1. Основные понятия баз данных

База данных (БД) - совокупность хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

База данных - организованная структура, предназначенная для хранения информации.

Рассмотрим классификацию БД по способу организации данных. Они делятся на три типа:

1. Неструктурированные.
2. Частично-структурированные.
3. Структурированные.

К неструктурированным БД могут быть отнесены базы, организованные в виде семантических сетей. Частично структурированными можно считать базы данных в виде обычного текста или гипертекстовые системы. Структурированные БД требуют предварительного проектирования и описания структуры БД. Структурированные БД делятся по типу используемой модели баз данных на:

1. Иерархические
2. Сетевые
3. Реляционные
4. Объектно-ориентированные

В иерархической модели записи данных образуют древовидную структуру, при этом каждая запись связана только с одной записью, находящейся на более высоком уровне. Доступ к любой записи осуществляется по строго определенным «веткам» и узлам такого дерева. Иерархические модели хорошо подходят для задач с явно выраженной иерархически соподчиненной структурой информации и запросов. Они обладают низким быстродействием, трудно модифицируемы, но эффективны с точки зрения организации машинной памяти.

В сетевых моделях каждая запись в каждом из узлов сети может быть связана с несколькими другими узлами; кроме данных записи содержат указатели, определяющие местоположение других записей, связанных с ними. Такие модели очень трудно редактировать, например удалять записи, так как вместе с данными нужно редактировать и указатели. Подобные модели хорошо работают в случае решения сетевых,

коммуникационных задач.

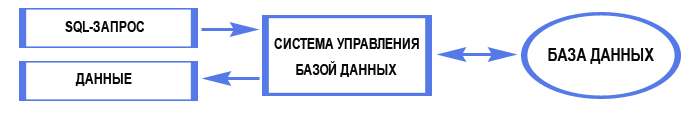
Реляционные модели баз данных основывается на математических принципах, вытекающих непосредственно из теории множеств и логики предикатов. Реляционная модель представляет собой совокупность данных, состоящую из набора двумерных таблиц. В теории множеств таблице соответствует термин отношение (relation), физическим представлением которого является таблица, отсюда и название модели - реляционная. Реляционная модель является удобной и наиболее привычной формой представления данных. При табличной организации данных отсутствует иерархия элементов. Строки и столбцы могут быть просмотрены в любом порядке, поэтому высока гибкость выбора любого подмножества элементов в строках и столбцах. Любая таблица в реляционной базе состоит из строк, которые называют записями, и столбцов, которые называют полями. На пересечении строк и столбцов находятся конкретные значения данных.

Объектно-ориентированные модели БД хранят данные в виде объектов их атрибутов, методов и классов. Обычно их используют, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру.

Система управления БД (СУБД) -  это совокупность языковых и программных средств,  позволяющая создавать БД, обновлять хранимую в ней информацию, обеспечивающая удобный доступ к ней с целью просмотра и поиска

СУБД - это система, позволяющая создавать базы данных и манипулировать сведениями из них. Осуществления доступа к данным СУБД происходит посредством специального языка - SQL.

Рассмотрим простейшую схему работы с БД:



В процессе реализации своих функций СУБД постоянно взаимодействует с базой данных и с другими прикладными программными продуктами пользователя. Современные СУБД имеют следующие возможности:

1. включают язык определения данных, с помощью которого можно определить базу данных, ее структуру, типы данных, а также средства задания ограничения для хранимой информации;
2. позволяют вставлять, удалять, обновлять и извлекать информацию из базы данных посредством языка запросов (SQL);
3. большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами;
4. многопользовательские СУБД имеют развитые средства администрирования баз данных.

СУБД могут использоваться как в однопользовательском, так и в многопользовательском режиме.

Рассмотрим более подробно про реляционную модель базы данных. Основными понятиями реляционных баз данных являются:

1. тип данных;
2. домен;
3. атрибут;
4. кортеж;
5. первичный ключ и отношение;

Понятие тип данных полностью адекватно понятию типа данных в языках программирования. Обычно в реляционных БД типы данных можно разбить на два класса: обычные и специальные. К обычным относятся:

1. символьные;
2. числовые данные;
3. битовые строки;
4. специализированных числовых данных (таких, как «деньги»);

К специальным типам данных можно отнести:

1. дата;
2. время;
3. временной интервал.

Развивается подход к расширению возможностей реляционных систем абстрактными типами данных.

Понятие домен имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена. Наиболее правильное интуитивное понимание домена – допустимое множество значений определенного типа. Данные считаются сравнимыми только в том случае, когда они относятся к одному домену.

Отношение – это именованное множество пар {имя атрибута, имя домена (или типа)}. Если все атрибуты одного отношения определены на разных доменах, целесообразно использовать для наименования атрибутов имена соответствующих доменов.

Кортеж, соответствующий данной схеме отношения, – это набор именованных значений заданного типа.

Обычным представлением отношения является таблица, в заголовке которой указывают наименование отношения, а в строках – кортежи; в этом случае имена атрибутов именуют столбцы таблицы. Поэтому когда говорят «столбец таблицы», имеют в виду «атрибут отношения». Такой терминологии придерживаются в большинстве коммерческих реляционных СУБД.

Реляционнаямодель данных с точки зрения СУБД описывает некоторый набор родовых понятий и признаков, которыми должны обладать сама СУБД и управляемые ею базы данных, если они основываются на этой модели. Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык.

Реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода: структурной, манипуляционной и целостной.

В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных БД, является нормализованное отношение – некоторый определенный набор ограничений, свойственный этому отношению. Нормализация связана с поиском наиболее простой структуры для данного множества данных и имеет дело с зависимостью между атрибутами; она позволяет избежать потери общей информации при удалении или вводе записей. Существует несколько формальных типов нормализации (более пяти).

Манипуляционная часть модели определяет механизм манипулирования реляционными БД – реляционная алгебра и реляционное исчисление, базирующиеся на теории множеств и логическом аппарате исчисления отношений.

В целостной части реляционной модели данных фиксируются два базовых требования целостности: целостности сущностей и целостности по ссылкам. Объекту, или сущности, реального мира в реляционных БД соответствуют записи (кортежи) отношений, и требование целостности состоит в сохранении отличий разных записей этого отношения; говорят, что любое отношение должно обладать первичным ключом.

Второе требование более сложно. Сущности реального мира представляются в реляционной БД в виде записей нескольких отношений. Для связи отношений используется атрибут, который служит внешним ключом. Отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом. Требование целостности по ссылкам состоит в том, что для каждого значения внешнего ключа должна найтись запись с таким же значением первичного ключа в отношении, на которое указывает ссылка, либо значение внешнего ключа должно быть неопределенным (т. е. ни на что не указывать).

Выполнение таких требований чрезвычайно важно при модификации отношений или удалении записей. Поддержке целостности при удалении кортежа служат: запрет на удаление кортежа, на который существуют ссылки; автоматическая замена значения внешнего ключа на неопределенное во всех ссылающихся кортежах; автоматическое удаление всех ссылающихся кортежей.

Реляционные СУБД обладают рядом особенностей, влияющих на организацию внешней памяти. К наиболее важным особенностям можно отнести:

1. наличие двух уровней системы: уровня непосредственного управления данными во внешней памяти и языкового уровня (например, уровня, реализующего язык SQL); тогда подсистема нижнего уровня должна поддерживать во внешней памяти набор базовых структур, конкретная интерпретация которых входит в число функций подсистемы верхнего уровня;
2. информация, связанная с наименованием объектов базы данных и их конкретными свойствами (например, структура ключа-индекса),поддерживается подсистемой языкового уровня;
3. регулярность структур данных во внешней памяти, поскольку основным объектом реляционной модели данных является плоская таблица;
4. для выполнения операторов языкового уровня над одним или несколькими отношениями во внешней памяти поддерживаются дополнительные «управляющие» структуры – индексы;
5. для выполнения требования надежного хранения баз данных поддерживается избыточность хранения данных во внешней памяти.

Следует подчеркнуть, что как бы ни были организованы индексы в конкретной СУБД, их основное назначение состоит в обеспечении эффективного прямого доступа к кортежу отношения по ключу. Обычно индекс определяется для одного отношения, и ключом является значение атрибута (возможно, составного). Организация индексов в больших БД

представляет сложную проблему. Все более популярным подходом к организации индексов является использование техники хэширования. Общей идеей методов хэширования является применение к значению ключа некоторой функции свертки (хэш-функции), вырабатывающей значение меньшего размера. Свертка значения ключа затем применяется для доступа к записи. В самом простом случае свертка ключа используется как адрес в таблице, содержащей ключи и записи.

При проектировании реляционных баз данных используют метод нормальных форм, который является итерационный и заключается в последовательном переводе отношения из 1 нормальной формы в нормальную форму более высокого порядка. Каждая следующая нормальная форма ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями БД, а также сохранении свойств предшествующих нормальных форм.

1. Первая нормальная форма. Отношение находится в первой нормальной форме, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения и не должно быть повторений строк в таблицах.
2. Отношение находится во второй нормальной форме, если оно находится в первой, и каждый не ключевой атрибут неприводимо зависит от первичного ключа. Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.
3. Отношение находится в третьей нормальной форме, когда находится во второй нормальной форме и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

БД в ГИС.

Совокупность цифровых данных о пространственных объектах образует множество пространственных данныхи составляет содержание *баз географических данных,*определяет принципы построения информационного обеспечения ГИС.

Анализ общего состава данных необходим уже на этапе проектирования ГИС и требует ответа, по крайней мере, на ряд основных вопросов:

1. имеется ли возможность сбора, хранения и обновления географических данных?
2. каковы ожидаемые объемы данных и их форматы?
3. Какой объем данных необходимо преобразовать в цифровую форму, сколько времени это займет и сколько будет стоить?
4. каковы качество и надежность данных?
5. какого рода затруднения могут возникнуть при обработке собранных данных?

Выявление географических объектов и явлений и последующий выбор адекватного представления данных о них являются составной частью процесса, именуемого проектированием базы данных (БД).

В ГИС пользователь рассматривает реальный мир через призму тематической базы данных. Измерения и выборки, содержащиеся в базе данных, должны как можно полнее и точнее соответствовать предмету исследования и его основным характеристикам. Представление данных должно учитывать типы их возможных преобразований. К созданию БД ГИС предъявляются высокие требования, связанные с пространственной формой организации и представления данных.

База данных должна быть:

1. Согласованной по времени – хранящиеся в ней количественные данные должны соответствовать определенному времени, быть актуальными;
2. Полной, достаточно подробной для предполагаемого создания ГИС или картографического произведения; категории данных и их подразделения должны включать все необходимые сведения для осуществления анализа или математико-картографического моделирования исследуемого объекта или явления;
3. Позиционно-точной, абсолютно совместимой с другими данными, которые могут добавляться в нее;
4. Достоверной, правильно отражающей характер явлений, для этого необходимо четко определить включенные в нее атрибуты явлений;
5. Легко обновляемой;
6. Доступной для любых пользователей.

В процессе проектирования БД обычно выделяют три основных уровня: концептуальный, логический и физический.

Концептуальный уровень не зависит от имеющихся аппаратных и программных средств. Для БД ГИС он связан с концептуальной моделью географических данных и включает:

1. описание и определение рассматриваемых объектов;
2. установление способа представления географических объектов в базе данных;
3. выбор базовых типов пространственных объектов – точки, линии, ареалы, ячейки растра;
4. решение вопроса о способе представления размерности и взаимосвязей реального мира в БД (например, следует ли показать здание в виде ареала или точки).

На концептуальном уровне определяется и содержание базы данных, в свою очередь определяемое сутью явления, характером его пространственного распространения и задачами, для которых создается БД. Здесь следует выделить задачи создания одной или серии карт, комплексного картографирования, создания синтетических карт для многоцелевого и многократного использования.

Логический уровень определяется имеющимися программными средствами и практически не зависит от технического обеспечения. Он включает разработку логической структуры элементов базы данных в соответствии с системой управления базами данных (СУБД), используемой в программном обеспечении.

Физический уровень связан с аппаратными и программными средствами. На этом уровне определяются объемы хранимой в БД информации и необходимые объемы памяти компьютера (оперативной и долговременной), рассматриваются вопросы о структурировании файлов на диске или других носителях информации для обеспечения программного доступа к ним, представления данных в памяти компьютера (целые, действительные числа, байты или буквенно-цифровые характеристики).

Пространственные данные, как упоминалось ранее, традиционно подразделяются на две взаимосвязанные составляющие – позиционные и непозиционные.

Позиционная составляющая характеризует положение географических объектов (или пространственную форму) в координатах двух- и трехмерного пространства – декартовых*(х,у, z)*или географических (срА).

Непозиционная составляющая данных включает качественную характеристику пространственных объектов (семантику) и статистику; эта информация называется атрибутивной и представляется в виде текстовых или числовых параметров. Она соответствует тематической форме данных или кодированному представлению взаимосвязей объектов (топологии). Почти всегда тип объекта маркируется и опознается по его атрибутивным параметрам (дорога имеет название и идентифицируется по ее классу – грунтовая, с асфальтобетонным покрытием). Обычно атрибутивная информация не имеет пространственного характера, хотя некоторая ее часть может быть связана с пространственной природой изучаемого объекта, например, площадь, периметр.

В качестве атрибутивной информации часто выступает время (временная форма), которое может отражаться несколькими способами: указанием временного периода существования объектов, соотнесением информации с определенными моментами времени, указанием скорости движения объектов.

Количественные атрибуты создаются в соответствии с номинальными, порядковыми, интервальными или пропорциональными шкалами измерений. Важно знать, какие шкалы измерений использованы для данных, поскольку это определяет характер возможных математических операций с ними.

/\****Представление точечных, линейных и площадных объектов в базе данных и на цифровой карте.***В БД ГИС картографические источники и итоговые карты представляются в виде *цифровых карт*(см. 2.1.3)

**[Геоинформатика, 1999].**

Любая БД состоит из цифровых представлений дискретных объектов. Содержание карты можно хранить в БД в виде цифровой карты, превратив объекты карты в объекты базы данных. Правда, всегда нужно помнить о том, что многое из показанного на картах умозрительно не представлено в реальном мире: горизонтали в природе не существуют, а вот дома и озера – это реальные объекты.

Итак, географические объекты, моделируемые с помощью карты или ГИС, имеют три формы представления:

- объект в действительности;

- объект, представленный в базе данных (некоторые авторы вводят для таких объектов наименование «предмет»);

- знак, который используется для показа объекта (предмета) на карте или на другом графическом изображении.

(Мы будем во всех случаях использовать наименование «объект», поскольку о чем идет речь, обычно понятно из контекста.

Предназначенный для отражения в БД или цифровой карте объект – это явление действительности, последнее в ряду подразделения однотипных явлений при выборе «элементарных кирпичиков» для информационного моделирования; например, город можно считать объектом, при его подразделении на составные части они уже будут не городами, а районами, кварталами и т.п.

*Объект в БД –*это цифровое представление всего реального объекта или его части. Способ цифрового представления объекта зависит от назначения ГИС, масштаба исследования, его задач и других факторов, например, географически город может быть представлен в виде точки, если рассматриваемая территория имеет масштабы материка; если речь идет о базе географических данных области, тот же город может быть представлен ареалом.)

Сходные явления, информация о которых хранится в базе данных, определяются как типы объектов – любая группа сходных явлений, которые должны иметь одинаковую форму хранения и представления, например дороги, реки, высоты, растительность; тем самым обеспечивается основа для формирования общего атрибута явлений. Каждый тип объектов должен быть точно определен, что помогает выявить перекрывающиеся категории данных и вносит ясность в содержание базы данных.\*/

Для цифрового представления типов реальных объектов необходимо выбрать подходящую форму объектов, являющихся представителями первых (кодами) в базе пространственных данных. Их классификация может быть основана на представлении пространственной размерности.

Такие объекты хорошо отражают тип пространственной локализации реальных объектов. Они могут быть объединены в классы, например множество точек для представления множества городов.

Пространственные типы объектов БД могут группироваться в слои, именуемые также покрытиями или темами. Один слой представляет один тип объектов или группу концептуально взаимосвязанных типов объектов. Например, слой может включать только отрезки водотоков, или же водотоки, озера, береговую линию и болота. Возможны самые разные варианты системы слоев, как и модели данных. Некоторые базы пространственных данных создаются путем объединения всех объектов в один слой.

Одни и те же географические явления можно представить в разных масштабах и с разной точностью. Переход от одного представления к другому достаточно сложен, например переход от мелкого масштаба (1: 250 000) к крупному (1 : 10 000). Поэтому часто встречаются базы данных, содержащие множественные представления одних и тех же явлений. Это неэкономно, но избежать этого пока не удается, ибо соответствующие методы перехода еще недостаточно разработаны.)

Как правило, ГИС создаются на основе уже существующих систем управления базами Данных (СУБД), приобретение или аренда СУБД составляет основную часть затрат на программное обеспечение системы. СУБД выполняет множество функций, которые в противном случае следовало бы программировать в ГИС. Различают два пути использования СУБД в ГИС:

1. Выполнение ГИС-процедур полностью через СУБД, тогда доступ ко всем данным осуществляется только через СУБД и все данные должны удовлетворять требованиям, заложенным при ее разработке;
2. Некоторые данные (обычно таблицы атрибутов и их отношений) доступны через СУБД, поскольку они вполне соответствуют модели, а к некоторым данным (обычно пространственно-локализованным) доступ прямой, так как они не удовлетворяют требованиям модели СУБД.

/\*(ГИС добавляет географический аспект к уже существующим методам поиска и запроса. Сложность и разнообразие представления данных в ГИС, различимость в представлении позиционной и атрибутивной составляющей информации, необходимость ее обработки в контексте пространственной близости предъявляют своеобразные и повышенные требования к СУБД по сравнению с традиционной формой их использования.)\*/

СУБД Oracle

СУБД Oracle версии 1 (под другим названием) появилась в 1978 г. И была написана на языке ассемблера для PDP-11 под OC RSX. Oracle версии 2 появилась в 1979 г. и стала первой коммерческой реляционной СУБД с использованием SQL, а в версии 3 ее ядро было переписано на языке С. С версии 8 фирма Oracle называет свою СУБД «Объектно-реляционной». Фактически же это была и остается SQL-ориентированная СУБД, использующая собственный диалект SQL.

Диалект SQL фирмы Oracle естественно оценивать с трех сторон, определяющих одновременно и факторы влияния на существующий его вид:

1. Реляционной теорией
2. Стандартного SQL,
3. Фирмы разработчика

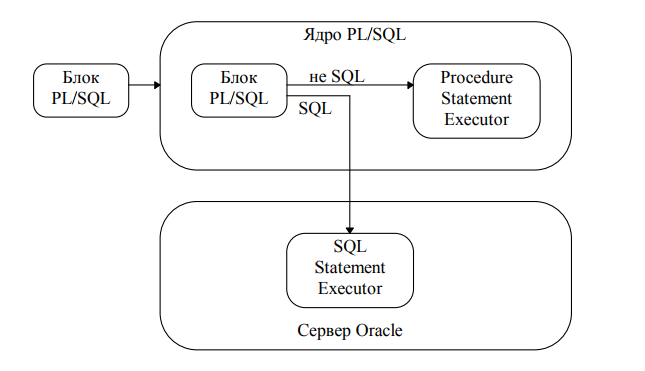
Некоторые собственные свойства диалекта SQL Oracle

1. Типы данных
2. Дополнительные к стандарту функции
3. Дополнительные конструкции SQL

PL/SQL есть процедурный язык, встроенный в СУБД Oracle для возможности работать с хранимой в БД Oracle программной логикой.

Процедурный язык Oracle позволяет создавать в БД триггерные процедуры, моделировать логику предметной области, проявлять самостоятельность БД, организовывать более сложную защиту доступа

Основной программно-лексической единицей языка является логический блок, который может представлять собой процедуру, функцию или анонимный блок. Схема обработки блоков PL/SQL показана на рисунке 2.1.



Ядро PL/SQL является одной из частей сервера Oracle и осуществляет передачу кода PL/SQL компоненту Procedure Statement Executor (исполнитель процедурных команд). Команды SQL, содержащиеся в блоке, передаются для обработки компоненту SQL Statement Executor (исполнитель команд SQL).

Каждый блок PL/SQL может включать до трех секций: описательную или декларативную (эта секция необязательна), секцию исполняемого кода (обязательна) и секцию обработки исключений (необязательна). Структура блока PL/SQL:

DECLARE - необязательно

- переменные, константы, курсоры

BEGIN - обязательно –

команды SQL, управляющие команды PL/SQL

EXCEPTION - необязательно –

действия в случае возникновения ошибок

END; - обязательно

Блоки PL/SQL состоят из строк, каждая из которых может содержать следующие лексические элементы:

1. разделители (простые и составные);
2. идентификаторы (включая зарезервированные слова);
3. литералы;
4. комментарии.

Для выполнения операций над пространственными данными Oracle предоставляет отдельно лицензируемый компонент Oracle Spatial.

Oracle Spatial — это компонент базы данных, состоящий из типов данных, набора функций и процедур, а также внешних по отношению к базе данных веб-сервисов, которые позволяют эффективно хранить, быстро получать доступ и анализировать пространственные данные.

**Преимущества хранения данных в Oracle Spatial:**

1. **Индустриальный формат**. Oracle Spatial поддерживает стандартный формат хранения всеми крупными ГИС-вендорами и поставщиками пространственных данных, т.е. не требуется конвертировать данные в разные форматы
2. **Скорость работы**. Обработка данных осуществляется в месте их хранения, что очень эффективно и масштабируемо. База данных знает природу данных и "помогает" приложению обрабатывать данных.
3. **Централизованное хранение**. От этого выигрывает безопасность и надежность хранения.
4. **Преимущества наследуемые от СУБД Oracle**. СУБД Oracle — это самая мощная СУБД и в нее встроены мощные возможности по обработке данных: отказоустойчивость, шифрование, репликация, транзакционность, независимость от платформы и т.д. Используя Oracle Spatial, мы дополняем ГИС-системы этими преимуществами.

Для хранения пространственных данных Oracle использует объектно-реляционную модель данных. За счет этого хранить и обрабатывать пространственные данные довольно просто.

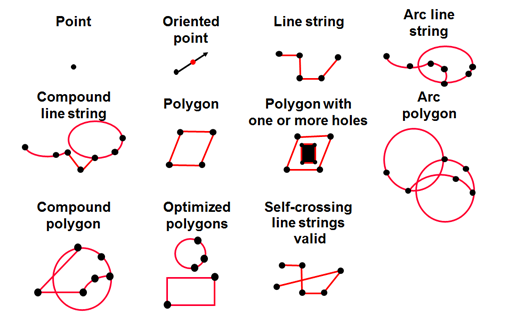
Первый тип пространственных данных, который может хранить в Oracle Spatial — это векторные данные. Для векторных данных характерно наличие точных координат вершин объекта. К таким данным можно отнести:

1. точки(скважины, остановки общественного транспорта, банкоматы);
2. полилинии(нефтепроводы, маршруты движения транспорта, улицы);
3. полигоны (нефтеносный участок, зона обслуживания банковского отделения, микрорайон).

Есть также более сложные (иногда многомерные) типы объектов:

1. мультиточка(станции метро с несколькими выходами),
2. мультиполигон (Россия — это большая территория + Калининград и острова),
3. мультилиния(нефтепроводы, имеющие несколько ниток),
4. полигон с дыркой (Московская область без территории Москвы),
5. поверхности (стена дома),
6. твердые тела (дом).

Приведенные объекты могут быть как двумерные, так и трехмерными.



Для хранения векторных объектов используется тип данных **SDO\_GEOMETRY**. Он является сложным (complex) объектным типом данных. Его структура приведена ниже:

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| SDO\_GTYPE | NUMBER | Описывает тип геометрии(точка, линия, полигон и т.д.) и кол-во измерений |
| SDO\_SRID | NUMBER | Используется для идентификации системы координат (пространственной системы отсчета), которая должна быть связана с геометрией |
| SDO\_POINT | SDO\_POINT\_TYPE | Если пространственный объект это точка, то хранит координаты точки. SDO\_POINT\_TYPE – имеет атрибуты x, y, z типа NUMBER |
| SDO\_ELEM\_INFO | SDO\_ELEM\_INFO\_ARRAY | Описывает то, как интерпретировать координаты в SDO\_ORDINATES |
| SDO\_ORDINATES | SDO\_ORDINATE\_ARRAY | Содержит координаты объекта в соответствие со способ описания, заданном в предыдущем поле |

.NET.

.NET – это стратегия создания крупных распределенных программных систем, разработанная Microsoft. Ключевым элементом .Net является платформа .NET Framework, т.е. компонентная модель программного обеспечения для работы в Internet. Компонентная модель позволяет совместно использовать отдельные программные компоненты, созданные на разных языках программирования, в виде единой функционирующей системы.

На рис. 1.1 показан чрезвычайно упрошенный вид архитектуры и взаимосвязей между элементами платформы .NET Framework. Среда CLR является основной, на которой располагаются все остальные компоненты .NET Framework. Над средой CLR располагается базовая платформа Base Framework и набор библиотек классов, которые могут совместно использоваться любыми языками совместимыми со средой .NET

Приложения

Web-формы

Web-службы

Клиент Windows

Базовая платформа Base Framework

Общеязыковая исполняющая среда (CLR)

Система выполнения

Система метаданных

Система типов

Рис. 1.1 Архитектура платформы .NET Framework

Среда Common Language Runtime является ключевым компонентом .NET Framework и состоит из трех основных элементов

1. Система типов, которая поддерживает многие типы и операции, имеющиеся в современных языках программирования.
2. Система метаданных, которая позволяет сохранять метаданные вместе с типами во время компиляции и запрашивать их с помощью других компиляторов CLR или системы выполнения во время выполнения программ.
3. Система выполнения запускает CLR программы, используя метаданные для предоставления таких сервисов, как, например, управление памятью.

На одном уровне среда CLR может рассматриваться как объединение компонентов многих языков и архитектур программирования. Она определяет стандартные типы, позволяет разработчикам определять их собственные типы, а также указывать интерфейсы и методы для их типов. Среда CLR позволяет компиляторам сохранять метаданные вместе с пользовательскими типами, предоставляя возможность другим языкам проверять, использовать и расширять эти типы

Базовая платформа Base Framework предлагает несколько фундаментальных классов таких как:

1. Класс Object является базовым для всех классов. Он предлагает несколько методов, включая те, которые разработчики используют для доступа к метаданным практически любого типа
2. Класс String представляет собой Unicode-строку, которая может совместно использоваться разными языками программирования и с разными региональными стандартами
3. Класс Type является фундаментальным строительным блоком, который позволяет выполняемым программам получать доступ к системе метаданных

Клиенты Windows. Пространство имен System.Windows.Forms платформы .NET Framework содержит типы для создания приложений с графическим интерфейсом пользователя (Graphic User Interfaces – GUI) для операционной системы Windows. Такие GUI-библиотеки активно используются в эффективных средствах быстрой разработки приложений (Rapid Application Development – RAD). Основным преимуществом этих библиотек является то, что они предлагают спецификацию и принимаемую по умолчанию реализацию GUI-приложений, а также требуют от разработчиков только переопределить поведение некоторых типов, если требования приложений отличаются предлагаемых функциональных возможностей.

Web-Формы ASP.NET. Технология ASP.NET предлагает полный набор типов для создания Web-ориентированных приложений. В ней определены типы, которые представляют все элементы полноценной Web-ориентированной системы: от типов, представляющих визуальные элементы Web-приложения, до типов, предлагающих такие функции Web-узла, как кеширование и обеспечение безопасности. Технология ASP.NET предлагает функции динамической компиляции Web-страниц, способность использовать многие другие .NET–совместимые языки, а также способность повторно использовать типы среды .NET в Web-страницах.

Web-службы ASP.NET являются новым стандартом предоставления доступа к программным функциям в Internet. Эти службы построены на основе таких открытых стандартов и протоколов, как HTTP, XML, SOAP, позволяющих компонентам взаимодействовать независимо от операционной системы на том компьютере, на котором они находятся. Платформа .NET Framework предлагает типы и службы для поддержки процесса, создания, развертывания и использования Web-служб.

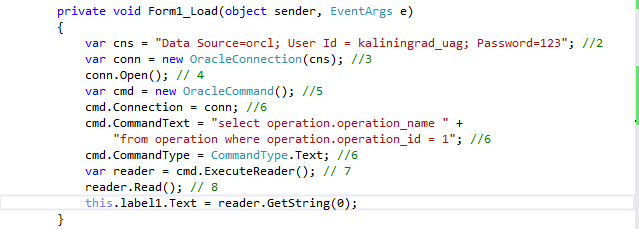
Приложения и платформа .NET Framework. В верхней части схемы платформы .NET Framework на рис 1.1 находится раздел приложений, который охватывает настольные приложения, Web-ориентированные приложения ASP.NET, а также Web-службы. Эти приложения используют функции предлагаемые на более низких уровнях этой диаграммы. Все они базируются на платформе .NET Framework и обладают следующими основными достоинствами.

1. Для всех приложений используются согласованные концепции и службы. Эта согласованность значительно сокращает процесс освоения новых компонентов, которые могут самыми разными способами предлагаться для открытого использования.
2. Расширяются возможности повторного использования компонентов.
3. Поддержка нескольких языков программирования означает, что выбор любого языка не связывает разработчика только с какой-то одной библиотекой, специфичной для данного языка.

Oracle Data Provider (ODP.NET).

В приложении, использующим платформу .NET Framework, использовался поставщик данных Oracle или ODP.NET. ODP.NET – это высокопроизводительный поставщик данных oracle, совместимый с ADO.NET 2.0, который предоставляет полный набор специальных возможностей и параметров настройки Oracle, также включая поддержку Real Application Clusters, XML DB и расширенную безопасность.( The Oracle Data Provider for .NET (ODP.NET) is Oracle's high performance ADO.NET 2.0 compliant data provider that exposes a complete set of Oracle specific features and tuning options including support for Real Application Clusters, XML DB, and advanced security. It is available for free download from the Oracle Technology Network website.)

Рассмотрим небольшой пример использования ODP.NET на языке C#.



1. В начале необходимо подключить библиотеку Oracle.ManagedDataAccess.Client
2. Создаем строку подключения cns, которая состоит из службы, имени пользователя и пароля.
3. Создаем объект класса OracleConnection, который представляет соединение с базой данных Oracle, передавая в параметры конструктора строку соединения (cns).
4. После этого открываем соединение с базой данных.
5. Для выполнение sql-запроса или хранимой процедуры создаем объект класса OracleCommand. Объект OracleCommand отвечает за формулировку запроса и передачу его в базу данных (The OracleCommand object is responsible for formulating the request and passing it to the database.)
6. Задаем параметры объекта OracleCommand такие как:
   1. Соединение(Сonnection)
   2. Тип команды(CommandType), который указывает как интерпретировать свойство CommandText
   3. Текст команды(CommandText)
7. Выполняем команду с помощью метода ExecuteReader(), который возвращает объект OracleDataReader представляющий собой набор результатов.
8. Получаем следующую строку из набора результата с помощью метода Read().