Непроцедурные языки DML позволяют определить весь набор требуемых данных с помощью одного оператора извлечения или обновления, С помощью непроцедурных языков DML пользователь указывает, какие данные ему нужны, без определения способа их получения. СУБД транслирует выражение на языке DML в процедуру, которая обеспечивает манипулирование затребованным набором записей. Данный подход освобождает пользователя от необходимости знать детали внутренней реализации структур данных и особенности алгоритмов, используемых для извлечения и возможного преобразования данных. В результате работа пользователя получает определенную степень независимости от данных.