БАЗЫ ДАННЫХ

4 семестр

Иванцова Ольга Владимировна

Основные понятия баз данных (БД)

БД – это система специальным образом организованных данных, программных, технических, языковых средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

Система управления БД (**СУБД**) – это совокупность языковых и программных средств, обеспечивающих для выполнение всех операций, связанных с организацией хранения данных, их корректирования и доступа к ним.

БД – это поименованная совокупность взаимосвязанных данных находящихся под управлением СУБД.

Требования к современным СУБД:

✓БД должна удовлетворять актуальным информационным потребностям организации. Получаемая информация должна по структуре и содержанию соответствовать решаемым задачам.

✓БД должна обеспечивать получение требуемых данных за приемлемое время, то есть отвечать заданным требованиям производительности.

✓БД должна удовлетворять выявленным и вновь возникающим требованиям конечных пользователей.

Требования к организации и управлению данными

Требования к организации данных:

1) **НЕ**избыточность,

2) Целостность = достоверность + **Не**противоречивость + полнота

3) **Не**зависимость данных от приложений – свойство БД, обеспечивающее устойчивость к ее развитию.

Требования к управлению данными:

1) эффективность доступа для каждого пользователя

2)защита данных

БД неизбыточна, если удаление какого-либо элемента данных или связи между данными ведет к потере информации о ПО. Наиболее неприятное последствие избыточности – увеличение трудоемкости обновления данных.

БД непротиворечива, если все хранящиеся в ней данные удовлетворяют определенным условиям (ограничениям целостности).

Классификация СУБД по способу доступа к БД

Файл-серверные

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. Ядро СУБД располагается на каждом клиентском компьютере.

Доступ к данным осуществляется через локальную сеть.

Синхронизация чтений и обновлений осуществляется посредством файловых блокировок.
Преимуществом этой архитектуры является низкая нагрузка на ЦП сервера,
а недостатками:
✓высокая загрузка локальной сети;
✓на каждой рабочей станции должна находиться полная копия СУБД.
✓управление параллельностью, восстановлением и целостностью усложняется, поскольку доступ к одним и тем же файлам могут осуществлять сразу несколько экземпляров СУБД.
На данный момент файл-серверные СУБД считаются устаревшими.
Примеры: Microsoft Access, Borland Paradox.

Классификация СУБД по способу доступа к БД

Клиент-серверные

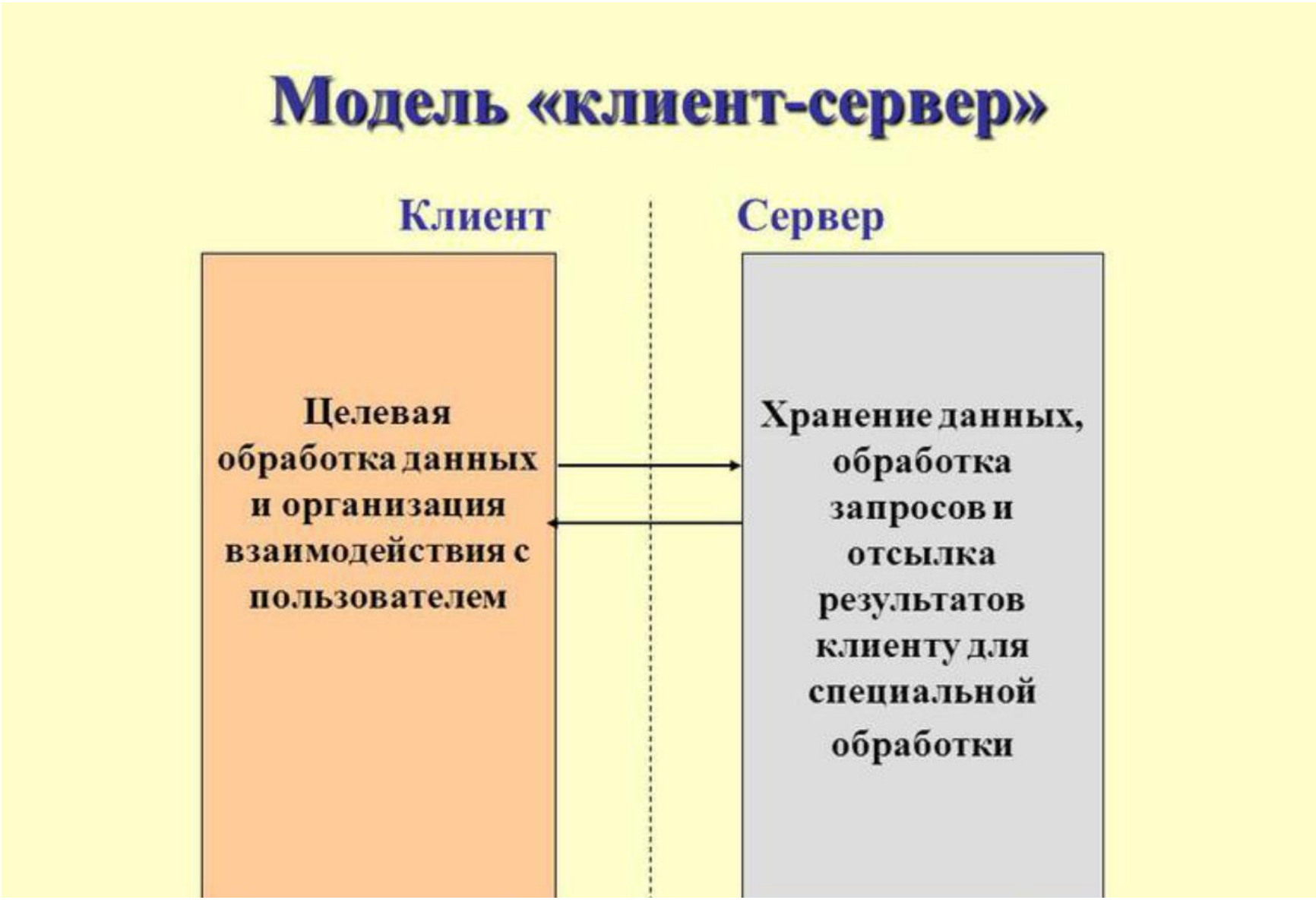
Такие СУБД состоят из клиентской части (которая входит в состав прикладной программы) и сервера.

Клиент-серверные СУБД, в отличие от файл-серверных, обеспечивают разграничение доступа между пользователями и мало загружают сеть и клиентские машины.

Сервер является внешней по отношению к клиенту программой, и по надобности его можно заменить другим.

Недостаток клиент-серверных СУБД в самом факте существования сервера (что плохо для локальных программ — в них удобнее встраиваемые СУБД) и больших вычислительных ресурсах, потребляемых сервером.
Примеры: Oracle, PostgreSQL, MS SQL Server, Sybase, MySQL.

Общая схема построения систем с архитектурой "клиент/сервер"



Клиент.

- ✓ Управляет пользовательским интерфейсом.
- ✓ Принимает и проверяет синтаксис введенного пользователем запроса.
- ✓ Выполняет приложение.
- ✓ Генерирует запрос к базе данных и передает его серверу.
- ✓ Отображает полученные данные пользователю.

Сервер:

- ✓ Принимает и обрабатывает запросы к базе данных со стороны клиентов.
- ✓ Проверяет полномочия пользователей.
- ✓ Гарантирует соблюдение ограничений целостности.
- ✓ Выполняет запросы/обновления и возвращает результаты клиенту.
- ✓ Поддерживает системный каталог.
- ✓ Обеспечивает параллельный доступ к базе данных.
- ✓ Обеспечивает управление восстановлением.

Модели клиент–сервер (Client–server architecture)

Архитектура клиент-сервер - архитектура распределенной вычислительной системы, в которой приложение делится на клиентский и серверный процессы.

В зависимости от того, как распределены логические компоненты приложения между клиентами и серверами, различают *следующие модели архитектуры клиент-сервер:*

- ✓модель "файл-сервер";
- ✓модель "сервер базы данных";
- ✓модель «удаленного доступа";
- ✓модель "сервер приложений".

Модель клиент-сервер (Client-server architecture)

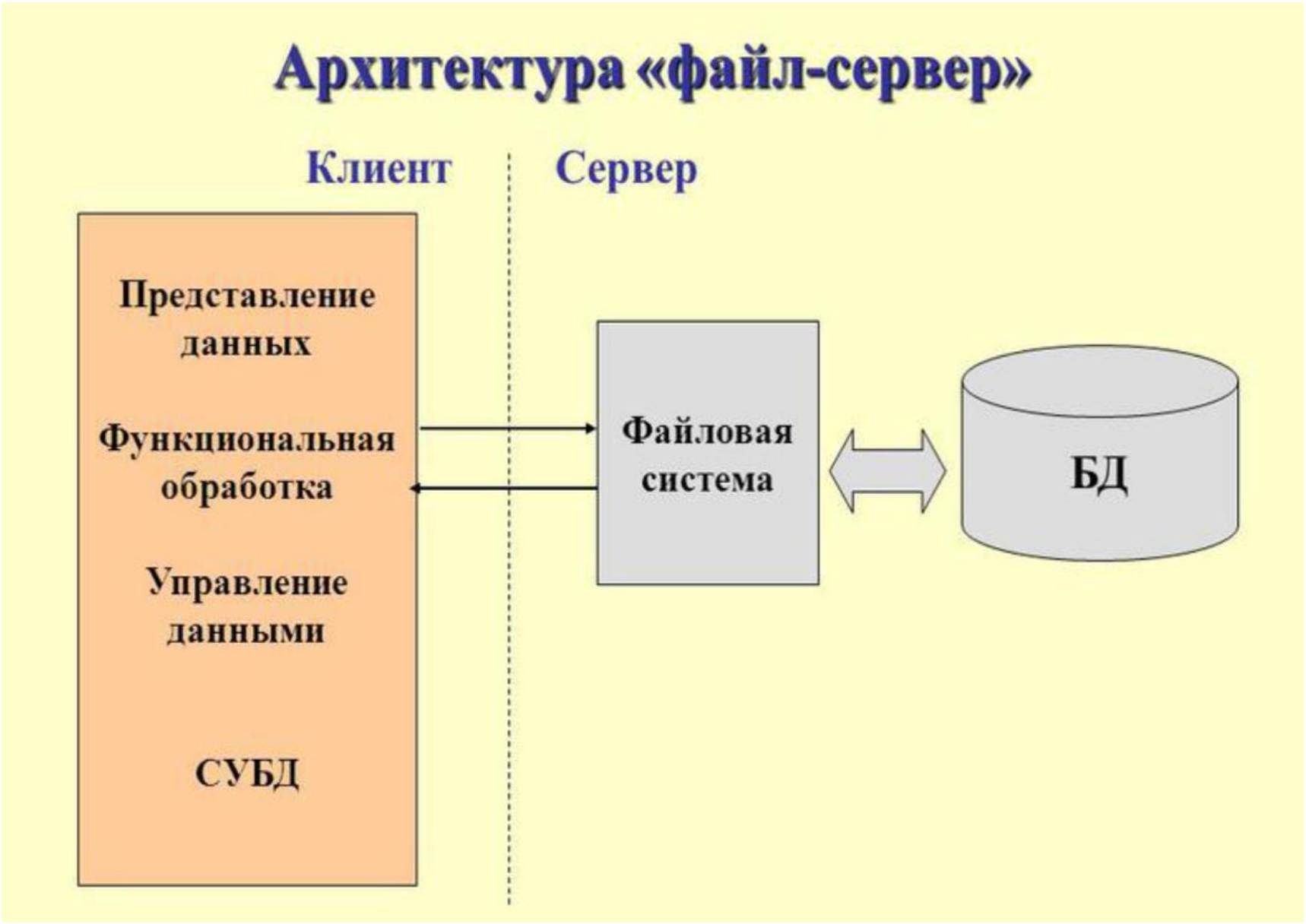
Модель "файл-сервер“

Модель "файл-сервер" - архитектура вычислительной сети типа "клиент-сервер", в которой сервер предоставляет в коллективное пользование дисковое пространство, систему обслуживания файлов и периферийные устройства.
Файловый сервер:
- обеспечивает управление доступом к файлам и базам данных;
- предоставляет в общее пользование дисковое пространство, принтеры модемы и другие ресурсы.

Модель "сервер базы данных“

Модель "сервер базы данных" - архитектура вычислительной сети типа "клиент-сервер", в которой *пользовательский интерфейс и логика приложений сосредоточены на машине-клиенте,* а *информационные функции (функции СУБД) - на сервере*. Обычно клиентский процесс посылает запрос серверу на языке SQL.

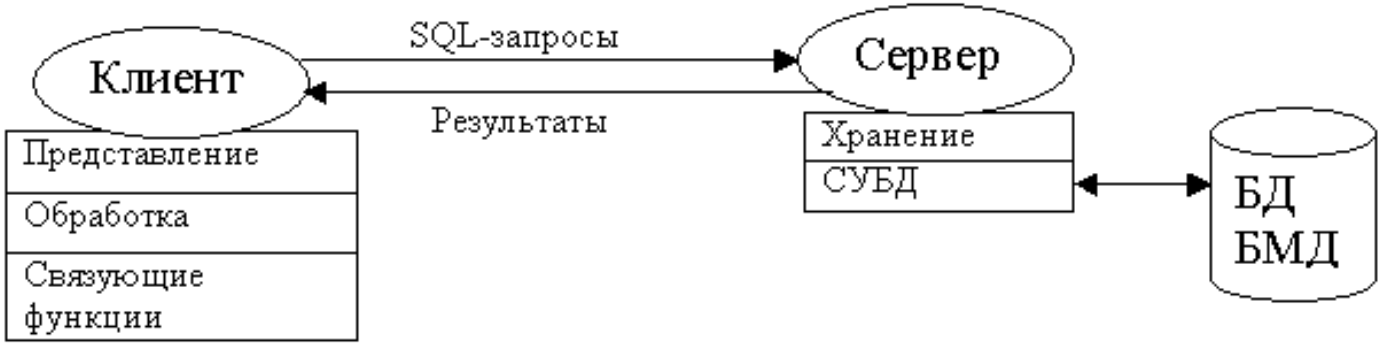
Архитектура с использованием файлового сервера



Модель удаленного доступа к данным

Отличием модели удаленного доступа к данным (Remote Data Access, RDA) от модели файлового сервера является то, что ядро СУБД расположено на сервере. Презентационная логика и бизнес-логика расположены на стороне клиента.
Достоинством данной модели можно считать значительное сокращение сетевого трафика, так как по сети передаются не запросы на ввод-вывод в файловой терминологии, а запросы на языке SQL
К недостаткам можно отнести:

- высокий сетевой трафик (несмотря на значительно сокращение сетевого трафика, по сравнению с модель файлового сервера, все-таки запросы на языке SQL при интенсивной работе клиентских приложений могут существенно загрузить сеть);
- дублирование кода приложений (запросы на получение одних и тех же данных присутствуют в виде копий в различных приложениях);
- пассивный сервер.



Модели клиент-сервер (Client-server architecture)

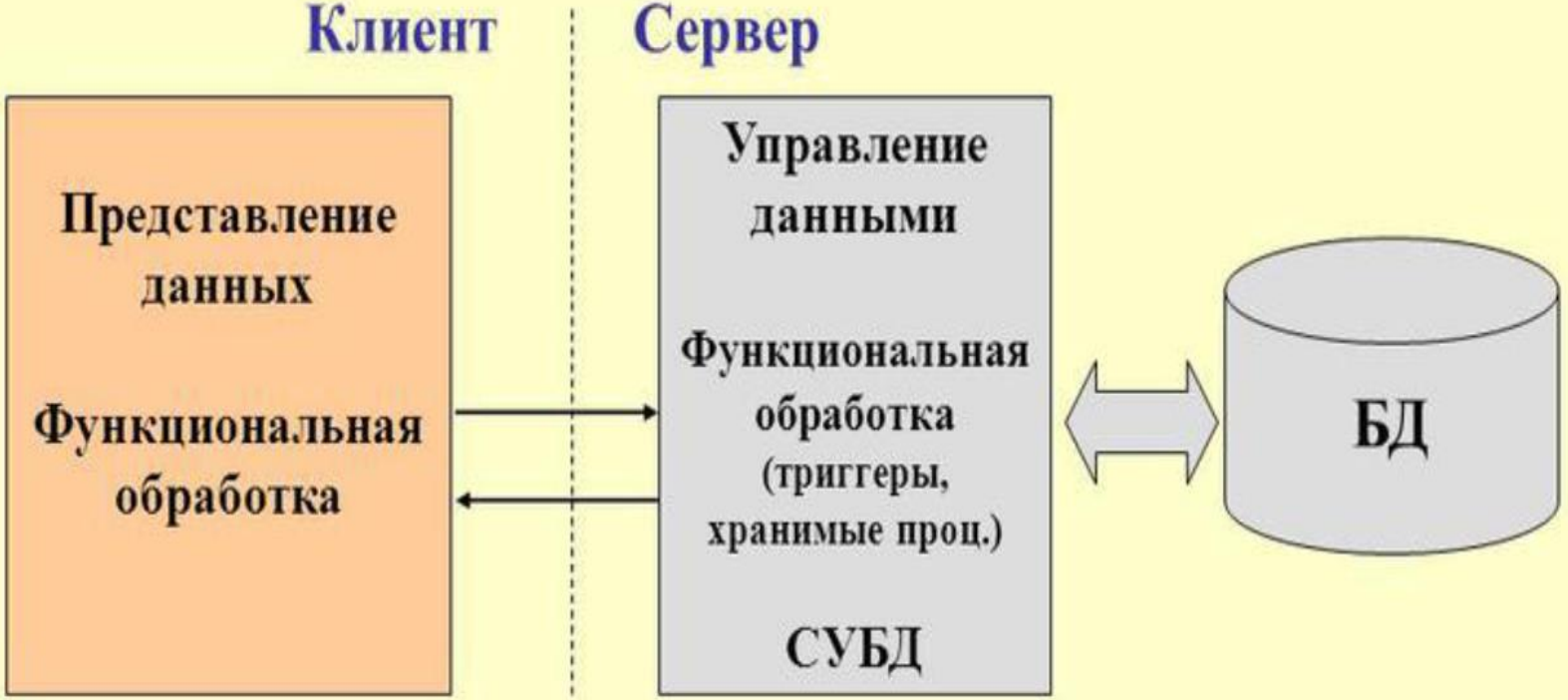
Модель "сервер базы данных“

Модель "сервер базы данных" - архитектура вычислительной сети типа "клиент-сервер", в которой *пользовательский интерфейс и логика приложений сосредоточены на машине-клиенте,* а *информационные функции (функции СУБД) - на сервере*. Обычно клиентский процесс посылает запрос серверу на языке SQL.

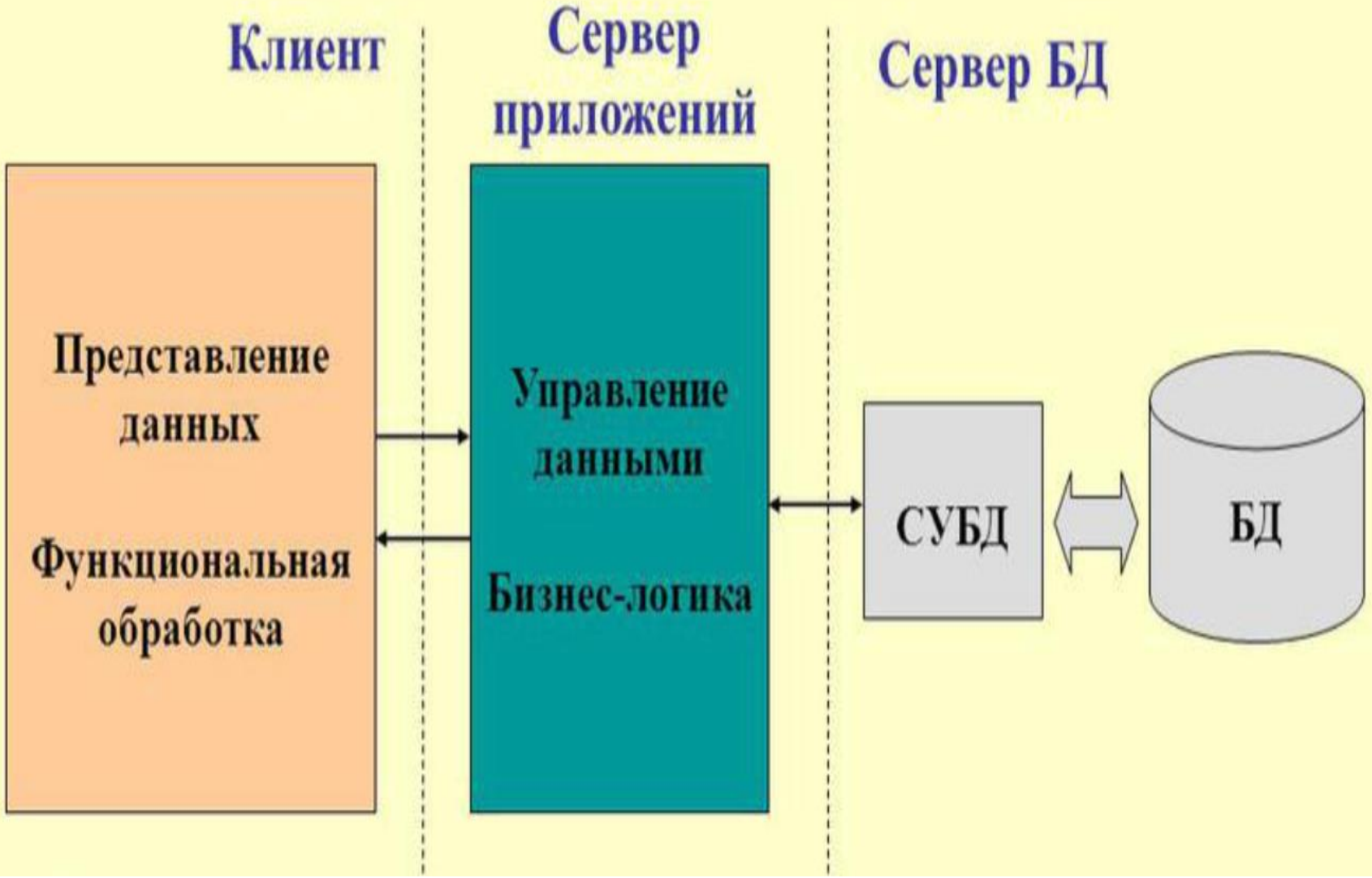
Модель "сервер приложений“

Модель "сервер приложений" - архитектура вычислительной сети типа "клиент-сервер", в которой функциональная логика размещена на сервере, а на машине-клиенте выполняется только компонент представления.

Архитектура «активный сервер БД»



Архитектура «сервер приложений»



Преимущества клиент-серверной архитектуры

- ✓ *Обеспечивается более широкий доступ к существующим базам данных*.
- ✓ *Повышается общая производительность системы* (клиенты и сервер находятся на разных компьютерах, их процессоры способны выполнять приложения параллельно).
- ✓ *Стоимость аппаратного обеспечения снижается*.
- ✓ *Сокращаются коммуникационные расходы*. Приложения выполняют часть операций на клиентских компьютерах и посылают через сеть только запросы к базе данных, что позволяет существенно сократить объем пересылаемых по сети данных.
- ✓ *Повышается уровень непротиворечивости данных*. Сервер может самостоятельно управлять проверкой целостности данных, поскольку все ограничения определяются и проверяются только в одном месте.
- ✓ *Эта архитектура хорошо согласуется с архитектурой открытых систем*.
- ✓ *Данная архитектура может быть использована для организации средств работы с распределенными базами данных*, т.е. с набором нескольких баз данных, логически связанных и распределенных в компьютерной сети.

Двухуровневая архитектура

Двухуровневая архитектура - архитектура приложения, в которой прикладные и пользовательские сервисы реализованы на клиентской рабочей станции, а данные централизованно хранятся на сервере.

В этой модели клиенты подключаются непосредственно к серверу, на все время работы приложения.

Трехуровневая архитектура

При данной архитектуре функциональная часть прежнего, толстого (интеллектуального) клиента разделяется на две части.

В трехуровневой архитектуре тонкий (неинтеллектуальный) клиент на рабочей станции управляет только пользовательским интерфейсом, тогда как средний уровень обработки данных управляет всей остальной логикой приложения.

Третьим уровнем здесь является сервер базы данных.

Эта трехуровневая архитектура оказалась более подходящей для некоторых сред, например, для сетей Internet и intranet, где в качестве клиента может использоваться обычный Web-броузер.

Архитектура базы данных. Физическая и логическая независимость

Уровень внешних моделей. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно ему.
Концептуальный уровень — центральное управляющее звено, здесь база данных представлена в наиболее общем виде, объединяет данные, используемые всеми приложениями, работающими с БД. Фактически концептуальный уровень отражает ***обобщенную модель предметной области*** (объектов реального мира), для которой создавалась БД.
Физический уровень — собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации.
Эта ***архитектура*** позволяет обеспечить ***логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3)*** независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных.

Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных.

Схема прохождения запроса

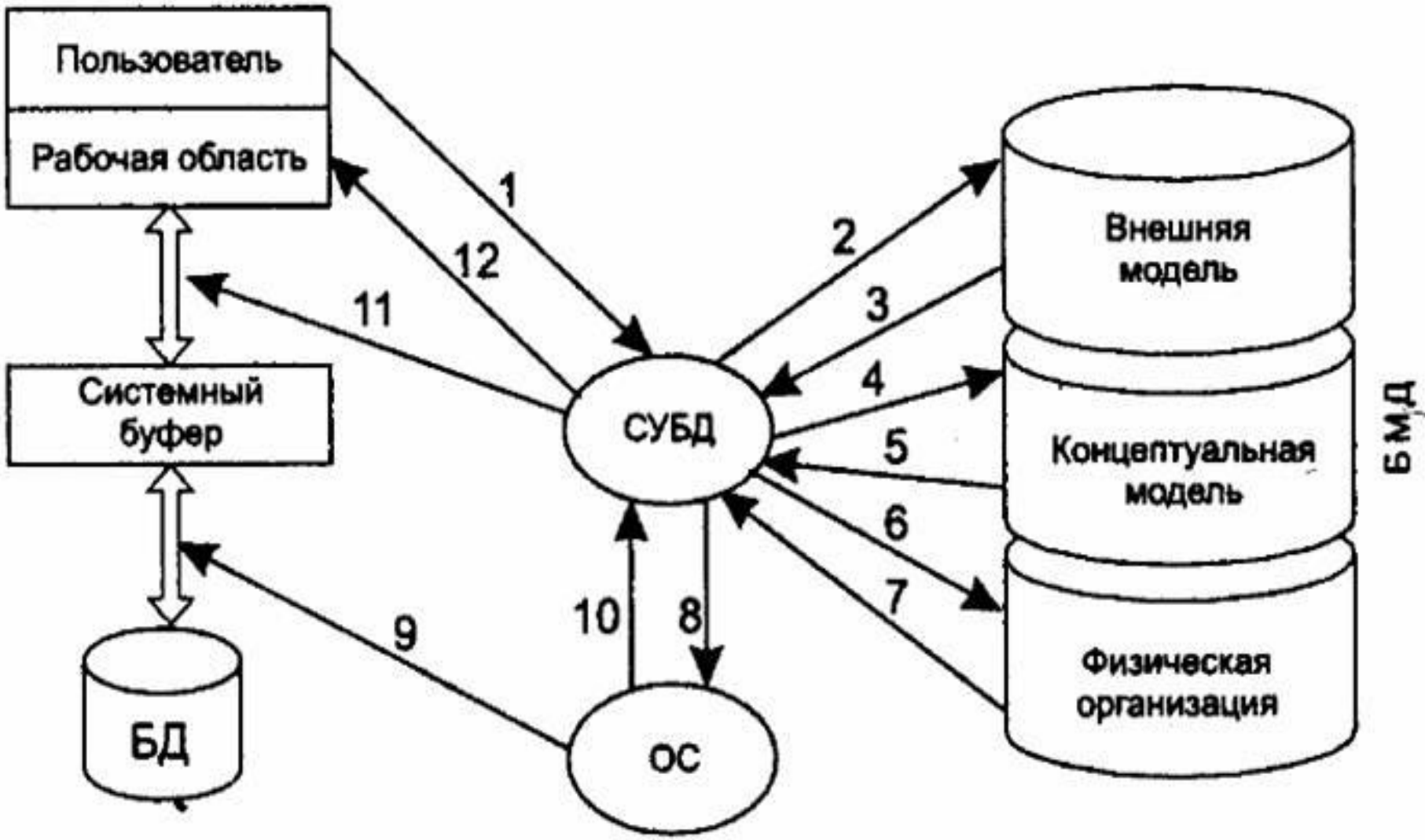


Схема прохождения запроса

1. Пользователь посылает СУБД запрос на получение данных из БД.
2. Анализ прав пользователя и внешней модели данных, соответствующей данному пользователю, подтверждает или запрещает доступ данного пользователя к запрошенным данным.
3. В случае запрета на доступ к данным СУБД сообщает пользователю об этом (стрелка 12) и прекращает дальнейший процесс обработки данных, в противном случае СУБД определяет часть концептуальной модели, которая затрагивается запросом пользователя (стрелка 4)
4. СУБД получает информацию о запрошенной части концептуальной модели.
5. СУБД запрашивает информацию о местоположении данных на физическом уровне (файлы или физические адреса).
6. В СУБД возвращается информация о местоположении данных в терминах операционной системы.
7. СУБД просит операционную систему предоставить необходимые данные, используя средства операционной системы.
8. Операционная система осуществляет перекачку информации из устройств хранения и пересылает ее в системный буфер.
9. Операционная система оповещает СУБД об окончании пересылки.
10. СУБД выбирает из доставленной информации, находящейся в системном буфере, только то,
что нужно пользователю, и пересылает эти данные в рабочую область пользователя.

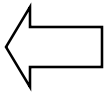
БМД — это *База Метаданных,* именно здесь и хранится вся информация об используемых структурах данных, логической организации данных, правах доступа пользователей и, наконец, физическом расположении данных.

Основные функции и возможности СУБД на примере системы ORACLE

1. Независимость от платформы .
2. Независимость от сетевого обеспечения .
3. Независимость от конечного окружения и графического интерфейса .
4. Независимость от баз данных (открытые шлюзы к DB2, Rdb, ADABAS, IDMS, RMS, SQL/DS, IMS) .
5. Полный набор функций среди RСУБД:

- ✓триггеры БД;
- ✓хранимые процедуры ;
- ✓декларативные ограничения целостности ;
- ✓расширенные возможности защиты;
- ✓ поддержка распределенных БД ;
- ✓параллельная обработка ;
- ✓лидирующие показатели по производительности.

6. Широкий спектр средств разработки (POWER OBJECTS, DESIGNER, DEVELOPER, DISCOVERER, CASE инструментарий).
7. Язык PL/SQL, предкомпиляторы (ADA, COBOL, C, FORTRAN, PASCAL, PL/1).
8. Полный набор приложений .
9. Связь с INTERNET (ORACLE Webserver) .
10. Поддержка национальных языков (NLS).



Факторы успеха баз данных Oracle

Серверы баз данных Oracle обладают целым рядом несомненных преимуществ:

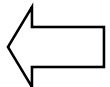
Безопасность и защита данных
В базах данных Oracle имеются высокоэффективные механизмы, позволяющие контролировать предоставление прав доступа к конфиденциальной информации.

В зависимости от имени, под которым пользователь входит в систему, он получает жестко ограниченные права на просмотр, модификацию и создание строго определенных видов данных.

Резервное копирование и восстановление данных
Базы данных Oracle имеют совершенные механизмы резервного копирования и восстановления данных.

Резервное копирование данных означает создание запасной копии информации, хранящейся в базе данных Oracle.

При восстановлении эта копия переписывается обратно в базу данных Oracle Server также обеспечивает систему резервного копирования и восстановления данных, при которой гарантируется круглосуточная доступность базы данных ≈ 7 дней в неделю, 24 часа в день и 365 дней в году.



Факторы успеха баз данных Oracle

Управление дисковым пространством
Oracle предлагает пользователям гибкие средства управления имеющимся дисковым пространством. Например, можно дать указание серверу, чтобы тот при первоначальном распределении дискового пространства зарезервировал определенную его часть для будущего использования и затем контролировал последующее распределение свободного места. Сервер также имеет ряд возможностей, специально ориентированных на управление особо большими базами данных.

Открытые стандарты межсистемного взаимодействия
Серверы Oracle поддерживают открытость стандартов, обеспечивающих взаимодействие с программными продуктами других фирм. Используя дополнительные приложения Oracle, пользователи могут работать с данными, накопленными в базах данных DB2 (компании IBM), Sybase или Microsoft Access. Кроме того, существует возможность доступа к хранящейся в базах данных Oracle информации и ее обработки с использованием таких программ, как Microsoft Visual Basic, PowerBuilder компании Powersoft и SQL*Windows фирмы Gupta.

Средства разработки приложений
Сервер баз данных Oracle, часто называемый ядром баз данных Oracle, поддерживает широкий спектр средств разработки приложений, средств генерации запросов конечными пользователями, готовых прикладных систем, а также средств управления информацией в масштабе офиса.

Объекты базы данных

Схема - это совокупность логических структур данных (*объектов схемы*).

Схема принадлежит пользователю базы данных и имеет то же имя, что и пользователь.

Каждому пользователю принадлежит единственная схема.

Все объекты базы данных и составляющие их части имеют имена и подразделяются на *объекты схемы* и *объекты не принадлежащие схеме*.

Объекты схемы (Schema Objects)

Типы объектов схемы

кластеры (clusters), каналы связи базы данных (database links), триггеры базы данных (database triggers), индексы (indexes), пакеты (packages), последовательности (sequences), снимки (snapshots), журналы снимков (snapshot logs), хранимые функции (stored functions), хранимые процедуры (stored procedures), синонимы (synonyms), таблицы (tables), представления (views).

Объекты не принадлежащие схеме (Non-Schema Objects)

Другие типы объектов также хранятся в базе данных и могут быть созданы и изменены с помощью SQL, однако, они не принадлежат никакой схеме: профили (profiles), роли (roles), сегменты отката (rollback segments), табличные пространства (tablespaces), пользователи (users).

Составляющие объектов (Parts of Objects)

Некоторые объекты состоят из составляющих, которые также должны иметь имена.

Это: столбцы (columns) в таблице или представлении, ограничения целостности таблицы (integrity constraints on a table), пакетные процедуры (packaged procedures), пакетные хранимые функции (packaged stored functions) и другие объекты пакетов

Файлы данных

Файлы данных содержат всю информацию, хранящуюся в БД.

БД Oracle состоят из одного или нескольких файлов данных, сгруппированных в табличные пространства. Важно понимать, что в файлах данных хранится *абсолютно вся информация, содержащаяся в базе данных*

Системные данные и данные пользователя
Всю информацию, хранящуюся в файлах данных базы данных, можно разделить на две категории: *системные данные* и *данные пользователя*.

Данные пользователя ≈ это данные, используемые приложениями пользователей. Именно к этой категории принадлежит вся относящаяся к пользовательским приложениям информация, хранимая в базе данных.

Системные данные ≈ это информация, необходимая базе данных для управления пользовательскими данными и для обеспечения собственной работоспособности. Системные данные Oracle содержат список всех пользователей, допущенных к работе с базой данных, их пароли, а также количество файлов данных, образующих базу данных, и их местонахождение.

Наиболее распространенные виды данных пользователя

Вид данных	Содержащаяся информация
Данные о клиенте	Фамилия, имя, номер телефона
Данные о товаре	Название товара, наличие на складе, цена
Медицинские данные	Результаты анализов, фамилия врача, фамилия медсестры
Складские данные	Количество товара, имеющееся в наличии и заказанное у поставщика
Финансовые данные	Курс акций и выплачиваемый по ним процент дохода

Наиболее распространенные виды системных данных

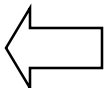
Вид данных	Содержащаяся информация
Таблицы	Названия полей таблицы и вид информации, которая может в них содержаться
Место на диске	Размер области дискового пространства, необходимой для хранения объектов базы данных
Пользователи	Имена, пароли и полномочия
Файлы данных	Номера, местоположение, дата и время последнего обращения к ним

Табличные пространства

Табличное пространство содержит один или несколько файлов данных.

Наличие следующих табличных пространств обязательно или желательно в большинстве баз данных:

- ✓ ***Системное*** табличное пространство system содержит информацию, необходимую для функционирования базы данных Oracle. Это табличное пространство является обязательным.
- ✓ ***Временное*** табличное пространство temp является рабочей областью Oracle.
- ✓ ***Вспомогательное*** табличное пространство tools применяется для хранения объектов, необходимых для вспомогательных программ, которые используются при работе с базой данных Oracle.
- ✓ ***Пользовательское*** табличное пространство users содержит объекты базы данных, принадлежащие отдельным пользователям.
- ✓ В табличном пространстве ***отката rollback***, как правило, хранятся особые объекты базы данных, называемые сегментами отката.
- ✓ Табличные пространства ***данных и индексов*** служат для хранения информации, используемой приложениями.



Табличные пространства

При полном табличном просмотре сервер базы данных Oracle читает все строки данных, связанных с конкретным объектом базы данных.

Данные для отката хранятся в специальном объекте базы данных, называемом сегментом отката.

Сегменты отката используются для восстановления прежнего значения объекта базы данных при ошибочно выполненной или незавершенной транзакции.

Табличная область – это раздел или логическая область памяти в базе данных, непосредственно соответствующая одному или более физическим файлам данных.

Цепочка *Таблица* - *Табличная область* - *Файл данных* обеспечивает логическую и физическую независимость данных.

Логическая и физическая независимость данных

Логическая независимость данных означает, что общая логическая структура данных может быть изменена без изменения прикладных программ (изменение, конечно, не должно заключаться в удалении из базы данных таких элементов, которые используются прикладными программами).

Физическая независимость данных означает, что физическое расположение и организация данных могут изменяться, не вызывая при этом изменений ни общей логической структуры данных, ни прикладных программ.

Логическая независимость достигается: раздельным хранением данных (метаданных), собственно данных и приложений. Взаимодействие осуществляют СУБД и ОС.

Физическая независимость достигается: многоуровневым описанием организации данных.

Приложение – программа, обеспечивающая автоматизацию обработки данных для прикладной задачи.

Транзакции

Транзакция – это неделимое действие над данными, хранящимися в БД.

Транзакции обеспечивают наибольшую защиту существующим данным.

Транзакции могут использоваться отдельно, либо вместе с буферизацией записи или таблицы.

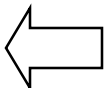
Только таблицы, которые находятся в БД, могут воспользоваться транзакциями.

Свойства транзакций:

- ✓ ***Атомарность*** – транзакция должна быть выполнена в целом или не выполнена вовсе.
- ✓ ***Согласованность*** – по мере выполнения транзакции данные переходят из одного согласованного состояния в другое.
- ✓ ***Изолированность*** – транзакции выполняются последовательно, изолированно друг от друга.
- ✓ ***Долговечность*** – если транзакция завершена успешно, то изменения не могут быть потеряны.

Команды управления транзакциями

COMMIT – зафиксировать транзакцию .
ROLLBACK – отменить изменения в текущей транзакции .
SAVEPOINT – создать контрольную точку внутри транзакции.



Основы администрирования и восстановления данных, сегменты отката

Сегмент отката - это область памяти на диске, которая используется для временного хранения старых значений данных, обновляемых транзакцией.

COMMIT и ROLLBACK

Когда над базой данных выполняется оператор COMMIT (завершить), транзакция заканчивается и:

- ✓ Вся работа, совершенная этой транзакцией, сохраняется.
- ✓ Другие сеансы могут видеть, какие изменения были внесены этой транзакцией.
- ✓ Все блокировки, установленные этой транзакцией, снимаются.

Синтаксис оператора COMMIT таков:

COMMIT [WORK];

Необязательное ключевое слово WORK (работа) применяется для повышения удобочитаемости программ.

Когда над базой данных выполняется оператор ROLLBACK (откатить), транзакция заканчивается и:

- ✓ Вся работа, совершенная этой транзакцией, отменяется, как если бы транзакция и не выполнялась.
- ✓ Все блокировки, установленные этой транзакцией, снимаются.

Методология резервирования/восстановления

Резервирование базы данных — вынужденное действие, направленное на сохранение данных для возможного в будущем восстановления базы на случай такого рода повреждения вычислительной системы, когда механизмы автоматического восстановления Oracle не могут справиться с ситуацией.

Постоянная готовность к спасению базы данных, регулярное проведение мероприятий по резервированию базы данных составляют значительную часть деятельности АБД.

Причины сбоя, вызвавшие необходимость проведения процедур восстановления базы данных, сводятся к следующим основным типам:

- ✓ Сбой экземпляра базы данных;
- ✓ Ошибки пользователя;
- ✓ Отказ физической среды.

Методология резервирования/восстановления

Процедура восстановления при **сбое экземпляра** осуществляется автоматически в процессе повторного его запуска.

Любые незафиксированные транзакции, имевшие место на базе данных в момент сбоя, откатываются.

Случайно **пользователь** может сделать такие ошибки, которые не могут быть исправлены откатом транзакций назад. Обычно это связано с применением DDL-команд, подобно DROP TABLE. Такие ситуации не влияют на работу системы в целом, хотя трагичны для конкретных пользователей. Восстанавливаются на резервной базе данных из резервированных данных и затем экспортируются в главную базу данных.

Отказ физической среды может быть связан с различными причинами, но если при этом не произошло повреждение дисков или невосстановимого повреждения компьютера в целом, то восстановление происходит подобно восстановлению экземпляра.

Существует *три стандартных метода резервирования баз данных Oracle*:

1. ***Полный экспорт*** — метод логического резервирования.
2. ***Автономное*** (NOARCHIVELOG).
3. ***Оперативное*** (ARCHIVELOG) — методы физического резервирования.

Журналы транзакций

Журналы транзакций ≈ это специальные файлы операционной системы, в которые Oracle записывает все изменения или транзакции, произведенные в базе данных.

В итоге на диск записывается лишь окончательная версия измененных данных.

Поскольку все транзакции полностью сохраняются в журналах повтора, при необходимости с помощью этих журналов *сервер БД всегда способен восстановить свое состояние на заданный момент времени*.

Каждая БД Oracle обязательно должна иметь как минимум два оперативных журнала транзакций.

Принципы организации журналов повтора
Журналы повтора работают по циклическому принципу.

По мере того, как транзакции создают, удаляют и модифицируют информацию в базе данных, все изменения заносятся в logA. Когда logA оказывается целиком заполненным, происходит переключение журналов, и все вновь произведенные транзакции начинают записываться в logB. По заполнении 1ogB происходит новое переключение журналов, и транзакции опять сохраняются в logA.

Журналы транзакций

Режим ARCHIVELOG: возможность полного восстановления

При работе базы данных в режиме ARCHIVELOG все журналы повтора транзакций сохраняются. Таким образом, перед началом перезаписи оперативного журнала повтора в результате очередного циклического переключения его прежнее содержимое копируется в другой файл.

В случае, если база данных не успела завершить подобное копирование, сервер Oracle приостанавливает все операции до его завершения.

В режиме ARCHIVELOG сервер Oracle не позволит переписать старый журнал транзакций, пока не будет сделана его архивная копия.

Т. е. БД хранит информацию обо всех произведенных транзакциях, что позволяет выполнить ее восстановление после любых типов сбоев, включая ошибочные действия пользователей или выход из строя жесткого диска.

Это самый безопасный режим функционирования базы данных.
При работе базы данных в режиме ARCHIVELOG содержимое всех журналов транзакций сохраняется перед началом их перезаписи.

Журналы транзакций

Режим NOARCHIVELOG

При работе базы данных в режиме NOARCHIVELOG (устанавливается по умолчанию) сохранение старых журналов транзакций не производится.

Т. к. в данном случае имеется информация лишь о последних по времени транзакциях, восстановление базы данных возможно только после некоторых типов сбоев, например, после внезапного отключения питания.

В режиме NOARCHIVELOG *сохранение архивных копий журналов транзакций перед их перезаписью НЕ ПРОИЗВОДИТСЯ* ,таким образом, в режиме NOARCHIVELOG имеются лишь ограниченные возможности по восстановлению после сбоев.

Данные и модели данных в концепции БД

Понятие «данные»– это набор конкретных значений, параметров, характеризующих объект, условие, ситуацию или любые другие факторы.
Данные: Иванова Анна Сергеевна, SU 4567, 25% и т.д.

Данные – не то же самое, что информация.

Информацию получают на основе данных, задав им определенную структуру.

Поэтому центральным понятием в области баз данных является понятие модели данных.

Модель данных – это некая абстракция, которая, будучи сопоставлена с конкретными данными, позволяет трактовать их как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними.

Классификация моделей данных

Классификация СУБД по модели данных

По типу управляемой базы данных СУБД разделяются на:

- ✓ Иерархические.
- ✓ Сетевые.
- ✓ Реляционные.
- ✓ Объектно-реляционные.
- ✓ Объектно-ориентированные.

Первые системы управления базами данных использовали иерархическую модель данных, и во времени их появление предшествует появлению сетевой модели.

Реляционная модель данных

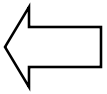
В реляционной модели данных объекты и взаимосвязи представляются с помощью связанных прямоугольных таблиц.

Каждая таблица представляет один объект и состоит из строк и столбцов (особенность таблицы - количество строк заранее неизвестно).

Все столбцы имеют имена, количество столбцов строго фиксировано, каждый столбец имеет свой тип и размер.

В реляционной модели каждая таблица должна иметь первичный ключ (ключевой элемент) − поле или комбинацию полей, которые единственным образом идентифицируют каждую строку в столбце.

Благодаря своей простоте и естественности представления реляционная модель данных получила наибольшее распространение в СУБД для персональных компьютеров.



Реляционная модель данных

Внешний ключ существует для связи нескольких таблиц.

Вся информация в реляционных базах данных представляется значениями в *таблицах (table)*.

В реляционных системах таблицы состоят из горизонтальных *строк (row)* и вертикальных *столбцов (column)*.

Иногда можно встретить такие понятия, как *отношение (relation), кортеж (tuple)* и *(attribute)*. Это соответственно синонимы понятий таблица, строка и столбец.

Доменом называется набор значений элементов данных одного типа, отвечающий поставленным условиям.

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, который забраковывает недопустимые значения.

Реляционная модель данных

Формальный реляционный термин	**Неформальный эквивалент**
Отношение	Таблица
Кортеж	Строка
Кардинальное число	Количество строк
Атрибут	Столбец
Степень	Количество столбцов
Первичный ключ	Уникальный идентификатор
Домен	Совокупность допустимых значений

Фундаментальные свойства отношений

- ✓ в одном отношении нет абсолютно одинаковых кортежей;
- ✓ все кортежи в отношении не упорядочены;
- ✓ все атрибуты в отношении не упорядочены;
- ✓ все значения атрибутов атомарные.

Ограничения целостности

Поддержание целостности базы данных может рассматриваться как защита данных от неверных изменений или разрушений (не путать с незаконными изменениями и разрушениями, являющимися проблемой безопасности).

Современные СУБД имеют ряд средств для обеспечения поддержания целостности.

Выделяют три группы правил целостности:

- ✓ Целостность по сущностям.
- ✓ Целостность по ссылкам.
- ✓ Целостность, определяемая пользователем.

Не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.

Значение внешнего ключа должно быть равным значению первичного ключа цели либо быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.

Для любой конкретной базы данных существует ряд дополнительных специфических правил, которые относятся к ней одной и определяются разработчиком. Чаще всего контролируется:

уникальность тех или иных атрибутов, диапазон значений (экзаменационная оценка от 2 до 5), принадлежность набору значений (пол "М" или "Ж").

Операторы реляционной алгебры

Традиционные операции над множествами

- Объединением (Union) двух отношений называется отношение, содержащее множество кортежей, принадлежащих либо первому, либо второму отношению.

$$R_1 = \{ r_1 \}, \quad R_2 = \{ r_2 \} \quad R_1 \cup R_2 = \{ r \mid r \in R_1 \vee r \in R_2 \}$$

- Пересечением (Intersect) отношений называется отношение, которое содержит множество кортежей, принадлежащих одновременно и первому и второму отношению.

$$R_1 \cap R_2 = \{ r \mid r \in R_1 \wedge r \in R_2 \}$$

- Разностью (Except) отношений называется отношение, содержащее множество кортежей, принадлежащих R_1 и не принадлежащих R_2:

$$R_1 \setminus R_2 = \{ r \mid r \in R_1 \wedge r \notin R_2 \}$$

- Декартовым произведением (Times) отношения степени n со схемой и отношения степени m со схемой, содержащее кортежи, полученные сцеплением каждого кортежа r отношения с каждым кортежем q отношения

$$R_1 = \{ r \}, \quad R_2 = \{ q \} \qquad R_1 \otimes R_2 = \{ (r, q) \mid r \in R_1 \wedge q \in R_2 \}$$

Специальные операции реляционной алгебры

- Операция выбора (Select), заданная на отношении R в виде булевского выражения, определенного на атрибутах отношения R, называется отношение , включающее те кортежи из исходного отношения, для которых истинно условие выбора:

$$R[\alpha(r)] \quad \{r | r \in R \ \wedge \alpha(r) = "Истина" \}$$

- Операция проектирования (Project) или вертикального выбора называется отношение со схемой, соответствующей набору атрибутов B, содержащему кортежи, полученные из кортежей исходного отношения R путем удаления из них значений, не принадлежащих атрибутам из набора B.

- Операция соединения (Join) возвращает отношение, кортежи которого – это сочетание двух кортежей, имеющих общее значение для одного или нескольких общих атрибутов этих двух отношений.

- Операция деления (Divide) возвращает отношение, содержащее все значения одного атрибута отношения, которые соответствуют (в другом атрибуте) всем значениям во втором отношении.

Понятия полной и транзитивной функциональной зависимости

- Функциональная зависимость (functional dependence – FD) – в отношении R атрибут Y функционально зависит от атрибута X в том и только в том случае, если каждому значению X соответствует в точности одно значение Y: R.X -> R.Y

- Полная функциональная зависимость – функциональная зависимость R.X -> R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X (точным подмножеством множества X называется любое его подмножество, не совпадающее с X).

- Транзитивная функциональная зависимость – функциональная зависимость R.X -> R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости
  R.X -> R.Z и R.Z -> R.Y.

Нормализация

Руководство по нормализации – это набор стандартов проектирования данных, называемых нормальными формами (normal form).

Общепринятыми считаются пять нормальных форм (НФ).

Создание таблиц в соответствии с этими стандартами называется *нормализацией*.

Нормальные формы изменяются в порядке от первой до пятой.

Каждая последующая форма удовлетворяет требования предыдущей.

Если вы следуете первому правилу нормализации, ваши данные будут представлены в первой нормальной форме.

Если ваши данные удовлетворяют третьему правилу нормализации, они будут находиться в третьей нормальной форме (а также в первой и второй формах).

Выполнение правил нормализации обычно приводит к разделению таблиц на две или больше таблиц с меньшим числом столбцов, выделению отношений первичный ключ − внешний ключ в меньшие таблицы, которые снова могут быть соединены с помощью операции объединения.

Одним из основных результатов разделения таблиц в соответствии с правилами нормализации является уменьшение избыточности данных в таблицах.

При этом вас не должно смущать наличие в базе одинаковых столбцов первичных и внешних ключей. Такое преднамеренное дублирование – это не то же самое, что избыточность.

Большинство разработчиков баз данных признают, что представление данных в третьей или четвертой нормальных формах полностью удовлетворяет все их потребности.

Нормальные формы (НФ)

I НФ

I НФ требует, чтобы на любом пересечении строки и столбца находилось единственное значение, которое должно быть атомарным. Кроме того, в таблице, удовлетворяющей первой нормальной форме, не должно быть повторяющихся групп.

II НФ

II НФ требует выполнения I НФ и, чтобы любой неключевой атрибут зависел от всего первичного ключа. Следовательно, таблица не должна содержать неключевых атрибутов, зависящих только от части составного первичного ключа.

III НФ

III НФ повышает требования второй нормальной формы. III НФ требует, чтобы ни один неключевой атрибут не зависел от другого неключевого атрибута. Любой неключевой атрибут должен зависеть только от первичного ключа.

IV и V НФ

IV НФ запрещает независимые отношения типа один-ко-многим между ключевыми и неключевыми атрибутами.

V НФ доводит весь процесс нормализации до логического конца, разбивая таблицы на минимально возможные части для устранения в них всей избыточности данных.

Проектирование баз данных

При проектировании базы данных решаются основные проблемы:

- Отображение объектов предметной области в абстрактные объекты модели данных таким образом, чтобы это отображение не противоречило семантике предметной области и было по возможности лучшим (эффективным, удобным и т.д.). Часто эту проблему называют проблемой логического проектирования баз данных.

- Обеспечение эффективного выполнения запросов к базе данных, т.е. рациональное расположение данных во внешней памяти, создание полезных дополнительных структур (например, индексов) с учетом особенностей конкретной СУБД. Эту проблему называют проблемой физического проектирования баз данных.

ER – модель (Entity-Relationship, Сущность-Связи)

Большинство современных подходов к проектированию баз данных основано на использовании разновидностей ER-модели.

Модель была предложена Ченом (Chen) в 1976 г.

Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в системах CASE, поддерживающих автоматизированное проектирование реляционных баз данных.

Основными понятиями ER-модели являются сущность, связь и атрибут.

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Для большей выразительности и лучшего понимания имя сущности может сопровождаться примерами конкретных объектов этого типа.

Пример сущности: сущность Вокзал с объектами Савеловский и Белорусский

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Связь представляется в виде **линии**, связывающей две сущности или ведущей от сущности к ней же самой. При этом в месте "стыковки" связи с сущностью используются трехточечный вход в прямоугольник сущности, если для этой сущности в связи могут использоваться много (many) экземпляров сущности, и одноточечный вход, если в связи может участвовать только один экземпляр сущности.

Обязательный конец связи изображается сплошной **линией**, а необязательный – прерывистой **линией**.

3 типа по множественности

Связи делятся на следующие типы:

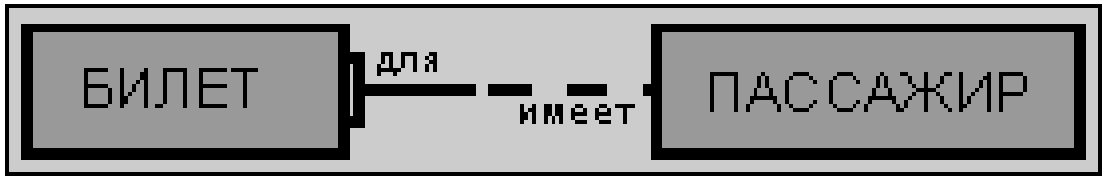
Один-к-одному (1:1) экземпляр одной сущности связан с одним экземпляром другой сущности.

Один-ко-многим (1:M) экземпляр одной сущности связан с несколькими экземплярами другой сущности.

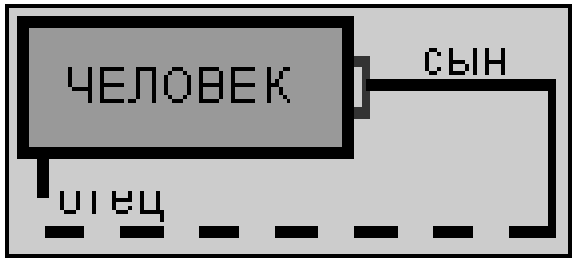
Многие-ко-многим (M:M) экземпляр одной сущности связан с несколькими экземплярами другой сущности и наоборот, любой экземпляр второй сущности связан с несколькими экземплярами первой сущности.

Связь – это типовое понятие, все экземпляры обеих пар связываемых сущностей подчиняются правилам связывания.

В изображенном ниже примере связь между сущностями БИЛЕТ и ПАССАЖИР связывает билеты и пассажиров. При том конец сущности с именем "для" позволяет связывать с одним пассажиром более одного билета, причем каждый билет должен быть связан с каким-либо пассажиром. Конец сущности с именем "имеет" означает, что каждый билет может принадлежать только одному пассажиру, причем пассажир не обязан иметь хотя бы один билет.



Каждый ПАССАЖИР может иметь один или более БИЛЕТОВ.



На следующем примере изображена рекурсивная связь, связывающая сущность ЧЕЛОВЕК с ней же самой.

Конец связи с именем "сын" определяет тот факт, что у одного отца может быть более чем один сын.

Конец связи с именем "отец" означает, что не у каждого человека могут быть сыновья.

Лаконичной устной трактовкой изображенной диаграммы является следующая:

Каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;
Каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ

Язык SQL

- В начале 1970-х годов в компании IBM была разработана экспериментальная реляционная СУБД IBM System/R, для которой был создан специальный язык SEQUEL, позволявший просто управлять данными в этой СУБД.
- Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, не имеющий навыков программирования. Первыми СУБД, поддерживающими новый язык, стали в 1979 году Oracle V2 для машин VAX и System/38 от IBM, основанная на System/R.
- В 1983 году приступили к разработке стандарта языка SQL.
- В 1986 году ANSI представил свою первую версию стандарта, под названием «Database Language SQL» (Язык баз данных SQL). Неофициально этот стандарт SQL-86 получил название SQL1. Со временем к стандарту накопилось несколько замечаний и пожеланий, особенно с точки зрения обеспечения целостности и корректности данных.
- В 1989 году данный стандарт был расширен, получив название SQL89.
- В настоящее время действует стандарт, принятый в 2003 году с небольшими модификациями 2008 года.

SQL*Plus

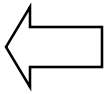
SQL*Plus представляет собой вариант языка SQL, разработанный корпорацией Oracle, где слово "Plus" обозначает предложенные Oracle расширения SQL.

Любые операции с реляционными базами данных осуществляются исключительно с помощью SQL-подобных языков программирования.

SQL* Plus обладает дружественным пользователю интерфейсом.

При программировании на SQL*Plus пользователь имеет дело с наборами данных как с единым целым (другими словами, вам не придется контролировать обработку одной записи вслед за другой).

Реализованные в SQL*Plus расширения SQL значительно облегчают генерацию насыщенных информацией отчетов.



Язык DDL

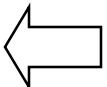
Язык определения данных DDL представляет собой подмножество команд языка SQL, предназначенное для создания и определения объектов базы данных. Определения объектов, созданные с помощью команд DDL, сохраняются в словаре базы данных.

Язык определения данных DDL обеспечивает:

- ✓ Создание объекта базы данных (CREATE).
- ✓ Удаление объекта базы данных (DROP).
- ✓ Изменение свойств объекта базы данных (ALTER).

Необходимо отметить, что при получении SQL-предложения на языке DDL сервер базы данных Oracle фиксирует состояние текущей транзакции как перед выполнением каждого предложения DDL, так и сразу после него. Поэтому, когда после внесения новых записей в базу данных вы выполняете, например, предложение DDL **create table**, то данные, внесенные предыдущими командами **insert**, окажутся зафиксированными в базе данных

Поскольку речь идет о языке определения данных, нетрудно догадаться, что основное назначение этого подмножества SQL ≈ помочь пользователю в определении формата создаваемых объектов БД.



Create table и Drop table

Для создания таблицы базы данных служит команда CREATE TABLE:

CREATE TABLE имя_таблицы (название_атрибута1 тип_данных 1 ограничение1, название_атрибута2 тип_данных 2 ограничение2, …..);

CREATE TABLE Хобби (название_хобби varchar2(25) primary key, тип_хобби varcar 2(20), Риск number(2));

Для удаления таблицы из базы данных служит команда DROP TABLE:

DROP TABLE имя_таблицы [CASCADE CONSTRAINTS];

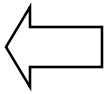
Удаление таблицы уничтожает все данные в ней и используемые ею индексы.

Опция CASCADE CONSTRAINTS удаляет все внешние ссылки на столбцы таблицы. Без этой опции, при наличии внешних ссылок, таблица не будет удалена.

Типы данных ORACLE

Тип данных определяет, к какому классу относится каждый отдельный элемент информации.

CHAR (размер)	**символьные строки фиксированной длины**
VARCHAR2 (размер)	**символьные строки переменной длины**
MBER (точность, масштаб)	**числа всех типов (целые, с плавающей точкой)**
DATE	**информация о времени (дата, время)**
LONG	**большие строки переменной длины (до 2Гб)**
ROW (размер)	**небольшие двоичные строки**
LONG ROW	**большие двоичные строки (до 2Гб)**



Декларативные ограничения

Можно задавать следующие типы декларативных ограничений:

- ✓ NULL / NOT NULL
- ✓ UNIQUE
- ✓ PRIMARY KEY
- ✓ FOREIGN KEY
- ✓ CHECK

Ограничение NOT NULL означает, что указанный столбец не может содержать NULL-значений.

Ограничение NULL явно указывает , что столбец может иметь NULL-значения.

Ограничение UNIQUE устанавливает правило, согласно которому значения столбца или комбинации столбцов данной таблицы должны быть уникальными.

Ограничение PRIMARY KEY объявляет столбец или комбинацию столбцов первичным ключом.

Ограничение foreign key обеспечивает поддержание целостности данных на уровне взаимосвязей различных таблиц. Или, как говорят, обеспечивает сылочную целостность (referential integrity).

Внешний ключ всегда ссылается на уникальный или первичный ключ.

Ограничение CHECK явно задает условие, которому должны удовлетворять значения столбцов в каждой строке таблицы.

Alter Table

Для изменения структуры таблицы используется команда ALTER TABLE.

ALTER TABLE table_name
[ADD
[(]column1 datatype1 [DEFAULT expr1] [col_constraint1]
[,column2 datatype2 [DEFAULT expr2] [col_constraint2]]
...[)]
]
[MODIFY
[(]column1 datatype1 [DEFAULT expr1] [col_constraint1]
[,column2 datatype2 [DEFAULT expr2] [col_constraint2]]
...[)]
]
[DROP drop_clause];

ADD − добавление нового столбца или ограничения.
MODIFY − изменение описания существующего столбца.
DROP − удаление заданного ранее *ограничения* целостности.

ПРИМЕРЫ

ALTER TABLE Студенты ADD (CONSTRAINT Check_name CHECK (Ср_балл between 3 and 5))
ALTER TABLE Студенты MODIFY (Фамилия VARCHAR2(55))
ALTER TABLE emp DROP CONTRAINT emp_self_key;

Условия создания ограничений с помощью команды ALTER TABLE

Тип ограничения	Добавление к существующим столбцам таблицы	Добавление с новыми столбцами таблицы
NOT NULL	Не может быть определено, если существует строка с пустым значением в этом столбце*	Не может быть определено, если в таблице есть строки
UNIQUE	Не может быть определено, если в ключе существуют повторяющиеся значения*	Всегда допускается
PRIMARY KEY	Не может быть определено, если в ключе существуют повторяющиеся или пустые значения*	Не может быть определено, если в таблице есть строки
FOREIGN KEY	Не может быть определено, если внешний ключ имеет значения, не ссылающиеся на родительский ключ*	Всегда допускается
CHECK	Не может быть определено, если столбец имеет значения, нарушающие условие ограничения*	Всегда допускается

Язык DML

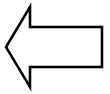
Язык обработки данных DML позволяет:

- ✓ добавлять (**insert**) строки в БД,
- ✓ изменять (**update**) данные в БД,
- ✓ удалять (**delete**) строки из БД,
- ✓ выбирать (**select**) информацию из БД.

В полном соответствии со своим названием, язык DML предназначен для работы с информационным содержанием базы данных.

Операторы языка DML представляют собой SQL-команды, позволяющие изменять и дополнять хранящуюся в базе данных информацию

```
insert into {ИМЯ_ТАБЛИЦЫ [(СТОЛБЕЦ, СТОЛБЕЦ, . . . )] | (подзапрос) }
        values (значение, значение, . . . );
update {ТАБЛИЦА | (подзапрос)} [ПСЕВДОНИМ]
       set СТОЛБЕЦ [,СТОЛБЕЦ . . . ]={ВЫРАЖЕНИЕ | подзапрос}
    [where условие];
delete  from  {таблица | (подзапрос)}
    [where условие];
```



Структура оператора SELECT

Команда select состоит из следующих предложений:

```
select [distinct] {*,СТОЛБЕЦ  [[as ]ПСЕВДОНИМ],  ...  }
    from {ТАБЛИЦА | (подзапрос)}
    [where критерий поиска для строк]
    [group by   группировка]
    [having  критерий поиска для групп]
    [order by {СТОЛБЕЦ [asc|desc],  ...   }]
    [OFFSET offset ROWS]
    [FETCH NEXT [ row_count | percent PERCENT ] ROWS [
                                                ONLY | WITH TIES ]
```

Команда select производит выбор данных из БД с использованием всех операторов реляционной алгебры.

Простейший вид: Select * from имя_таблицы;

Предотвращение выборки повторяющихся строк (distinct)

Чтобы исключить одинаковые строки из результирующей таблицы задается квалификатор distinct

Определение условий выборки

Для выполнения сравнения используются следующие операторы:

=, <>, <, >, <=, >=

Is null или Is not null

Between 100 and 300

In(‘Москва’,’ Дубна’, ‘Дмитров’) или not in(‘Москва’,’ Дубна’, ‘Дмитров’)

Like ‘A%’

Функции преобразования

Эти функции предназначены для преобразования значения одного типа в другой.

Формат

TO_CHAR*(число|дата[,’формат’])* преобразует числа и даты в символьные строки.

TO_NUMBER*(строка)* преобразует символьные строки, содержащие цифры, в числа.

TO_DATE*(‘строка’[,’формат’])* преобразует символьные строки, в даты на основании шаблона, заданного в формате.

Групповые функции

Групповые функции оперируют с набором строк. Они формируют результаты основанные на обработке группы строк.

По умолчанию все строки таблицы результатов трактуются как одна группа.

Чтобы разбить строки на группы, используется предложение group by, задаваемое в команде select.

AVG ([distinct | ALL | n]) возвращает среднее значение группы полей в столбце, не включая пустые значения.

COUNT([distinct|ALL|выражение]) возвращает количество строк в группе, в которых выражение имеет не пустое значение.

COUNT(*) возвращает общее количество строк в группе.

MAX([distinct | ALL | выражение]) возвращает максимальное значение выражения в группе.

MIN([distinct | ALL | выражение]) возвращает минимальное значение выражения в группе.

SUM([distinct | ALL | n]) возвращает сумму значений по группе, игнорируя пустые значения.

Подзапросы

Подзапрос - это команда select, помещенная в другую команду select для получения

промежуточных результатов.

Правила задания подзапросов

1. Внутренний подзапрос берется в круглые скобки и должен стоять в правой части оператора сравнения внешнего запроса.
2. *Подзапрос не может содержать предложения order by.*
3. Предложение order by ставится последним в основном запросе.
4. *Имена столбцов в предложении select внутреннего запроса должны стоять в той же последовательности, что и имена столбцов в левой части оператора сравнения внешнего запроса. Типы столбцов также должны попарно соответствовать.*
5. Подзапросы всегда выполняются от внутренних к внешним, если они не коррелируют с друг другом.
6. *При задании критериев поиска могут использоваться логические операторы, SQL - операторы, а также операторы ANY и ALL.*

Подзапросы

Подзапросы могут:

- ✓ возвращать одну или более строк; возвращать один или более столбцов;
- ✓ использовать группы или групповые функции;
- ✓ задаваться в сложных критериях поиска внешних запросов с использованием предикатов and и or;
- ✓ соединять таблицы; обращаться к таблице, отличной от той, к которой обращается внешний запрос;
- ✓ стоять в командах select, UPDATE, DELETE, INSERT, CREATE TABLE;
- ✓ коррелировать с внешним запросом;

Операторы ANY и ALL

Операторы ANY и ALL могут применяться в подзапросах, возвращающих более одной строки. Они задаются в предложениях where или having вместе с логическими операторами (=, <>, >, <, >=, <=).

ANY (синоним SOME) сравнивает значение левой части оператора сравнения с каждым значением, возвращаемом подзапросом. Результат сравнения положителен, если хотя бы одно значение из найденных по подзапросу удовлетворяет условию сравнения.

Пример. Получить список студентов, имеющих средний балл больше минимального балла в заданной группе.

```
select * from students
where Avg_ball > ANY (select distinct Avg_ball from students  where N_gr='БИ-10')  order by Avg_ball;
```

Оператор ALL сравнивает значение со всеми значениями, возвращаемые подзапросом. Результат сравнения положителен, если все найденные по подзапросу значения удовлетворяет условию сравнения.

Пример. Получить список студентов, имеющих средний балл больше чем любой студент в заданной группе .

```
select * from students
where Avg_ball > ALL (select distinct Avg_ball from students  where N_gr='БИ-10')  order by Avg_ball;
```

Коррелированные подзапросы

Коррелированные подзапросы - это вложенные подзапросы, выполняющиеся для каждой “строки-кандидата” из главного запроса, и для выполнения которых требуется информация из строк главного запроса.

Последовательность выполнения коррелированного подзапроса:

- внешним запросом выбирается “строка-кандидат”;
- выполняется внутренний запрос, используя полученное значение из “строки-кандидата”;
- результат выполнения внутреннего запроса возвращается во внешний запрос для проверки соответствия критерию “строки-кандидата”;
- процедура повторяется для всех строк из внешнего запроса.

Пример

Получить список студентов, имеющих балл больше среднего по группе, в которой они учатся

```
select * from students s
where Avg_ball > (select AVG(Avg_ball) from students
                        where N_gr =s. N_gr)
order by N_gr;
```

Операторы над множествами

✓ *Оператор UNION* ***(операция объединения)***

Используется для получения всех неодинаковых строк из результирующих таблиц:

```
  select N_z from students  where N_gr =‘БИ-10’
UNION
  select N_z from students  where N_gr =‘ПИ-10’
```

✓ *Оператор INTERSECT* ***(операция пересечения)***

*И*спользуется для получения всех одинаковых строк из результирующих таблиц:

```
  select N_z from students
INTERSECT
  select N_z from st_hobby ;
```

✓ *Оператор MINUS* ***(операция разности)***

Используется для получения всех строк, найденных по одному запросу, но не найденных по другому запросу:

```
   select N_z from students
MINUS
  select N_z from st_hobby;
```

Общие правила использования операторов над множествами

- ✓ Команды select должны выбирать одинаковое количество столбцов.
- ✓ Соответствующие столбцы должны иметь одинаковый тип.
- ✓ Предикат distinct не допускается.
- ✓ В результирующей таблице столбцы именуются по первой SQL команде.
- ✓ Предложение order by может стоять только в конце запроса и относится к окончательной таблице результатов.
- ✓ Ссылка на столбцы в предложении order by - только по порядковым номерам.
- ✓ Операторы над множествами могут задаваться в подзапросах.
- ✓ Команды select выполняются сверху вниз.

Представления (особый взгляд на данные)

Представление – объект базы данных, позволяющий получить определенную пользователем выборку данных из одной или нескольких таблиц.

В отличие от таблицы, представление не содержит никаких данных, а лишь запрос на языке SQL, дающий возможность прочитать из базы данных необходимую информацию и представить ее в табличной форме.

На самом деле пользователям, не принимавшим участия в создании данного представления, вряд ли придет в голову, что они имеют дело с представлением, а не с настоящей таблицей.

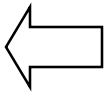
К представлениям можно применять операторы insert, update, delete и select.

Использование представлений

Представления обеспечивают дополнительный уровень безопасности базы данных. Например, можно создать общую таблицу со сведениями обо всех сотрудниках компании, но разрешить менеджерам компании получать информацию только об их подчиненных.

Представления позволяют скрыть от пользователей сложность структуры хранимых данных.

Представления позволяют использовать разумные названия отдельных столбцов.



Представления (особый взгляд на данные)

Если к представлению можно применить операторы обновления (INSERT, UPDATE или DELETE), то представление является *обновляемым* (updateble), иначе оно является *читаемым* (read-only).

Ниже приведены критерии того, является ли представление обновляемым в SQL:

- ✓ оно базируется на одной таблице;
- ✓ оно должно включать первичный ключ таблицы;
- ✓ оно не должно включать полей, полученных в результате применения функций агрегирования;
- ✓ оно не может содержать спецификации DISTINCT;
- ✓ оно не должно использовать GROUP BY или HAVING;
- ✓ оно не должно использовать подзапросы;
- ✓ оно может быть определено на другом представлении, но это представление должно быть обновляемым;
- ✓ оно не может содержать константы, строки или выражения в списке выбираемых выходных полей;
- ✓ для INSERT оно должно включать поля из таблицы, которые имеют ограничения NOT NULL.

Представления

- Для обновляемых представлений применимы любые операторы обновления, которые при выполнении по отношению к представлению транслируются на исходную таблицу, на основе которой было создано представление. Однако это правило на первый взгляд не согласуется с функцией представлений по защите данных, описанной ранее. Действительно, если представить, что при помощи представления были скрыты столбцы или строки, но пользователь знает об их существовании, он может выполнить операции обновления данных, затрагивающие скрытую в представлении информацию.

- Для предотвращения описанной ситуации применяется параметр WITH CHECK OPTION, который может указываться при создании обновляемых представлений после оператора SELECT. При наличии этого параметра СУБД предотвращает операции обновления данных, если они нарушают условия горизонтальной или вертикальной фильтрации, определенные в предложении в операторе SELECT

Язык DCL

Чтобы выполнить над таблицей Oracle некоторую операцию, например INSERT или DELETE, необходимо иметь соответствующие полномочия, которые предоставляются и отменяются с помощью SQL-команд GRANT и REVOKE.

Оператор GRANT используется для открытия другой схеме доступа к привилегии.

Оператор REVOKE − для запрещения доступа, разрешенного оператором GRANT.

Оба оператора могут использоваться как для объектных, так и для системных привилегий.

Объектные и системные привилегии

Существуют привилегии двух различных видов: объектные и системные.

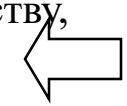
Объектная привилегия (object privilege) разрешает выполнение определенной операции над конкретным объектом (например, над таблицей).

Системная привилегия (system privilege) разрешает выполнение операций над целым классом объектов.

Объектные привилегии DDL - ALTER (изменить), INDEX (индексировать), REFERENCES (ссылки) ≈ нельзя применить непосредственно в PL/SQL (за исключением модуля DBMS_SQL), так как они разрешают выполнение операций DDL над определенным объектом.

Роли

В больших системах ORACLE, в которых работает множество пользователей, управление привилегиями − достаточно сложная задача. Для ее упрощения можно использовать функциональное средство ORACLE, называемое ролями. Роль (role) является, по существу, совокупностью привилегии, как объектных, так и системных.



Системные привилегии, привилегии на объекты, роли

Системная привилегия CREATE ANY TABLE дает возможность создавать таблицы в других схемах.

GRANT

Для объектных привилегий синтаксис оператора GRANT таков:

GRANT *привилегия* ON *объект* TO *обладатель_привилегий* [WITH GRANT OPTION];

где *привилегия* - это нужная привилегия, *объект* - это объект, к которому разрешается доступ, а *обладатель_привилегий* - пользователь, получающий привилегию.

Для системных привилегий синтаксис оператора GRANT таков:

GRANT *привилегия* TO *обладатель_привилегий* [WITH ADMIN OPTION];

где *привилегия* - это предоставляемая системная привилегия, а *обладатель_привилегий* - пользователь, получающий привилегию.

REVOKE

Для объектных привилегий синтаксис оператора REVOKE таков:

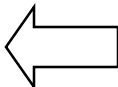
REVOKE *привилегия* ON *объект* FROM *обладатель_привилегий* [CASCADE CONSTRAINTS];

где *привилегия* - это отменяемая привилегия, *объект* - это объект, на который предоставлена привилегия, а *обладатель_привилегий* - пользователь, получающий эту привилегию.

Для системных привилегий синтаксис оператора REVOKE таков:

REVOKE *привилегия* FROM *обладатель_привилегий;*

где *привилегия* - это отменяемая системная привилегия, а *обладатель_привилегий* - пользователь, который более не будет ее иметь.



Последовательности

В реляционных или объектно-реляционных базах данных часто возникает необходимость в генерации целочисленных столбцов, где каждая строка имеет свое уникальное значение, как правило, используемое в качестве первичного ключа. Генераторы последовательностей позволяют избежать ненужного дискового ввода-вывода за счет помещения генерируемых чисел в специальный кэш-буфер, находящийся в оперативной памяти сервера.

Отметим, что в отличие от таблиц или представлений данных, приложения не могут осуществлять поиск определенных чисел непосредственно в сгенерированной последовательности.

Последовательность создается с помощью команды DDL CREATE SEQUENCE имя_последовательности.

После того, как последовательность создана, к ней можно обратиться:
последовательность.CURRVAL
или
последовательность.NEXTVAL,
где *последовательность* – это имя последовательности, CURRVAL (текущее значение) и NEXTVAL (следующее значение) – это псевдостолбцы, применяющиеся в последовательностях.

CURRVAL возвращает текущее значение последовательности, а NEXTVAL увеличивает ее и возвращает новое значение. Как CURRVAL, так и NEXTVAL возвращают значения, имеющие тип NUMBER.

Значения последовательностей могут быть использованы в списках выбора запросов, в списках значений (VALUES) операторов INSERT и в командах SET операторов UPDATE. Однако их нельзя указывать в условиях WHERE и в процедурных операторах PL/SQL.

Создание последовательности

CREATE SEQUENCE My_sequence
[INCREMENT BY n]
[START WITH n]
[MAXVALUE n | NOMAXVALUE]
[MINVALUE n | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE n | NOCACHE]
[ORDER | NOORDER];

INCREMENT BY n - определяет величину приращения (шаг последовательности). Может быть любым положительным или отрицательным целым числом, содержащим до 28 цифр.По умолчанию приращение равно 1.

START WITH n - определяет начальное значение последовательности - целое число длиной до 28 цифр

MAXVALUE n - максимальное значение, которое может быть сгенерировано последовательностью - целое число длиной до 28 цифр. Должно быть меньше или равно значению, заданному в START WITH. По умолчанию NOMAXVALUE, что эквивалентно 10^{27} для возрастающей последовательности и -1 для убывающей.

MINVALUE n - минимальное значение, которое может быть сгенерировано последовательностью.

CYCLE - указывает, что последовательность является циклической, то есть продолжает генерировать значения и после достижения заданных ограничений. По умолчанию принимается NOCYCLE.

CACH n - задает сколько элементов последовательности для ускорения доступа Oracle должен хранить в оперативной памяти (кешировать). n - целое число длиной до 28 цифр. Минимальное значение равно 2. Максимальное значение определяется формулой (CEIL (MAXVALUE–MINVALUE)) / ABS(INCREMENT). По умолчанию принимается CASH 20.

ORDER - предписывает выдавать значения последовательности строго в соответствии с временем их запроса. Имеет смысл только при использовании с опцией Parallel Server в параллельном режиме, так как при работе без использования данной опции (exclusive mode) значения всегда генерируются строго последовательно. По умолчанию принимается NOORDER.

Два пользователя, обращаясь к одной и той же последовательности никогда не получат одинаковых номеров, но