Базы данных. Основные понятия и общие представления

Цели лекции

- изучение концепций построения и технологий приложений баз данных различного предназначения;
- практическое овладение технологиями защищенных баз данных

Применение баз данных

- основная цель создания приложений баз данных - хранение, накопление, обработка и представление информации различного рода;
- области и масштабы применения баз данных могут быть очень обширными – от небольших приложений для настольных компьютеров до крупных хранилищ в центрах обработки данных (ЦОД), доступ к которым осуществляется в том числе с использованием интернет-технологий.

Примеры приложений БД

- цель: учет информации (текстовая и числовая, аудио, видео и т.п.):
  - адресная/телефонная книга;
  - учет клиентов и заказов фирмы;
  - каталог продукции;
  - электронный каталог библиотеки;
  - «Электронный Университет»;
  - геоинформационные системы (ГИС);
- очевидно, что требования к приложениям могут зависеть как от предметной области, так и от масштабов системы.

Некоторые требования к приложениям БД

- производительность;
- характер накапливаемой информации (числовые и текстовые данные, аудио и видео, изображения и т.п.)
- масштабируемость:
  - объем накапливаемой информации;
  - многопользовательский режим и количество пользователей;
- возможность сетевого доступа к данным;
- безопасность;
- низкая сложность администрирования и поддержки;
- низкие накладные расходы при модификации базы данных и приложения (рефакторинге);
- переносимость;
- надежность и отказоустойчивость;
- требования к квалификации пользователей;
- наличие дополнительных возможностей (загрузка и обработка данных, построение форм и отчетов);
- …

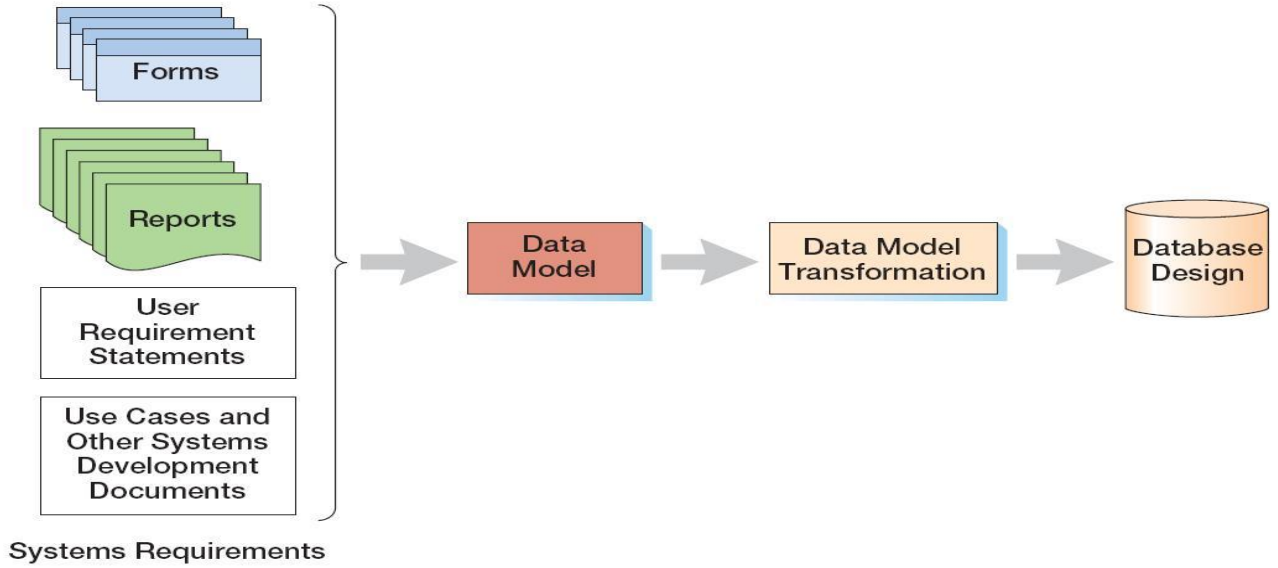
Процесс разработки БД

- общие стратегии:
  - разработка сверху вниз (*top-down database development*)
    - получение абстрактной модели данных исходя из анализа стратегических целей организации, способов их достижения, а также информации и способов ее представления, которые необходимы для достижения этих целей;
    - лучшее взаимодействие подсистем, глобальный охват;
  - разработка снизу вверх (*bottom-up database development*)
    - для разработки выбирается конкретная система, которая выполняет только часть функций предприятия;
    - исходя из сформированных требований, выбранная подсистема создается быстрее и с меньшими рисками;
- одна из основ эффективной реализации приложения базы данных – ясное представление модели предметной области.

Моделирование данных

- база данных является «моделью модели», то есть через объекты базы данных описывает представление пользователей о конкретной предметной области;
- при любом процессе разработки необходимо сформулировать требования и построить на их основе модель данных, что является во многом творческой задачей, решение которой основано на опыте и интуиции;
- моделирование данных (*data modeling*) - процесс создания логического представления базы данных (пользовательская модель данных /*user data model*/, модель требований к данным /*requirements data model*/, концептуальная модель /conceptual data model/);
- модель данных содержит языковые и изобразительные стандарты для своего представления:
  - модель сущность – связь (entity-relationship model);
  - семантическая объектная модель (semantic object model);

Три уровня моделей информационных систем

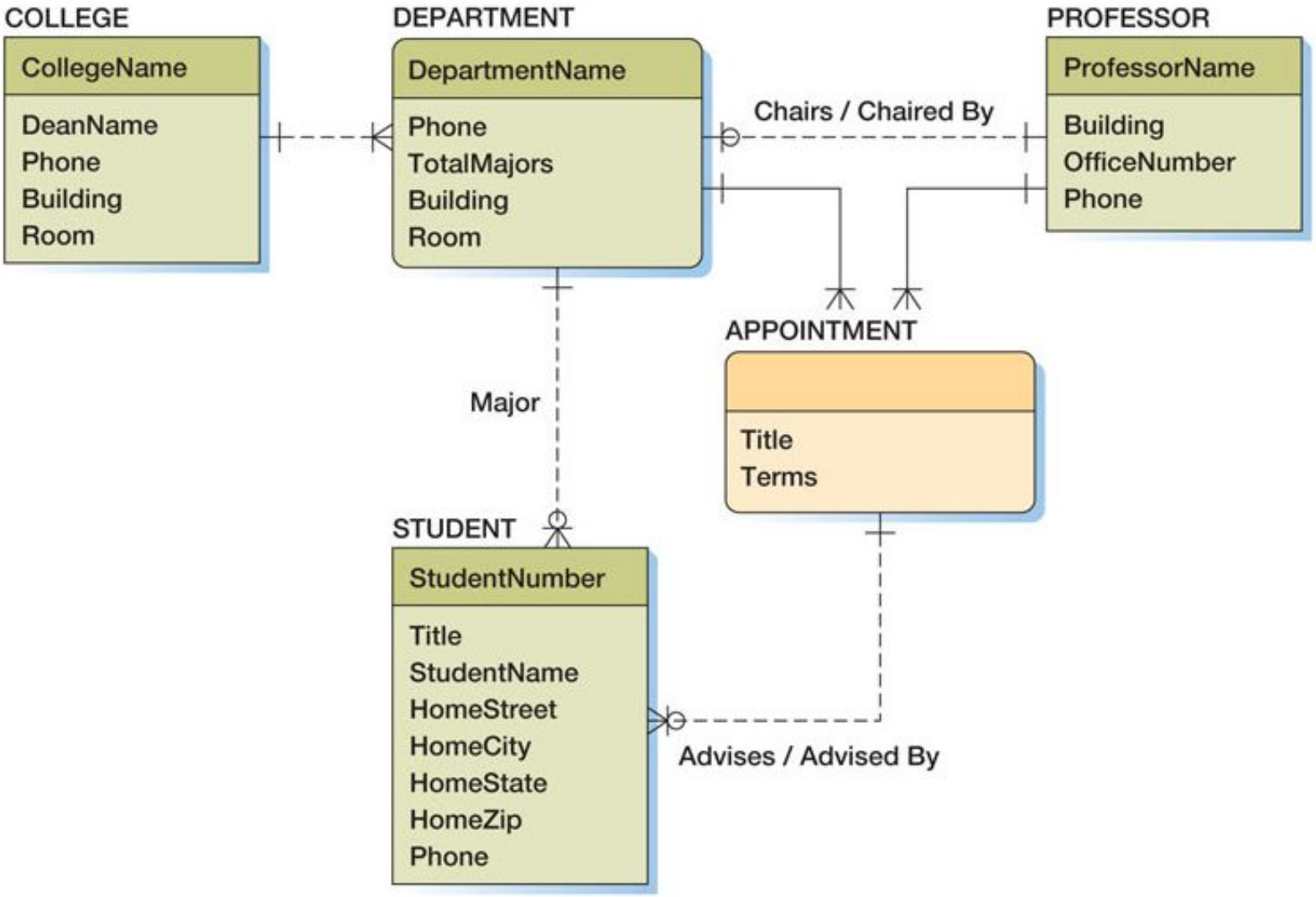


- внешние модели (*external schema*): представление пользователей о системе (user view);
- концептуальная модель (*conceptual schema*): абстрактное представление системы (данных);
- внутренние модели (internal schema).

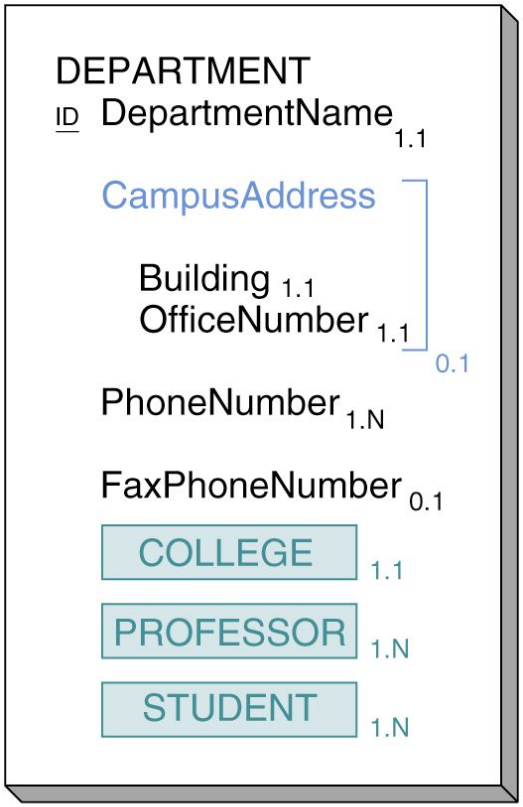
Модели данных (ANSI)

- внешняя модель (external schema) – набор представлений пользователей о той части предметной области, с которой он сталкивается;
- концептуальная модель (conceptual schema / conceptual data model) – набор ключевых объектов предметной области, их взаимосвязей (взаимодействий) и ограничений, накладываемых на объекты;
- логическая модель (logical schema / logical data model / information schema) – представление концептуальной модели в соответствии с логической моделью, используемой для хранения данных;
- физическая модель (physical data model) – описание физического представления, хранения и обработки данных.

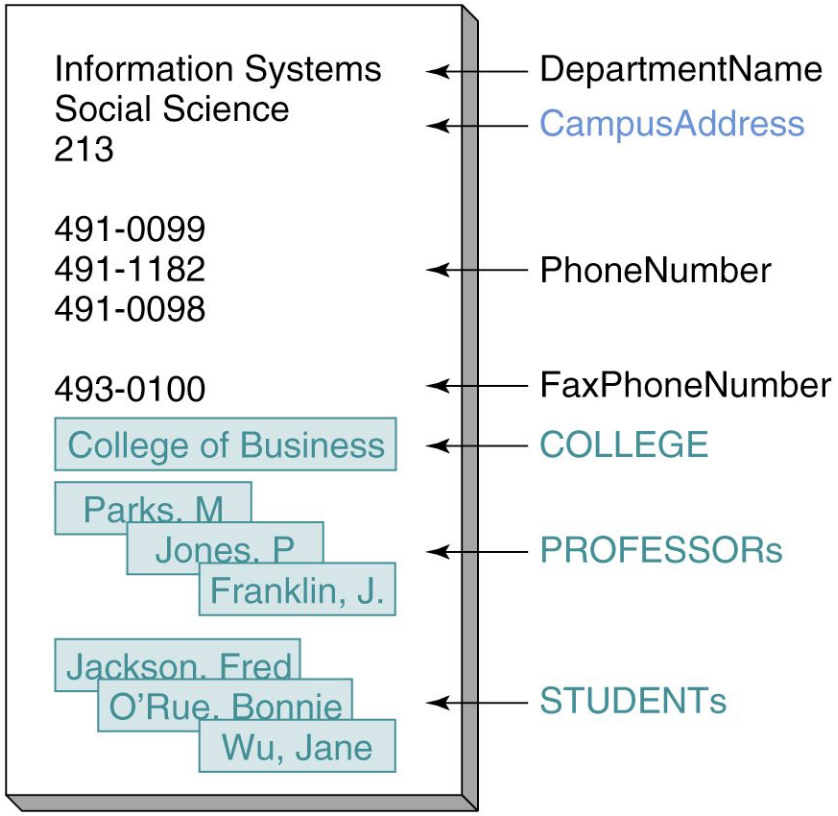
Модель сущность-связь



Семантическая объектная модель



(b)



История развития баз данных

- крупные БД (системы обработки транзакций масштаба крупных организаций) настольные БД сетевые БД интернет:
  - до 1970-х – переход от ведения записей вручную к системам обработки файлов (file-processing systems);
  - 1970-1980 – появление первых баз данных (dBase, ADABAS, Total, System2000, IDMS, IMS);
  - 1978-1985 – появление реляционных баз данных (DB2, Oracle);
  - 1982-1992 – развитие баз данных для настольных компьютеров (dBase-II, Paradox, Access, dBase-IV, FoxPro);
  - 1985-2000 – появление и развитие объектно-ориентированных баз данных;
  - 1995-н.в. – интеграция технологий баз данных и интернет-технологий (в т.ч. NoSQL);

Системы обработки файлов

- разделенные и изолированные файлы, каждому приложению может соответствовать свой набор файлов;
- зависимость прикладных программ от форматов файлов (в том числе, даже если некоторые поля не используются) и несовместимость файлов;
- дублирование данных (data duplication) – возможное нарушение целостности данных (data integrity);
- трудность представления данных в файлах в различных видах / совместного использования данных (в том числе из-за отсутствия явно определенной связи между данными);
- сложность разработки и поддержки.

База данных

- база данных – самодокументированный набор интегрированных записей;
- самодокументированность: база данных содержит не только данные, но и их описание – метаданные (metadata):
  - возможность определить структуру и содержимое базы данных путем обращения к самой базе данных;
  - изменения структуры в первую очередь затрагивает метаданные, и, следовательно, ограничивает количество программ, на которые влияют эти изменения;
- интегрированность:
  - байты поля записи файлы;
  - метаданные;
  - индексы (описывают связи между данными и увеличивают производительность);
  - метаданные приложения (дополнительные данные для построения форм и отчетов);

Реляционная модель

- реляционная модель (relational database model)
  - введена в 1970г. Э.Ф.Коддом (E.F.Codd, A Relational Model of Data for Large Shared Databanks);
  - основана на применении концепции реляционной алгебры к проблеме хранения больших объемов данных;
  - определяет способ хранения данных в виде таблиц со строками и столбцами, при этом минимизируется дублирование и исключаются определенные типы ошибок обработки;
  - дает стандартный способ структурирования и обработки данных;
- нормализация (*normalization*) – последовательность преобразований, обеспечивающих определенный набор требований.

EmpNum	EmpName	DeptNum	DeptName
100	Jones	10	Accounting
150	Lau	20	Marketing
200	McCauley	10	Accounting
300	Griffin	10	Accounting

(a) One-Table Design

OR?

DeptNum	DeptName
10	Accounting
20	Marketing

EmpNum	EmpName	DeptNum
100	Jones	10
150	Lau	20
200	McCauley	10
300	Griffin	10

(b) Two-Table Design

Объекты реляционной базы данных

- основной объект – отношение (relation), представляющее собой двумерную таблицу:
  - столбцы (поля, атрибуты) – определяют набор данных, которыми обладает описываемый объект предметной области;
  - строки (записи) – содержат набор значений атрибутов для конкретного объекта;
- домен (domain) – множество допустимых значений атрибута:
  - физическое описание: тип данных и дополнительные наложенные ограничения;
  - семантическое (логическое) описание: описывает назначение данного атрибута;
- ключ (key) – набор из одного или нескольких атрибутов, однозначно идентифицирующий конкретную запись.

Реляционная БД: пример

- структура таблиц (отношений), выбор ключей и процедура нормализации напрямую зависит от модели данных, которая была сформирована на основе требований пользователей к системе

STUDENT Relation

SID	Name
100	Jones
200	Chau
300	Garrett
400	Jones

CLUB Relation

Club	Cost
Climbing	150
Scuba	400
Skiing	550

PAYMENT Relation

SID	Club	AmtPaid
100	Scuba	0
100	Skiing	550
200	Scuba	400
300	Climbing	150
400	Skiing	550

Язык SQL

- язык SQL (*Structured Query Language* - «язык структурированных запросов») – язык манипулирования реляционными данными;
- язык SQL представляет собой совокупность:
  - операторов;
  - инструкций;
  - вычисляемых функций;
- операторы языка SQL делятся на:
  - операторы определения данных (*Data Definition Language, DDL*) - CREATE, ALTER, DROP;
  - операторы манипуляции данными (*Data Manipulation Language, DML*) - SELECT, INSERT, UPDATE, DELETE;
  - операторы определения доступа к данным (*Data Control Language, DCL*) – GRANT, REVOKE, DENY;
  - операторы управления транзакциями (*Transaction Control Language, TCL*) – COMMIT, ROLLBACK.

Метаданные в реляционной базе данных

- хранение метаданных осуществляется в системных таблицах (system tables), которые могут располагаться в специализированной системной базе данных

USER_TABLES Table

TableName	NumberColumns	PrimaryKey
STUDENT	3	StudentNumber
CLASS	4	ClassNumber
GRADE	3	(StudentNumber, ClassNumber)

USER_COLUMNS Table

ColumnName	TableName	DataType	Length (bytes)
StudentNumber	STUDENT	Integer	4
StudentName	STUDENT	Text	50
EmailAddress	STUDENT	Text	50
ClassNumber	CLASS	Integer	4
Name	CLASS	Text	50
Term	CLASS	Text	5
Section	CLASS	SmallInteger	2
StudentNumber	GRADE	Integer	4
ClassNumber	GRADE	Integer	4
Grade	GRADE	Decimal	(3, 2)

Создание базы данных

- создание базы данных заключается в определении схемы базы данных (database schema), т.е. набора объектов, которые отражают разработанную модель данных предметной области:
  - таблиц;
  - связей;
  - доменов;
  - объектов, реализующих бизнес-логику;
- бизнес-логика - набор ограничений на возможные действия пользователя (с данными, хранящимися в БД);
- бизнес-логика может быть реализована:
  - на уровне СУБД: независимо от источника действий, ограничения выполняются в любом случае;
    - хранимые процедуры и функции;
    - триггеры – процедуры, выполняющиеся при наступлении определенных событий в базе данных;
  - на уровне приложения (создание форм и отчетов): ограничения выполняются только в рамках конкретного приложения;

Категории целостности данных

- целостность данных (data integrity) подразделяется на следующие категории:
  - *сущностная целостность*: определяет строку как уникальную сущность в конкретной таблице, поддерживается через введение ключа (если исходя из модели данных выделить ключ невозможно, то вводится суррогатный ключ – *surrogate key*);
  - *доменная целостность*: определяет достоверность записей в конкретном столбце;
  - *ссылочная целостность*: сохраняет определенные связи между таблицами при вводе или удалении записей, поддерживается внешними ключами, т.е. наличием атрибута, который может принимать только значения первичного ключа другой таблицы;
  - *пользовательская целостность*: позволяет определять бизнес-правила, не входящие ни в одну из категорий целостности.

Представления (Views)

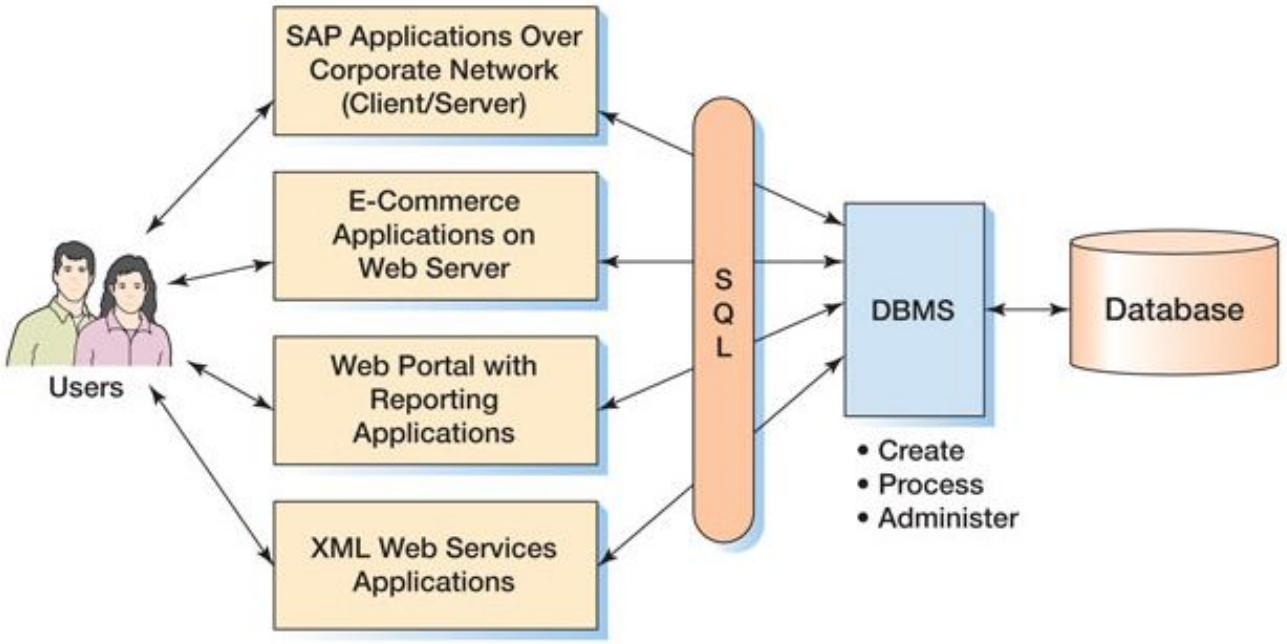
- представление - это виртуальная таблица, формируемая из совокупности именованных столбцов и строк данных, содержимое которой определяется запросом на основе данных других таблиц и представлений;
- функции представлений:
  - упрощение и настройка восприятия каждым пользователем информации базы данных;
  - реализация механизмов безопасности, обеспечивающих возможность обращения пользователей к данным без предоставления им разрешений на непосредственный доступ к базовым таблицам, лежащим в основе представлений;
  - обеспечение интерфейса обратной совместимости, моделирующего таблицу, которая существует, но схема которой изменилась;

Индексы (Indexes)

- индексы базы данных представляют собой специальные сохраняемые структуры данных, которые предназначены для ускорения выполнения запросов к данным, выполняющих, например:
  - сортировку;
  - отбор записей таблицы по условию, содержащему один или несколько проиндексированных атрибутов;
- индексы требуют дополнительных накладных расходов на обновление и хранение;

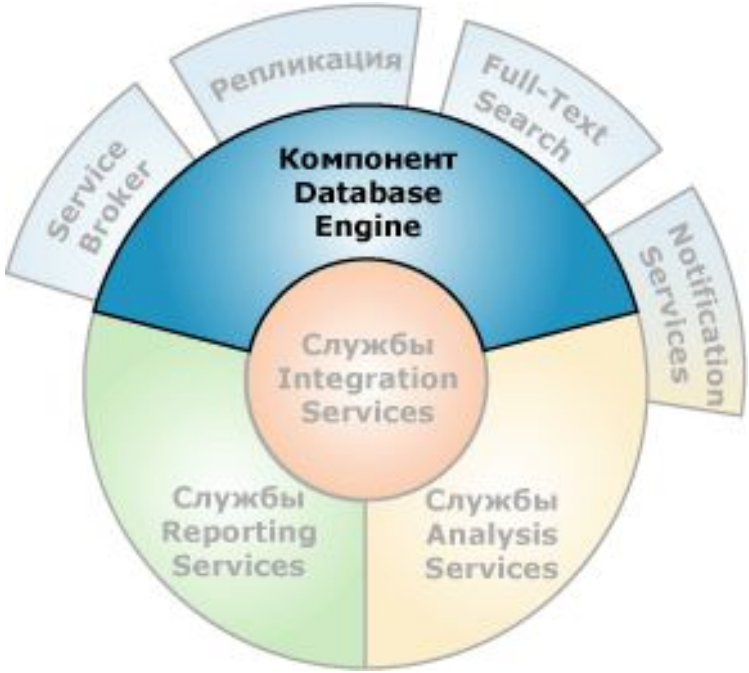
СУБД

- синонимичные термины:
  - система обработки баз данных (database processing system);
  - СУБД - система управления базами данных (DBMS, database management system);
- СУБД - является посредником между приложением и данными:
  - изоляция от физической организации хранения данных;
  - данные интегрированы;
  - уменьшение дублирования данных;
  - независимость программ от форматов файлов;
  - возможность гибкого выбора формы представления данных;



Компоненты современных СУБД

- ядро СУБД (database engine) обеспечивает хранение, обработку и защиту данных:
  - преобразование запросов в действия над данными;
  - безопасность;
  - управление транзакциями и блокировками;
  - резервное копирование и восстановление;
- средства интерактивной аналитической обработки (OLAP) и средства интеллектуального анализа данных;
- средства интеграции данных (экспорт и импорт данных);
- средства создания и публикации форм и отчетов, создания форм;
- средства репликации (копирования и распространения данных и объектов баз данных из одной базы данных в другую с возможностью последующей синхронизации между базами данных для поддержания согласованности);
- средства выполнения полнотекстовых запросов к неструктурированным символьным данным в таблицах;
- …



Управление параллельной обработкой в БД

- управление параллельной обработкой (*concurrency control*) направлено на то, чтобы исключить *непредусмотренное* влияние действий одного пользователя на действия другого;
- управление параллельной обработкой, как правило, предполагает компромисс между уровнем изоляции различных операций друг от друга и производительностью системы;
- транзакция (transaction) является последовательностью операций, выполненных как одна логическая единица работы (logical unit of work), т.е. либо все действия внутри транзакции выполняются успешно, либо не выполняется ни одно из них.

OLTP vs OLAP

- большинство приложений подразделяются на две основные категории приложений баз данных:
  - оперативная обработка транзакций (OLTP – Online Transaction Processing):
    - приложениями пользуются многие пользователи, которые одновременно совершают транзакции, изменяя таким образом текущие данные;
    - хотя отдельные запросы пользователей обращаются лишь к небольшому числу записей, при этом одновременно производится большое количество таких обращений;
    - управление параллелизмом в системе базы данных гарантирует, что два пользователя не могут одновременно изменять одни и те же данные и что пользователь не может изменить какие-либо данные, пока другой пользователь не закончит работу с ними;
    - атомарность гарантирует успешное выполнение всех шагов транзакции как группы операций;
  - поддержка принятия решений (хранилища данных, OLAP – Online Analitycal Processing)
    - предназначены для организации хранения большого количества неизменных данных с целью упрощения их анализа и получения;
- характеристики этих типов приложений оказывают существенное влияние на специальные требования к разработке базы данных.

Наиболее распространенные СУБД (реляционные)

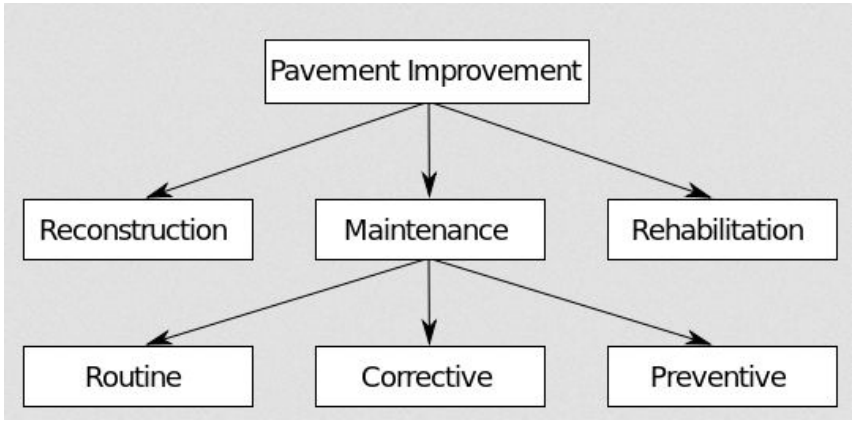
- Oracle Corporation
  - Oracle
- Microsoft
  - Access
  - FoxPro
  - SQL Server
- IBM
  - DB2
- Open Source
  - MySQL
  - SQLite
  - PostgreSQL
- Teradata

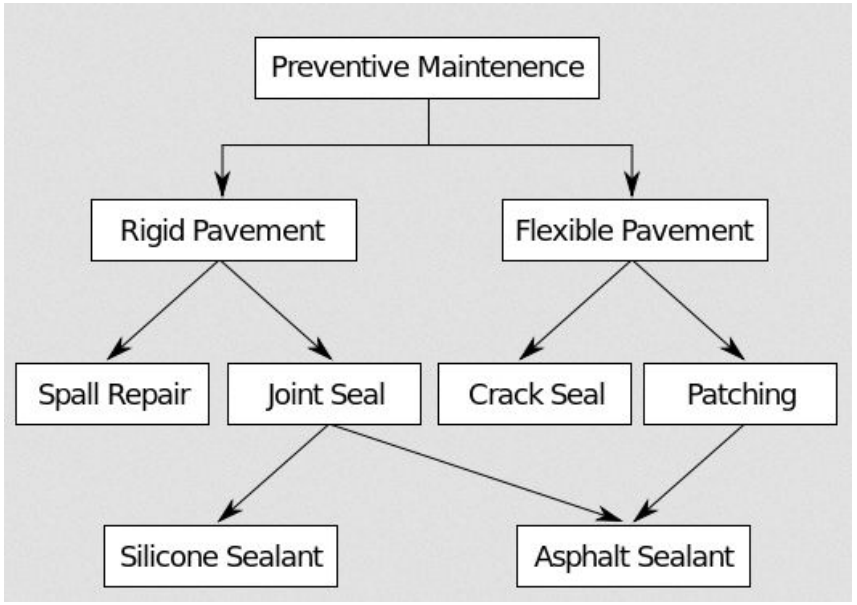
Модели баз данных

- логическая модель (logical data model) базы данных определяет:
  - набор поддерживаемых типов структур данных;
  - набор допустимых операций над поддерживаемыми структурами данных;
  - набор общих правил целостности данных, явно или неявно определяющих корректные состояния базы данных или их изменения;
- logical data model □□ information model;
- модели баз данных:
  - иерархическая (hierarchical model);
  - сетевая (network model);
  - реляционная (RDBMS, relational DBMS) – горизонтальное и вертикальное хранение (row / column store);
  - пост-реляционные модели, NoSQL (Not only SQL) базы данных:
    - объектно-ориентированные (OODBMS, object-oriented DBMS);
    - хранилища ключей и значений (KV-store, key-value store, KVP, key-value pairs);
    - документо-ориентированные (document-oriented model);
    - графовые базы данных (graph databases);
    - столбцовые базы данных;
  - многостороннее хранение (polyglot persistence).

Иерархическая и сетевая модели

- иерархическая модель (hierarchical model) – модель данных, в которой данные представляют собой древовидную структуру;
  - пример: реестр Windows (Windows Registry);
- сетевая модель (network model) – модель данных с сетевой структурой, возможно наличие циклов (как на уровне модели данных, так и на уровне самих данных).





Объектно-ориентированные базы данных

- object-oriented programming, объектно-ориентированное программирование (конец 1980х);
- объектно-ориентированные СУБД, object-oriented DBMS – предназначены для прозрачной и унифицированной обработки структур данных ООП, так как реляционная модель для этого не вполне подходит:
  - сложность переноса накопленных данных;
  - не обладают достаточной универсальностью (поддержка наследования);
  - примеры: Cache, ObjectDB;
- Oracle: object-relational databases;
- системы ORM (*Object-relational mapping*, объектно-реляционное отображение) – системы преобразования объектов (в терминах ООП) в форму, которая подходит для их хранения в файлах и базах данных:
  - примеры: NHibernate.

Хранилища ключей и значений

- KV-store, key-value store (KVP, key-value pairs);
- хранилища сопоставляют значения ключам;
- примеры: memcached (memcachedb, membase), MS Velocity, Redis, Riak, Tokyo Cabinet, MongoDB, Tarantool, Apache Cassandra, Google BigTable, Amazon Dynamo;
- дополнительные возможности:
  - поддержка различных типов значений: текст, изображения, XML-документы, составные типы данных (например, отсортированные множества)
  - поддержка веб-технологий (HTTP, REST-*representational state transfer*);
  - механизм запросов;
  - коммуникационные средства (публикация-подписка, очереди);
- могут рассматриваться более сложные способы организации данных на уровне хранения (тройки /triplestore//subject-predicate-object, entity-attribute-value model/ и т.д.).

Документо-ориентированные базы данных

- документо-ориентированные базы данных (document-oriented databases) предназначены для хранения и обработки документо-ориентированной (полуструктурированной, semi-structured) информации;
- основаны на понятии *документа*, как сущности, включающей в себя некоторый набор данных, закодированных в стандартных форматах (XML, YML, JSON, BSON, PDF, MS Office);
- документы могут содержать вложенные структуры;
- каждый из документов может содержать в себе как общие для некоторого набора документов поля, так и уникальные, что делает структуру документа более гибкой по сравнению с реляционной моделью;
- база данных предоставляет средства извлечения документов (document retrieval) на основе содержимого документов (content) в виде API или языка запросов (query language);
- примеры: CouchDB, Redis, MongoDB, LotusNotes, OrientDB, …

Графовые базы данных

- графовые базы данных (graph databases) предназначены для хранения структуры графа: узлов и связей между ними;
- как с узлами, так и со связями могут быть ассоциированы свойства (атрибуты), представляющие собой пару ключ-значение и позволяющие хранить дополнительные данные;
- язык запросов или API графовых баз данных, а также оптимизация производительности ориентированы на предоставление возможности максимально удобного и быстрого обхода узлов по связям (поиск в ширину, нахождение подграфов и т.д.);
- примеры: DEX, Neo4j, FlockDB, SonesDB …

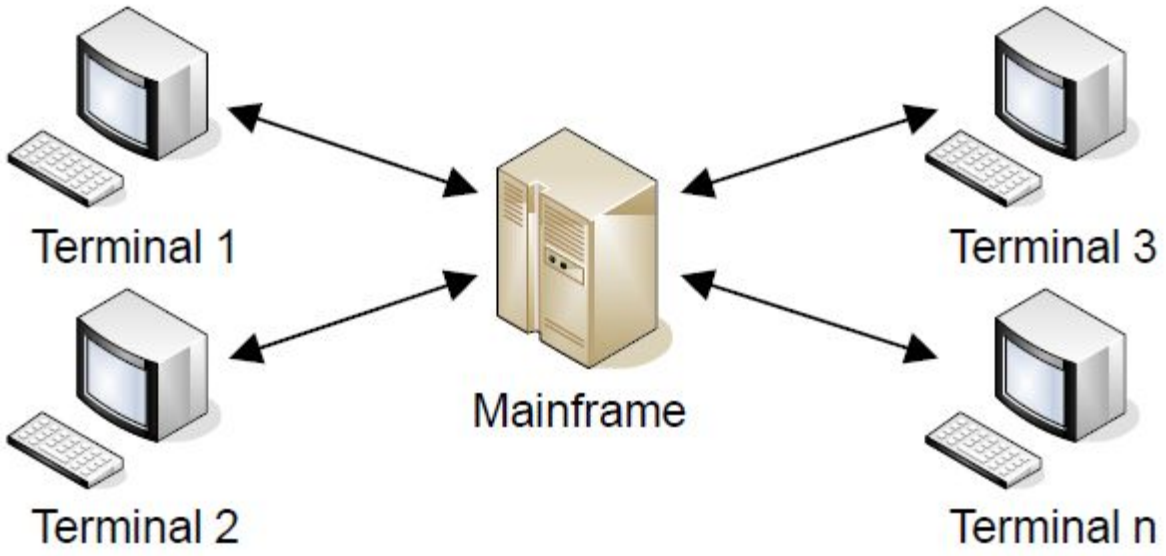
Столбцовые базы данных

- столбцовые базы данных (column-oriented databases) ориентированы на хранение данных по столбцам, в отличие от реляционных баз данных (хранение по строкам):
  - несущественные накладные расходы на добавление нового столбца к уже существующим данным;
  - гибкость схемы данных (набор столбцов у различных строк может быть разным);
  - возможна высокая степень сжатия данных для столбцов с повторяющимися значениями;
- примеры: Apache HBase, Apache Cassandra, Google BigTable …

Схемы доступа к данным

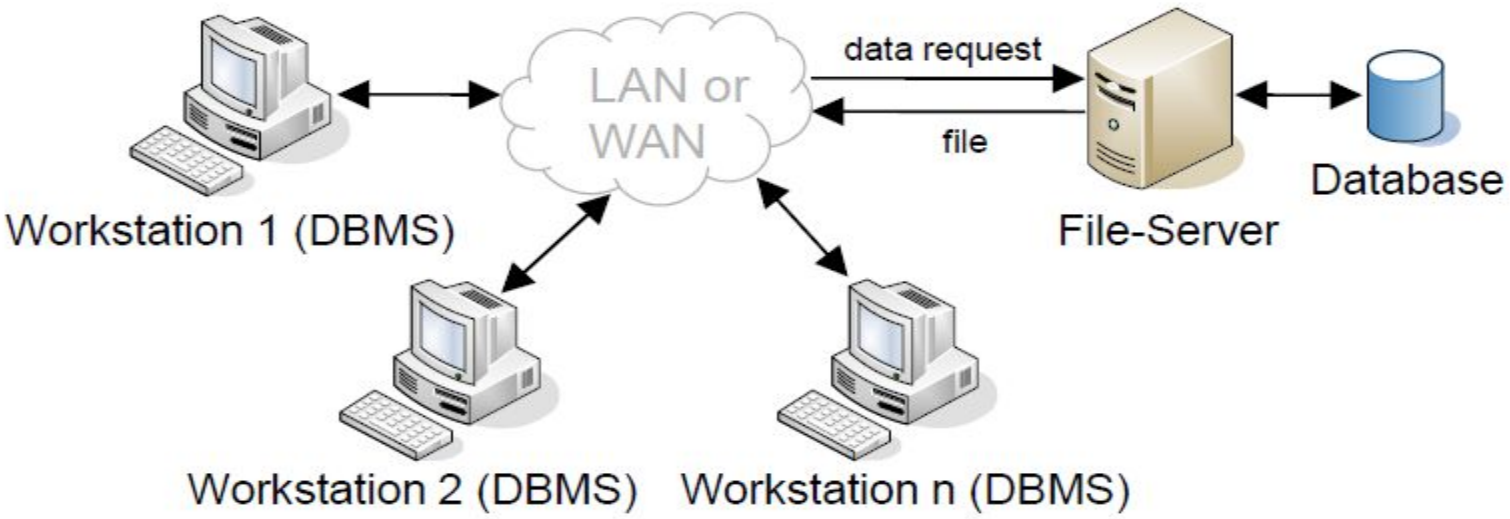
- терминальный доступ (teleprocessing);
- доступ в режиме разделения файлов (*file-sharing*);
- клиент-серверные системы (*client-server systems*):
  - двухзвенные;
  - трехзвенные;
  - многозвенные;
- архитектура «точка-точка» (peer-to-peer);
- параллельные базы данных (parallel databases);
- распределенные СУБД (distributed database systems, DDBMS);
- архитектуры, основанные на службах (SOA, service-oriented architectures);
- облачные вычисления (Cloud computing);
- мобильные и встраиваемые базы данных.

Терминальный доступ



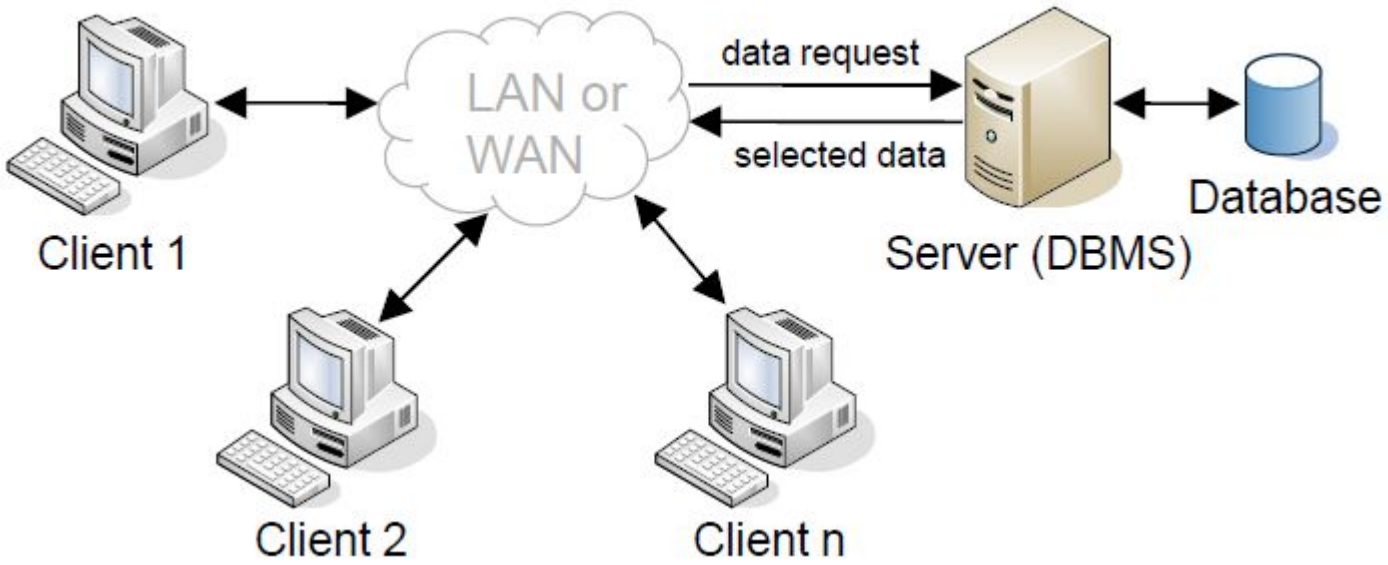
- единственный мейнфрейм и множество терминалов (предназначенных исключительно для ввода-вывода);
- недостаток: высокая нагрузка на мейнфрейм, т.к. на нем:
  - выполняются приложения и СУБД;
  - производится подготовка информации для отображения на терминалах.

Доступ в режиме разделения файлов



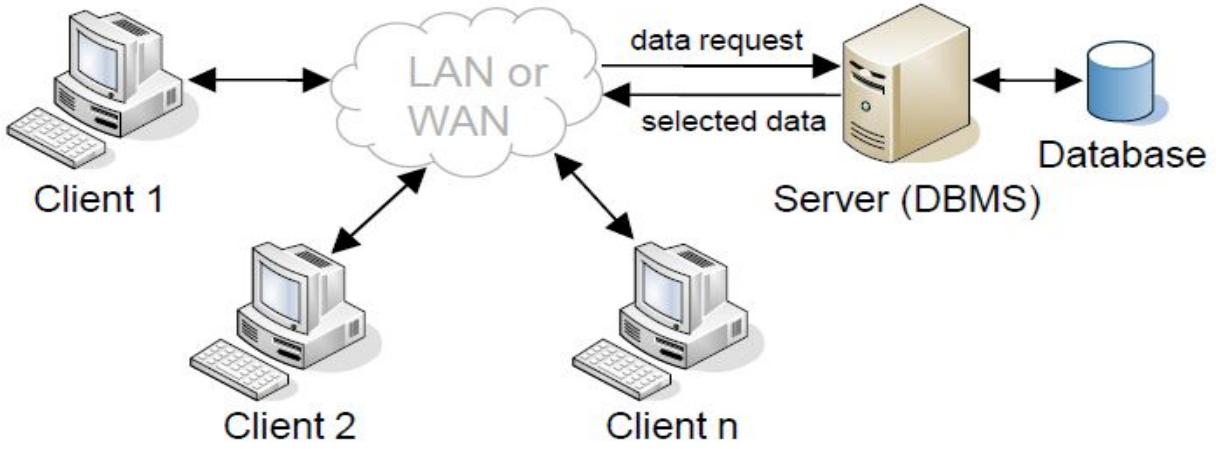
- файл-сервер (file-server) – компьютер, преимущественно использующийся для разделяемого хранения файлов;
- все или часть компонентов СУБД выполняются на компьютере пользователя, для этого может понадобиться передача файлов;
- недостатки:
  - высокая нагрузка на сеть;
  - сложные схемы обеспечения целостности данных, совместного доступа к данным и восстановления;
  - высокая стоимость владения (установка и обслуживание всех компонентов СУБД на каждом компьютере);

Двухзвенная система «клиент-сервер»



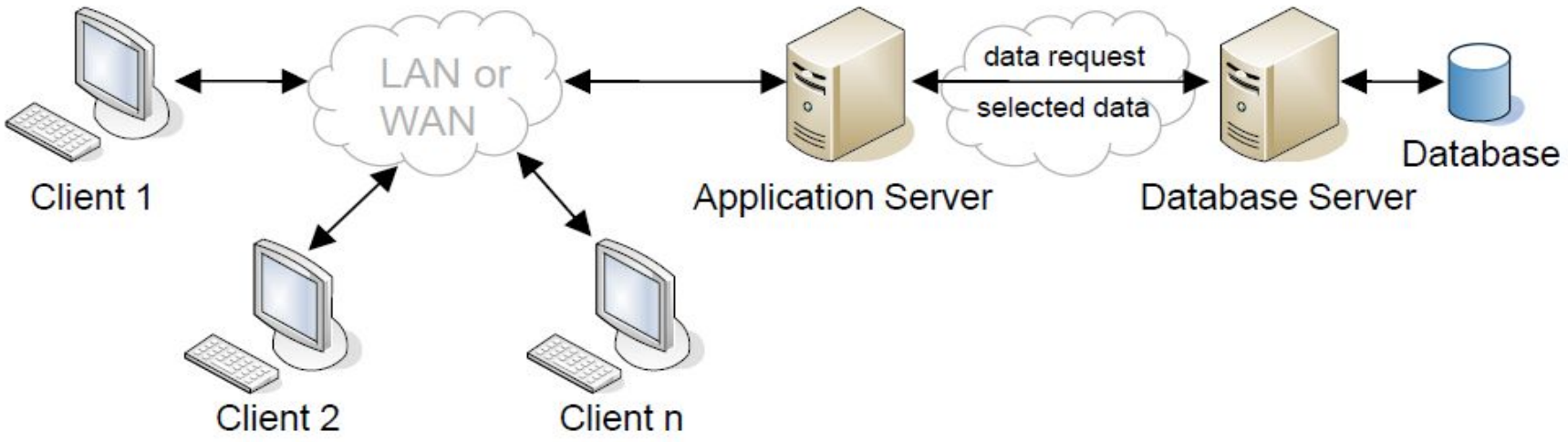
- на клиенте выполняется приложение пользователя, которое обращается к СУБД на сервере: толстый (thick) / тонкий (thin) клиент;
- функции клиента:
  - представление данных (пользовательский интерфейс);
  - выполнение некоторой бизнес-логики;
  - отправка запросов и прием результатов их выполнения;
- функции сервера:
  - выполнение бизнес-логики;
  - обеспечение совместного доступа к данным;

Двухзвенная система «клиент-сервер»: преимущества и недостатки



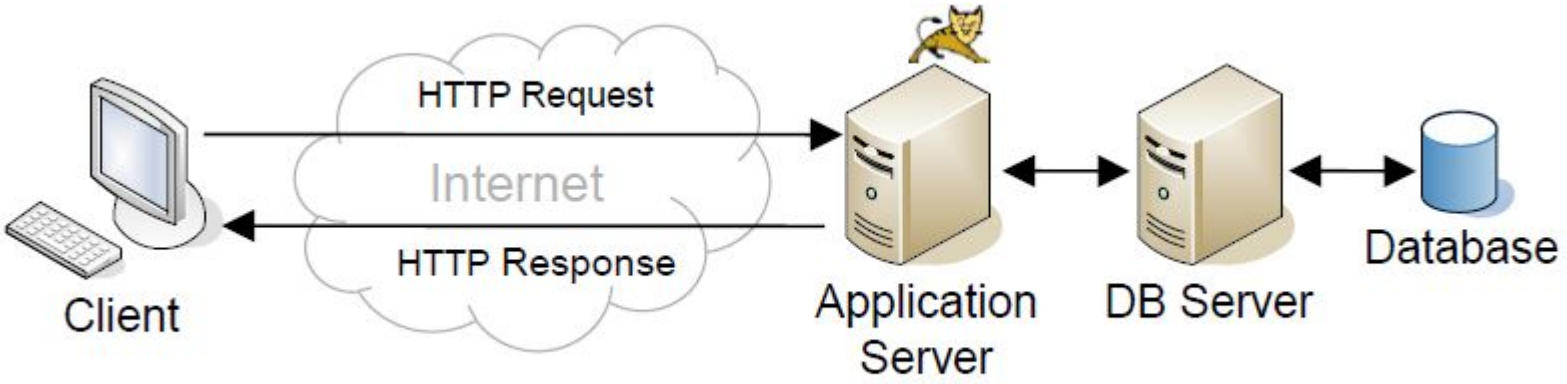
- преимущества:
  - возможность увеличения производительности (параллельное выполнение запросов и настройка сервера под конкретную СУБД);
  - возможность обеспечения целостности данных (централизованная проверка ограничений) и безопасности;
  - снижение сетевого трафика;
  - снижение стоимости владения;
- недостатки:
  - ограниченная возможность масштабирования;
  - большие издержки на поддержание клиентских приложений и требований к производительности (в случае толстого клиента);

Трехзвенная система «клиент-сервер»



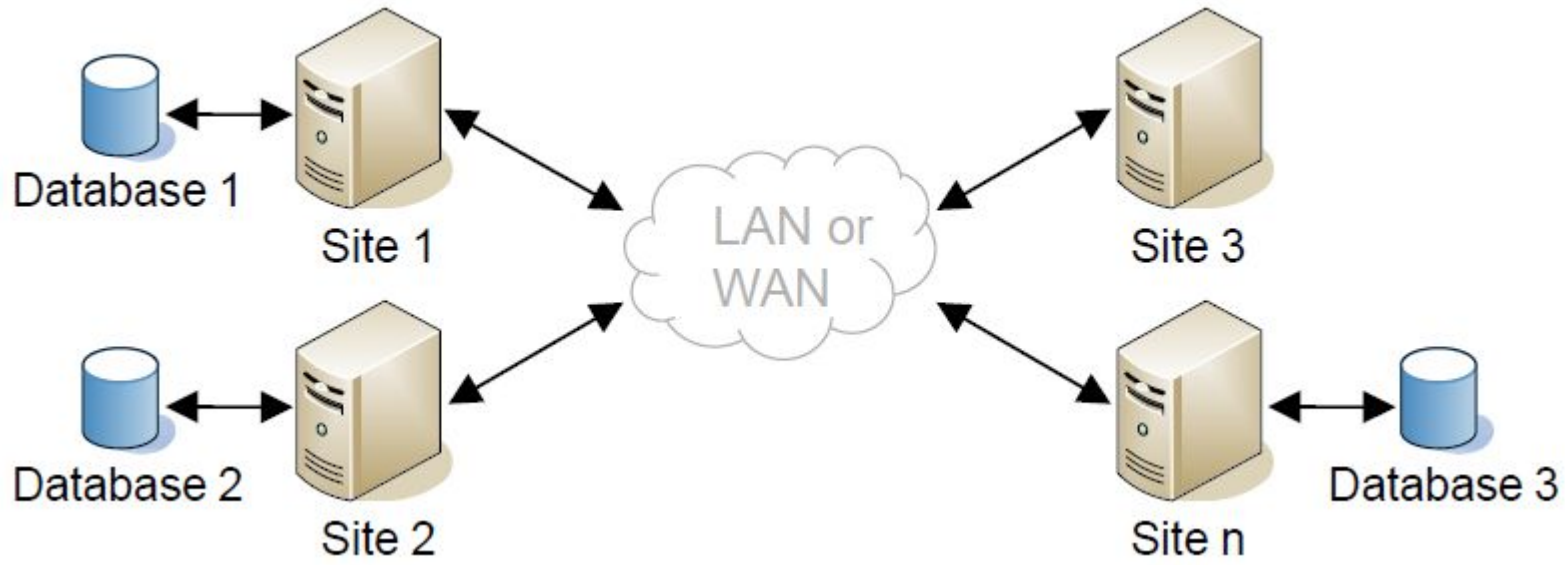
- появилась в 90х для обеспечения требований масштабируемости (например, для веб-приложений);
- приложение включает три уровня:
  - представления (presentation tier – клиент /client/):
    - представление данных (пользовательский интерфейс);
    - базовая проверка ввода (тонкий клиент);
    - отправка запросов и прием результатов их выполнения;
  - бизнес-логики (logic tier – сервер приложений /application server/):
    - выполнение бизнес-логики и обработка данных;
  - данных (data tier – сервер баз данных /database server/):
    - базовая проверка корректности данных;
    - обеспечение совместного доступа к данным;
    - обеспечение целостности данных.

Трехзвенная система «клиент-сервер» (2)



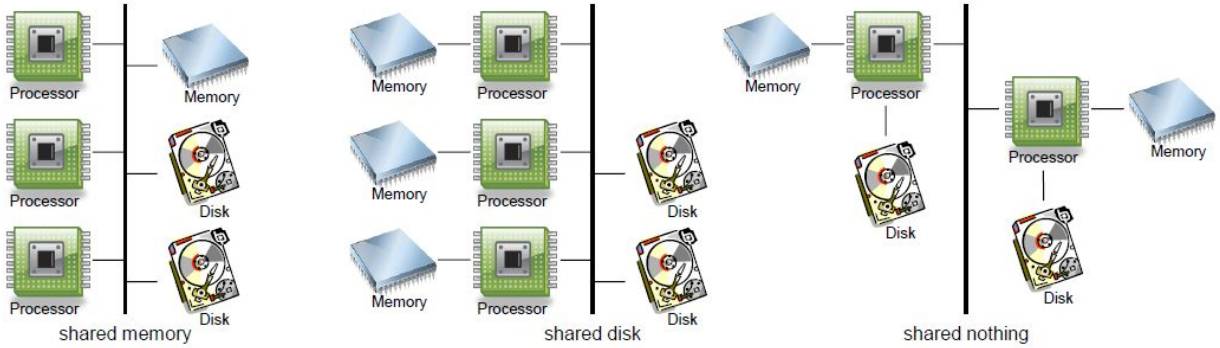
- преимущества:
  - снижение требований к производительности клиентов;
  - централизация бизнес-логики на сервере приложений (упрощается задача поддержки и установки обновлений);
  - увеличение модульности;
  - возможность балансировки нагрузок на каждом из уровней;
- пример: архитектура веб-приложений (баузер выступает в качестве тонкого клиента);
- возможно увеличение количества промежуточных уровней для повышения гибкости системы (например, повышения эффективности балансировки нагрузок).

Архитектура «точка-точка»



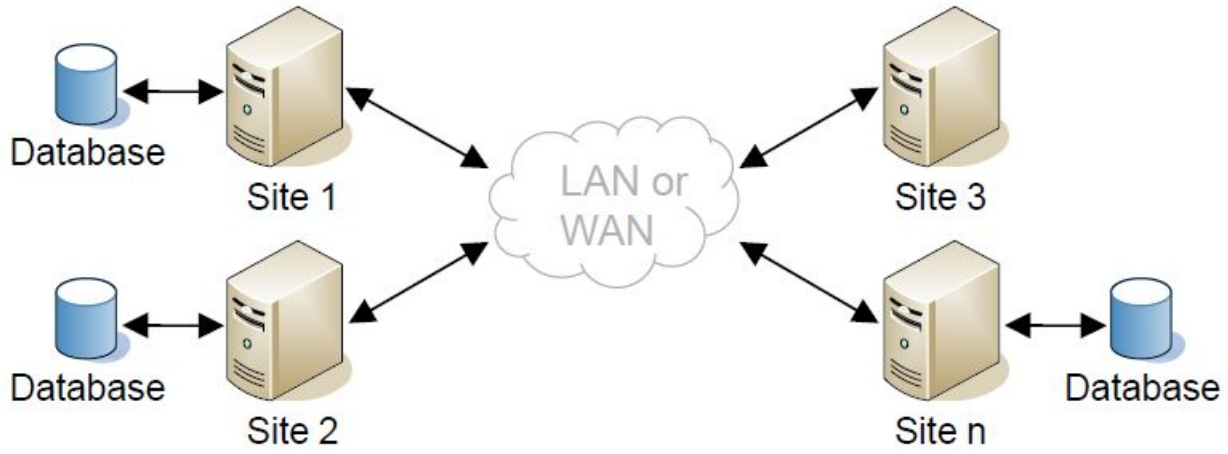
- отсутствует четко выраженное разделение клиентов и серверов – связи могут формироваться динамически для обмена данными и услугами;
- в связи с отсутствием выделенного центрального узла и глобальной схемы данных могут потребоваться дополнительные средства для скрытия разнородности данных/систем и обеспечения унифицированного представления (mediation) услуг и данных: различные протоколы и интерфейсы (Java RMI, CORBA, XML RPC, подписка на события, ODBC и т.п.) ;

Параллельные базы данных



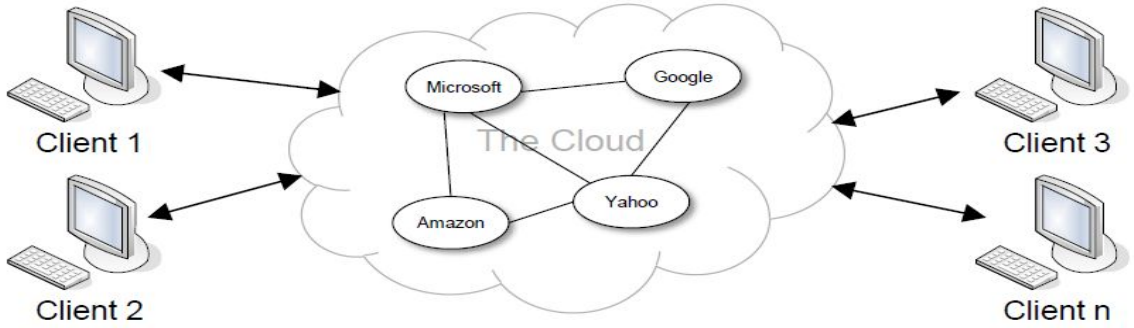
- позволяют повысить производительность за счет параллельного выполнения транзакций:
  - разделяемая память (shared memory):
    - крайне эффективный обмен между процессорами;
    - невозможность масштабирования (шина является узким местом);
  - разделяемое дисковое пространство (shared disk):
    - определенная степень отказоустойчивости при отказе памяти / процессора + отказоустойчивость дисковой подсистемы за счет организации RAID-массивов;
    - узким местом является обмен с дисковой подсистемой;
  - нет разделяемых ресурсов (shared nothing):
    - наиболее масштабируемая;
    - неравномерная скорость доступа к различным дискам;
  - иерархическая (hierarchical): комбинация вышеперечисленных подходов.

Распределенные базы данных



- распределенная база данных: логическая совокупность данных и метаданных, распределенная внутри сети;
- распределенная СУБД: система, обеспечивающая прозрачное управление распределенной базой данных;
  - средства распределенного выполнения транзакций;
  - средства согласования метаданных в случае объединения разнородных подсистем;
- преимущества:
  - возможность интеграции и прозрачного доступа к данным других подсистем;
  - высокая степень доступности данных (availability), особенно при использовании репликации;
  - более дешевые и масштабируемые, чем кластерные системы.

Облачные вычисления



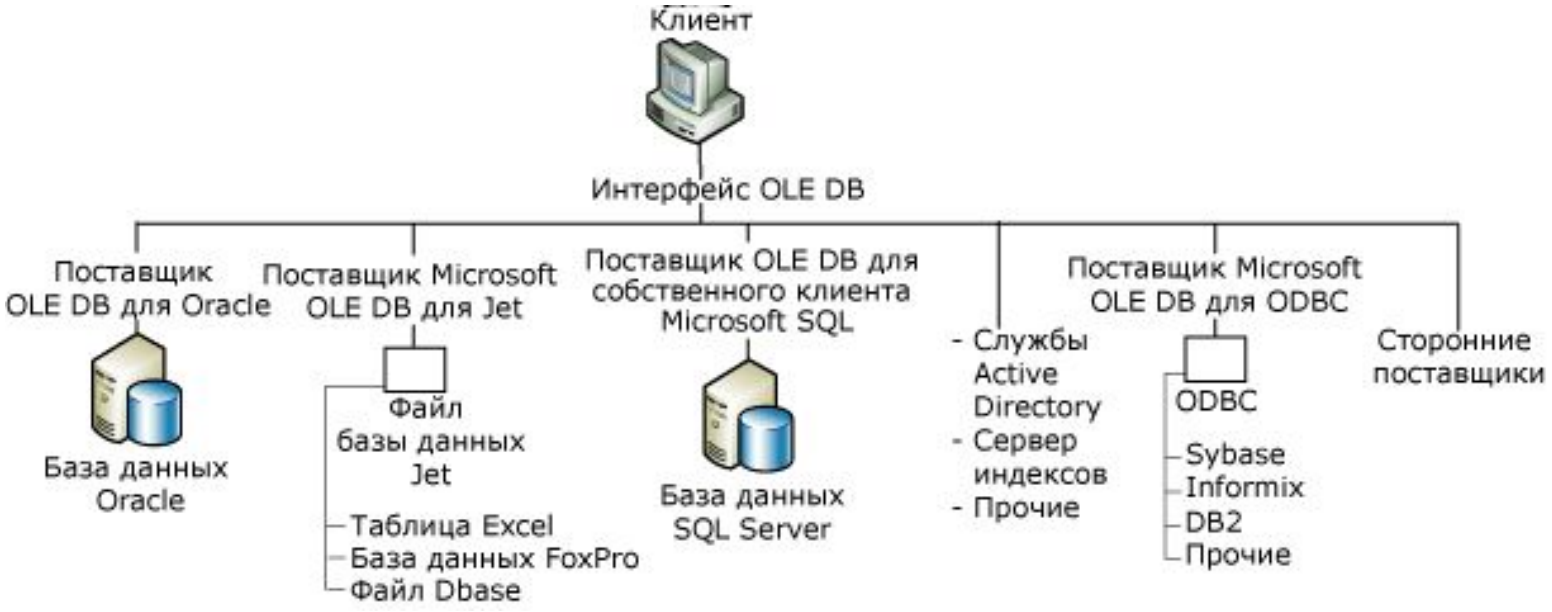
- облачные вычисления частично основаны на концепциях архитектуры, ориентированной на службы:
  - функциональность разбита на набор взаимодействующих служб:
    - службы не являются тесно связанными;
    - обеспечивается интероперабельность служб, реализованных на разных платформах;
    - службы могут инкапсулировать функции других служб произвольным образом;
  - задача может быть решена путем последовательного обращения к некоторому набору служб;
- модели служб:
  - инфраструктура как услуга (IaaS, Infrastructure as a Service): виртуализация физических ресурсов;
  - платформа как услуга (PaaS, Platform as a Service): операционная система, СУБД, веб-сервер;
  - программное обеспечение как услуга (SaaS, Software as a Service): приложения и базы данных;
- в области баз данных:
  - изменяются подходы к надежности, производительности, резервному копированию, восстановлению и т.п.
  - примеры: Amazon S3/EC2/SimpleDB, Windows Azure Tables/Blobs/SQL Databases/HDInsight.

Базы данных для мобильных устройств

- необходимость доступа к данным с мобильных устройств;
- при этом возникают относительно новые требования к приложениям баз данных:
  - компактность программного обеспечения для обеспечения возможности его функционирования на мобильных устройствах в условиях ограничения доступных ресурсов;
  - учет контекста и местоположения пользователей при формировании запросов;
  - различные способы подключения к центральному серверу баз данных;
  - необходимость синхронизации данных на центральном сервере и на мобильном устройстве (репликация данных);
  - необходимость учитывать возможные сбои в сети (кеширование данных и транзакций);
  - учет требований безопасности при репликации данных на мобильное устройство;
  - возможность обмена информацией между мобильными устройствами при появлении такой возможности (в режиме «точка-точка»).

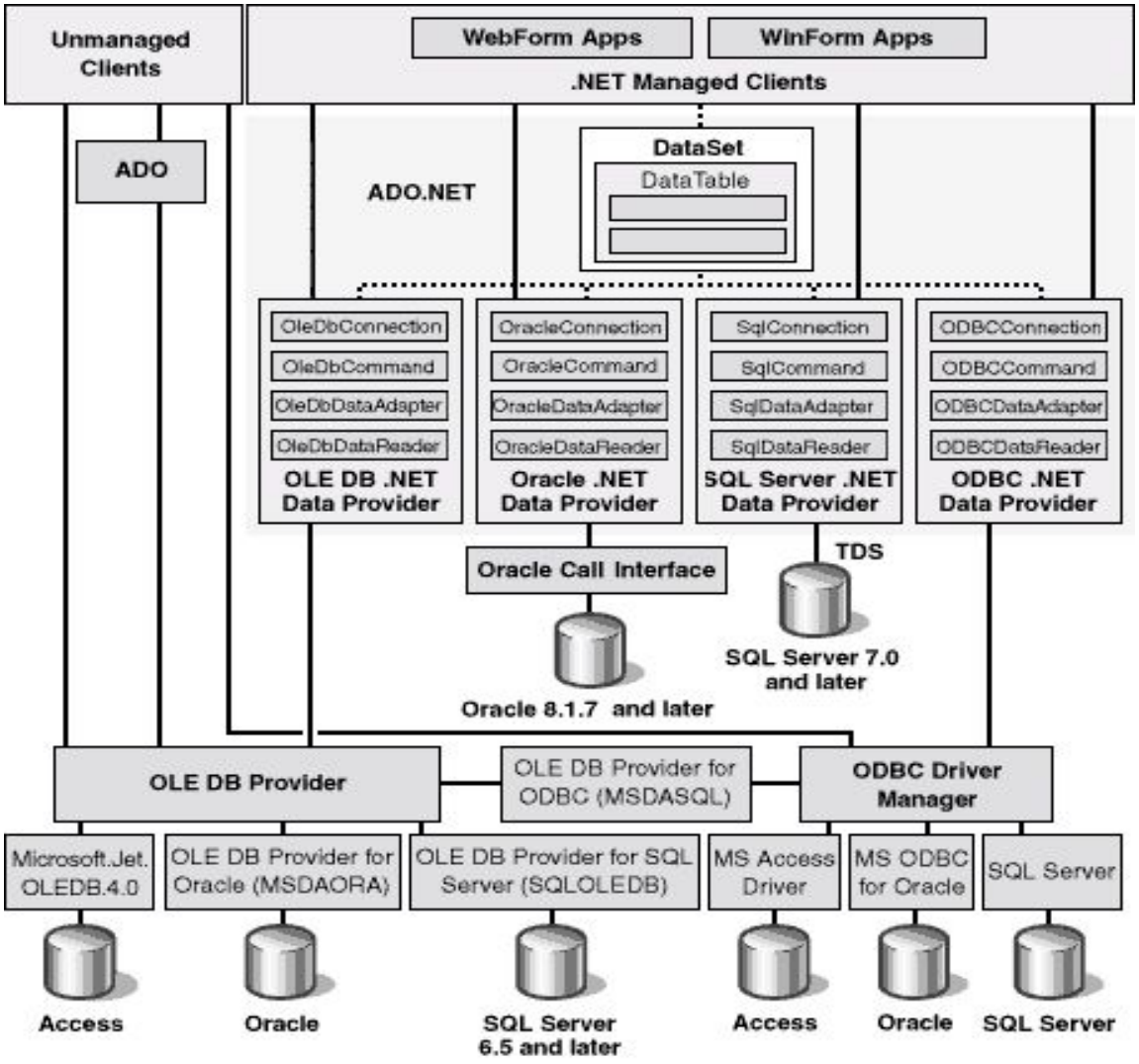
Взаимодействие приложений и СУБД

- приложения взаимодействуют с БД через программные интерфейсы (API):
  - стандартные (реализованы многими производителями СУБД)
    - ODBC;
    - OLEDB;
    - ADO.NET;
  - конкретного производителя СУБД (native / proprietary):
    - OCI (Oracle Call Interface)



Стек доступа к данным ADO.NET

Доступ к данным в технологии ADO.NET осуществляется путем обращения к специальным библиотекам - поставщикам (провайдерам, providers), реализующим определенный набор типов (прежде всего, классов и интерфейсов) доступа к данным



Выводы

- создание защищенных приложений баз данных является сложной инженерной задачей, при решении которой:
  - эффективность созданного программно-аппаратного комплекса во многом зависит от адекватного понимания разработчиком тех требований, которые предъявляет пользователь к данному комплексу;
  - необходимы знания в областях, включающих:
    - аппаратное обеспечение,
    - алгоритмы и структуры данных,
    - системное программирование,
    - сетевые технологии,
    - разработка пользовательского интерфейса.

Вопросы к экзамену

- Приложения баз данных. Требования к приложениям баз данных. Процесс разработки баз данных.
- Моделирование данных. Модель «сущность-связь» и модель семантических объектов: основные понятия.
- История развития баз данных. Системы обработки файлов и базы данных. Системы управления базами данных (СУБД).
- Понятие реляционной модели и нормализации. Реляционные базы данных. Метаданные в реляционных базах данных.
- Системы управления базами данных (СУБД). Ядро СУБД. Язык SQL. Дополнительные компоненты современных СУБД.
- Системы управления базами данных (СУБД). Создание баз данных и основные объекты ядра СУБД.
- Модели баз данных. Иерархическая и сетевая модели. Реляционная модель.
- Модели баз данных. Реляционная модель. Обзор постреляционных моделей (объектно-ориентированные, документо-ориентированные, графовые и столбцовые базы данных).
- Схемы доступа к данным. Терминальный доступ. Доступ в режиме разделения файлов. Архитектура «клиент-сервер».
- Схемы доступа к данным. Архитектура «точка-точка». Параллельные базы данных. Распределенные базы данных.
- Схемы доступа к данным. Облачные вычисления. Базы данных для мобильных устройств.
- Управление параллельной обработкой в БД. Приложения баз данных для оперативной обработки транзакций (OLTP) и поддержки принятия решений (OLAP).