**Информационнокоммуникационные системы**

Преподаватель: Людмила Игоревна Мстиславская

Оглавление

[Структура удаленных сетей 2](#_Toc2598243)

[Архитектура «Файл сервер» 2](#_Toc2598244)

[Архитектура «Клиент-сервер» 2](#_Toc2598245)

[Многоуровневая архитектура 4](#_Toc2598246)

[Язык SQL 6](#_Toc2598247)

[Чтение данных 7](#_Toc2598248)

[Сравнение 9](#_Toc2598249)

[Модификация данных 10](#_Toc2598250)

[Модификация метаданных 10](#_Toc2598251)

[Управление правами доступа средствами языка SQL 10](#_Toc2598252)

[Управляющие серверы 14](#_Toc2598253)

[INTERBASE. Структура 14](#_Toc2598254)

[Понятие доменов 17](#_Toc2598255)

[Правила создания таблиц 18](#_Toc2598256)

[Суррогатные ключи 18](#_Toc2598257)

[«Деревянные» списки 19](#_Toc2598258)

Структура удаленных сетей

Архитектура «Файл сервер»

Файл-сервер – это выделенный сервер, оптимизированный для выполнения файловых операций ввода-вывода. Предназначен для хранения файлов любого типа. Как правило, обладает большим объёмом дискового пространства.

Файл-серверные приложения схожи по своей структуре с локальными приложениями и используют сетевой ресурс для хранения программы и данных. К функциям сервера относятся хранения данных и кода программы. Обработка данных происходит исключительно на стороне клиента. Количество клиентов ограничено десятками. Отметим **плюсы** этой архитектуры:

1. Низкая стоимость разработки.
2. Невысокая стоимость обновления и изменения ПО.

**Минусы:**

1. Низкая производительность, которая зависит от производительности сети, клиента, сервера.
2. Плохая возможность подключения новых клиентов.
3. В целом, ненадежная система.

Сетевое многопользовательское приложение, построенное по типу «Файл-сервер», работает следующим образом. Данные в виде одного или нескольких файлов размещаются на файловом сервере, который принимает запросы, поступающие по сети от компьютеров клиента, и передаёт им требуемые данные. На каждом из компьютеров запускается полная копия процессора обработки данных Jet Engine.

Любая копия процессора независимо управляет данными или файлами БД и связь между этими независимыми действиями – это файл блокировок. Файл блокировки обязательно создаётся для каждого файла БД. При этом каждая копия процессора Jet выполняет изменение индексов, работу с системными таблицами и другие функцию, входящие в компетенцию СУБД.

Архитектура «Клиент-сервер»

В архитектуре «Клиент-сервер» сервер БД берёт на себя не только обеспечение доступа к данным, но и обработку этих данных. Клиент посылает на сервер запросы на чтение или изменение данных, сформулированных на языке SQL.

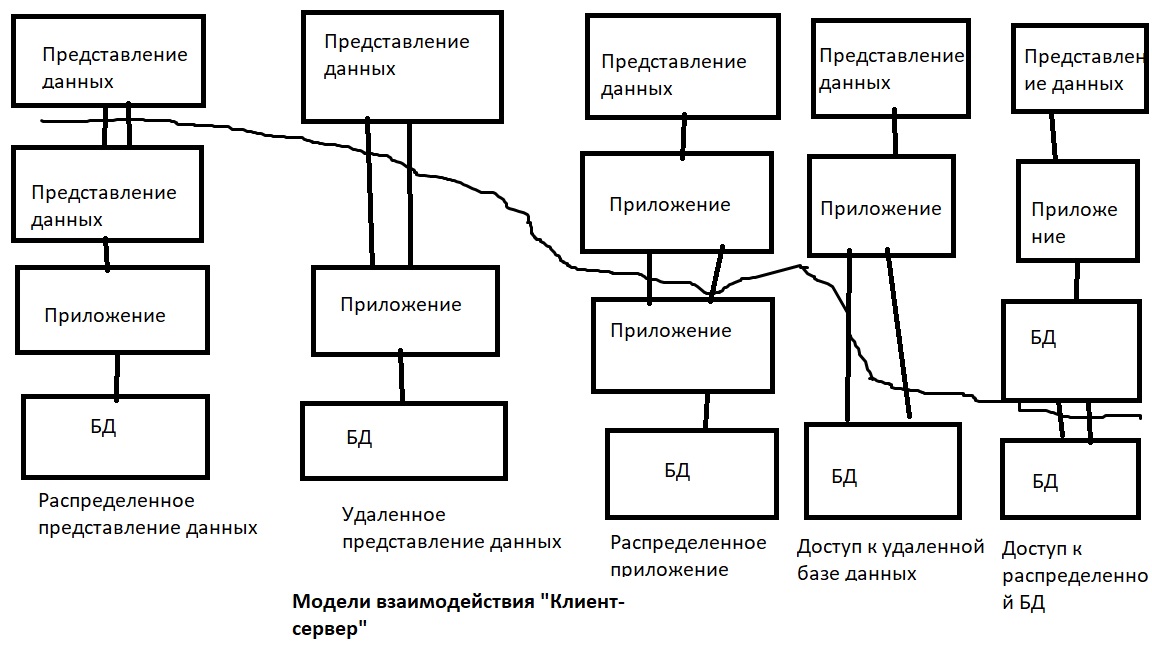
Архитектура «Клиент-сервер» устраняет недостатки «Файл-сервер». Принцип работы заключается в следующем. Функции подразделяются на 3 группы:

1. Ввод и отображение данных (взаимодействие с пользователем).
2. Прикладные функции, характерные для конкретной предметной области.
3. Функция управления ресурсами (файловые системы, БД и т.д.)

Поэтому в любом приложении выделяются следующие компоненты:

1. Компонент представления данных.
2. Прикладной компонент.
3. Компонент управления ресурсом.

Связь между этими компонентами осуществляется по определенным правилам, которые называются протоколом взаимодействия. Со временем появилось несколько моделей взаимодействия структур «Клиент-сервер».



*(волнистой линии разделены два узла: клиентской узел (и все компоненты на нём) и серверный узел (и все компоненты на нём). Двойные линии обозначает двухстороннюю связь сервера и клиента)*

Исторически первой появилось модель распределенного распределения данных, которая реализовывалась на универсальной ЭВМ с подключенными к ней не-интеллектуальными терминалами. Управление данными и взаимодействие с пользователем объединялись в одной программе. На терминал передавалась только «картинка», сформированная на центральном компьютере.

Вторая модель – удаленное представление данных было реализовано с появлением ПК и локальных сетей. При этом некоторое время ещё сохранялась «Файл-сервер» архитектура. Протокол обмена при этом представлял набор низкоуровневых вызовов операций файловой системы.

С появлением первых специализированных серверов БД стало возможным создать архитектуру «Клиент-сервер». Однако сетевой трафик оставался по-прежнему высоким, кроме того сохранялось удовлетворительное администрирование приложений, поскольку в одной программе совмещены различные функции.

Позже была разработана концепция «активного сервера», который использовала механизм хранимых процедур. Это позволило часть прикладного компонента перенести на сервер. В этом случае процедуры хранятся в словаре БД, распределяются между несколькими клиентами и выполняются на том же компьютере, что и SQL-сервер.

**Достоинства** такого подхода:

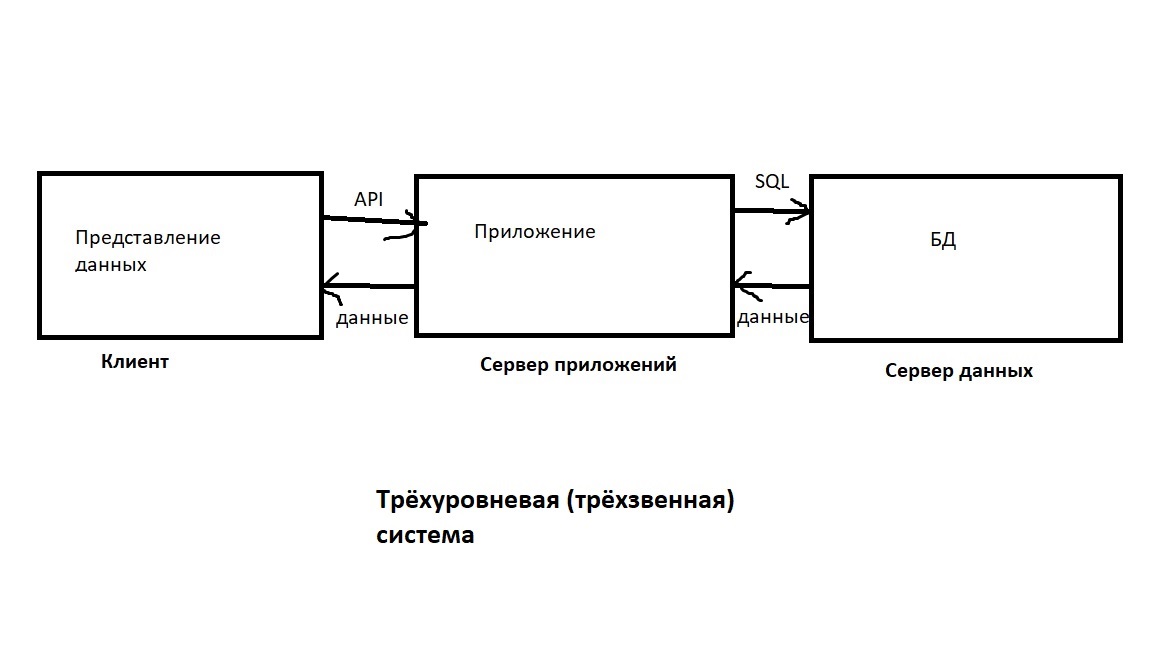
1. Снижение сетевого трафика.
2. Централизованное администрирование прикладных функций.

**Недостатки** также остаются:

1. Ограниченность средств разработки хранимых процедур по сравнению с языками общего назначения.

Многоуровневая архитектура

Принцип многоуровневой архитектуры заключается в реализации двух основных принципов:

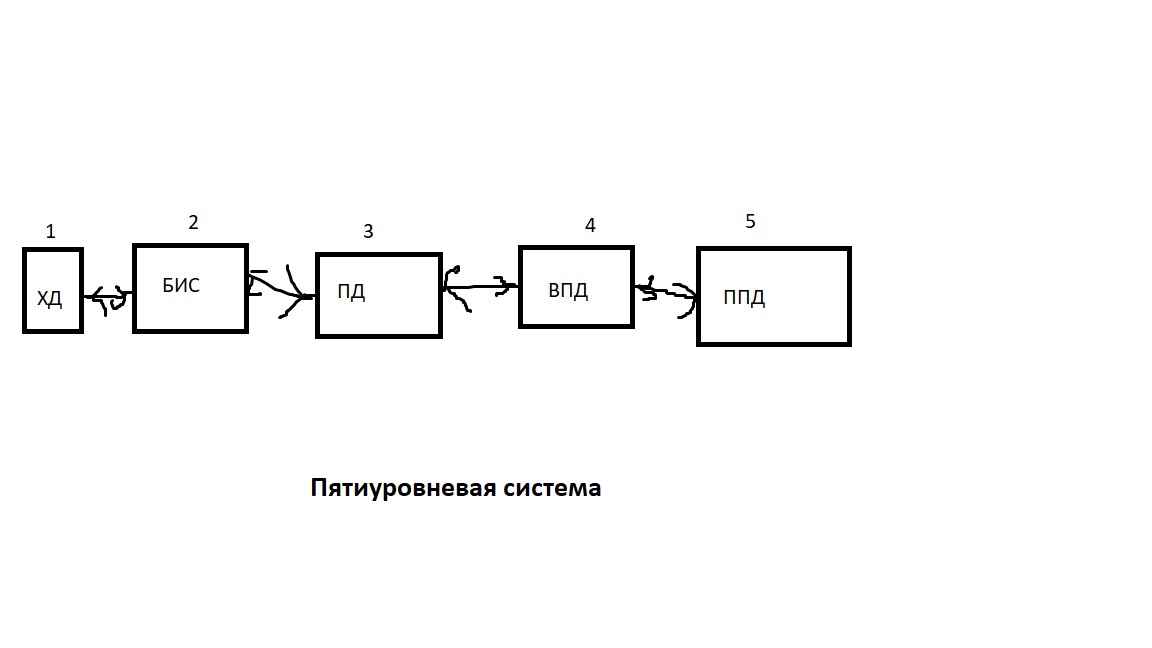


1. Минимизация функциональности клиентских компонентов, оставляющие за клиентом только функции пользовательского интерфейса.
2. Освобождение сервера БД от несвойственных ему функций.

На практике эти принципы выполняются введением третьего звена.

**Достоинства:**

1. Снижается трафик.
2. Дополнительная защита.
3. Более разумное администрирование и использование функциональных блоков.



ХД – хранилище данных, что означает набор зарегистрированных БД, структура которых задана в системе регистрации данных.

БИС – базовые информационные структуры, объединение которых составляет содержание коллекции.

ПД – провайдер данных – приложение, обеспечивающие обработку унифицированных именованных запросов коллекции и формирование внутреннего представления документов (ВПД).

Подразумевается, что в роле пользователя выступает не человек, а пользовательский агент. В случае HTML-документов это может быть браузер.

ВПД (внутреннее представление документов) – формирует унифицированные именованные запросы коллекции и осуществляет отбор информации.

На 5-ом этапе формируется презентационное представление документа (визуализация информации в формате, понятному человеку).

Данная архитектура обеспечивает взаимодействие следующих служб:

1. Публикация данных, поддержка их аутентичности и качества.
2. Поиск и предоставление информации.
3. Анализ распределенных данных.
4. Поддержка диспетчеризации.
5. Поддержка системной безопасности и контроль доступа.
6. Поддержка использования данных в удаленных архивах.

Все эти механизмы составляют основу системы усвоения данных, т.е. систему превращения информации в систему библиотек.

Язык SQL

Язык SQL является универсальным компьютерным языком, применяемым для создания модификаций управления данными в реляционных базах данных.

1970 год – в компании IBM разработана экспериментальная СУБД System R. Для неё был создан специальный язык SEQUEL.

Целью этой разработки стало создания простого не процедурного языка, который мог бы воспользоваться любой пользователь, даже не имеющий навыков программирования. К началу 1980 году существовало несколько СУБД от разных производителей, при чём каждый из них обладал собственной реализацией языка запросов. Важным вопросом стало совместимость данных. Поэтому в 1983 международная организация по стандартизации, а также американский национальный институт стандартов приступили к разработке стандартов языка SQL. В настоящее время действует стандарт, принятый в 2003 году с небольшими модификациями, внесенными позже.

Функции языка SQL:

1. Организация данных – создание и изменение структуры БД;
2. Чтение данных;
3. Обработка данных з удаление, добавление и корректировка данных;
4. Управление доступа к данным – предоставление привилегий (ограничение возможностей) пользователю для чтения и изменения данных;
5. Совместное использование данных – координация общего пользования данных многими пользователями;
6. Целостность данных – защита данных от разрушения при сбое системы или других обстоятельствах.

Сервер БД выполняет SQL – запрос и возвращает пользователю только ту информацию из БД, которая соответствует этому SQL-запросу. Каждое SQL-предложение это либо запрос из БД, либо обращение к БД, которое приводит к изменению данных в БД.

Запросы SQL делятся на следующие типы:

1. Запросы на создание или изменения новых или существующих объектов в БД;
2. Запросы на получение данных;
3. Запросы на добавление новых записей;
4. Запросы на удаление данных;
5. Перекрестные запросы;
6. Агрегатные запросы.

Основные элементы SQL:

1. ключевые слова;
2. имена;
3. константы;
4. типы данных;
5. встроенные функции;
6. выражения.

Ключевые слова – это фиксированный набор английских слов, которые определяют тип запроса и необходимую информацию для выполнения этого запроса.

Имена используются для обозначения таблиц, столбцов в таблице, а также владельцев таблиц.

Константы служат для явного указания величин – чисел, строк, дату и время – в командах SQL.

Типы данных служат для представления информации в БД. Их можно разделить на следующие группы:

1. строковые (фиксированной и переменной длины);
2. числовые (целочисленные, с фиксированной и плавающей точкой, двоичные строки);
3. представления даты и времени;

В SQL определен набор типов данных (character (n) или char (n), varchar (n), integer, smallint, decimal (n,m), float (n), doule precision, date, time, timestamp, interval)

Выражения в SLQ представляют собой имена, константы, встроенные функции, связанные между собой знаками арифметических операций. В сложных выражениях для изменения порядка вычислений применяются круглые скобки.

Встроенные функции предназначены для преобразования типов данных и для обработки строк.

Current\_date() – возвращает текущую дату;

Current\_time(точность) – возвращает текущее время;

Char\_length (строка) – возвращает длину строки;

Extract – возвращает значение части day, hour и т.д. даты;

Lower(строка) – возвращает строку, преобразованного к нижнему регистру;

Upper (строка) – то же самое, что и Lower, но наоборот;

Month (дата) – возвращает значения месяца из даты в виде целого числа;

Year (дата) – возвращает значение года из указанной даты в виде целого числа.

Чтение данных

Оператор Select читает данные из БД и возвращает их в виде таблицы результата запроса.

Предложение Select, с которого начинается оператор Select, содержит элементы данных, которые будут возвращены в виде результирующей таблицы. Указанные в предложении Select элементы данных будут составлять столбцы возвращаемой таблицы. В качестве возвращаемых столбцов могут быть указаны:

1. имя столбца таблицы БД:
2. константа, которая будет содержаться в соответствующем столбце возвращаемой таблицы;
3. выражение, которое будет вычисляться для каждой строки возвращаемой таблицы, и помещаться в соответствующем столбце этой таблицы.

Select\*

SELECT CompanyName, ContactName, ContactTitle.

Предложение FRM начинается с ключевого слова FROM, за которым следует в простом случае список спецификаций таблиц, разделенных запятыми. В общем случае за ключевым словом FROM указывается операция соединения (JOIN) исходных таблиц.

Спецификатор таблицы определяет таблицу, из которой запрашиваются исходные данные для формирования возвращаемой таблицы. Спецификатор таблицы представляет собой либо имя исходной таблицы, либо имя исходной таблицы вместе с псевдонимом, указываемым после имени через пробел.

Псевдоним может быть использован в следующих предложениях оператора SELECT вместо имени

SELCT Sname, City

FROM salespeaople;

SELECT Sname, salespeople. City, Customers.

From Salespeaople, Customers;

Операция соединения определена в стандарте SQL2.

Операция CROSS JOIN дает декартово произведение двух таблиц, UNION JOIN дает сложение двух таблиц. Ключевое слово INNER задает внутреннее объединение (по умолчанию), OUTER – внешнее объединение (левое, правое и полное); NATURAL – естественное объединение (равенство значений в одноименных столбцах, при этом предложения ON и USING не обязательны).

Предложение WHERE осуществляет отбор нужных строк из таблицы, получаемой в предложении FROM. Отбор строк производится в соответствии с условием поиска, которое указывается в предложении за ключевым словом WHERE. Условие поиска представляет собой выражение, которое вычисляется для каждой строки возвращаемой таблицы. При вычислении выражения данные соответствующей строки берутся из столбцов, указанных в выражении. Для каждой проверяемой строки условие поиска может иметь одно из трёх значений – TRUE, FALSE и NULL.

В SQL существует пять простых условий поиска:

1. сравнение;
2. проверка на принадлежность диапазону;
3. проверка на принадлежность множеству;
4. поверка на соответствие шаблону;
5. проверка на равенство значению NULL.

Сравнение

Операторы сравнение: <>,=,<,>,=>,=<.

Проверка на принадлежность диапазону эквивалента следующему логическому выражению: (A>=B) AND (A<=C), где A – проверяемое выражение, а B и С – границы диапазона.

WHERE применяется совместно с операторами:

1. ALL, ANY при сравнении со списком значений;
2. BETWEEN при проверке нахождения значения внутри заданного интервала, включая его границы;
3. IN для проверки наличия значений в списке;
4. LIKE при проверке соответствия значения заданной маске.

А также с операторами AND, OR, NOT.

SELECT Company\_Name, Customers, ContactName

FROM Customers

WHERE CompanyName BETWEEN “A” AND “D”

WHERE может применяться совместно с предложением ORDER BY для сортировки результирующего набора данных по одной или нескольким колонкам. По умолчанию данные сортируются по возрастанию. Для сортировки по возрастанию используется ключевое слово ASC, по убыванию – DESC.

WHERE может применяться совместно с предложением GROUP BY для вычисления суммарных значений на основе данных одной или нескольких таблиц.

Предложение HAVING

Для того, чтобы из групп строк, получаемых после предложения GROUP BY, выбрать требуемые группы, используется предложение HAVING. В этом предложении за ключевым словом HAVING следует условие поиска групп, аналогичное условию писка в предложении WHERE.

Условию поиска применяется к группам строк, получаемых после предложения GROUP BY и потому на элементы, входящие в условие поиска, налагаются те же ограничения, какие были перечислены для возвращаемых столбцов при рассмотрений предложения GROUP BY. Таким образом, в условие поиска могут входить константы, агрегатные функции, столбцы группировки и выражение их нам.

Предложение HAVIG в основном используется вместе с предложением GROUP BY. Но язык SQL допускает и отдельное применение предложения HAVING. В этом случае результат запроса рассматривается как одна группа и предложение HAVING выполняет те же функции, что и предложение Where.

В SQL предусмотрены специальные функции, которые называются агрегатными (статистическими) функциями. В SQL имеются следующие агрегатные функции:

SUM () – вычисляет сумму значений, содержащихся в аргументе.

AVG () – вычисляет среднее значение.

MUN (), MAX () – находит минимальное, максимальное значения.

COUNT () – подсчитывает в столбце количество не NULL-значений.

COUNT (\*) – подсчитывает количество строк в запрашиваемой таблице.

Выражение должно содержать имена столбцов и определять столбец, элементы которого вычисляются с помощью соответствующих элементов этих столбцов.

Так как агрегатная функция возвращает одно значение, то она не может быть аргументом другой агрегатной функции.

С ними используются следующие ключевые слова.

DISTINCT указывает, что строки результирующего набора данных должны быть уникальны.

ALL указывает , что возвращать следует все строки.

TOP используется для возврата первых n строк таблицы.

Модификация данных

Оператор UPDATE применяется для изменения значений в одной или нескольких колонках таблицы.

Например, для повышения цен всех продуктов, стоящих меньше 10 долл., можно выполнить следующий запрос:

UPDATE Products SET UnitPrice = UnitPrice \* 1.1

WHERE UnitPrice < 10

Оператор Delete. Применяется для удаления строк из таблицы.

Delete from Products

Where Discontinued =

Оператор Insert: добавление строк таблиц.

Insert Into Customers (Customer ID, CompanyName) values (‘XYZFO’,’XYZ Deli’);

Модификация метаданных

Метаданные – это данные, описывающие страницу БД.

Операция Create table используется для создания новой таблицы. Следующий запрос создает таблицу с именем Simple с 4-мя колонками – LastName, FirstNname, ema-l и HomePage.

Create Table Simple (FirstNname varchar (50) NOT NULL, LastName varchar (50), ema-l varchar (50), HomePage varchar (255).

Оператор ALTER TABLE – применяется для изменения страницы существующей таблицы.

ALTER TABLE Simple

ADD Phone varchar (30);

Оператор DROP – применяется для удаления таблиц или индексов.

DROP TABLE table.

Управление правами доступа средствами языка SQL

У БД может быть несколько пользователей с различными правами доступа. Например, одним пользователям разрешено только просматривать таблицы; другим разрешено просматривать и добавлять новые записи, но запрещено удалять; некоторые пользователи могут устанавливать права доступа к объектам БД для других пользователей.

Разграничение прав доступа является важным средством предостережения БД от неправильного использования содержимой в ней информации различных категорий.

В зависимости от привилегий пользователи подразделяются на следующие категории:

1. администраторы БД;
2. владелец объекта БД;
3. привилегированный пользователь, имеющий право предоставлять привилегии;
4. привилегированные пользователи без права предоставления привилегий;
5. рядовые пользователи.

Создание пользователя:

Create User имяПользователя

[WITH

[SYSID индетификаторПользователя]

[PASSWORD ‘пароль’]]

[CREATEDB | NOCREATE DB]

[CREATEUSER | NOCREATEUSER]

[INGROUP имяГруппы […]]

[VALID UNTIL ‘время’];

Удаление пользователя производится через оператор DROP.

Предоставление привилегий.

GRANT списокПрав

ON объект

TD СписокПользователей

[WITH GRANT OPTION];

* SELECT – право просмотра;
* DELETE – право удаление записей;
* INSERT [(список столбцов)];
* UPDATE [(список столбцов)];
* REFERENCES [(список столбцов)]; // связка двух таблиц
* USAGE – право на домены, наборы символов, сопоставление;
* UNDER – право на стр-рованные типы данных;
* TRIGGER – “счётчик”;
* EXECUTE – право на выполнение;
* ALL PRIVILEGES

Объект в операторе GRANT может принимать следующие значения:

* [TABLE] – имя таблицы;
* DOMAIN – имя домена;
* COLLATION – имя сопоставления;
* CHARACTER SET – имя символьного набора;
* TRANSLATION – имя трансляции;
* TYPE – схематически обозначенный пользовательский тип;
* SEQUENCE – имя генератора последовательности.

Спецификатор указателя шаблона.

Роли и группы.

CREATE ROLE имяРоли;

После создания роли она назначается пользователям;

GRANT имяРоли TO имяПользователей;

Назначение прав для роли производится с помощью следующего выражения:

GRANT СписокПрав

ON объект

TO имяРоли

[WITH GRANT OPTION]

Если реализация SQL поддерживает понятие не роли, а группы, то синтаксис выражения для создания группы может быть следующий:

CREATE GROUP имяГруппы WITH именаПользователей;

Право на просмотр данных.

GRANT SELECT CREATE VIEW Продажи\_вид1АС

ON Продажи SELECT Товар, Количество, Описание

TO PUBLIC; FROM Продажи;

GRANT SELECT

ON Продажи\_вид1

TO PUBLIC;

Право изменять данные.

GRANT UPDATE (Цена) GRANT UPDATE

ON Товары, Продажи ON Товары, Продажи

TO SalesManager; TO SalesManager;

Право удалять данные

GRANT DELETE

ON Клиент, Сотрудник, Товары

TO SalesManager;

Право на использование ссылок.

2 таблицы связаны между собой. В одной таблице внешний ключ ссылается на первичный ключ другой таблицы. Например, таблица «Сотрудник\_Сокращение» содержит список сотрудников, представленных к увольнению. Предположим, что одному сотруднику стало известно, что первичный ключ стал ID, но пользователь не имеет никаких прав, даже на просмотр.

CREATE TABLE mytab (

CREATE INTEGER REFERENCES Сотрудник\_Сокращение)

GRANT REFERENCES (10)

ON Сотрудники\_Сокращение TO PersonManager;

Право на домены.

CREATE DOMAIN Bonux

DECIMAI (9; 2)

CHECK (Bonus >= 0 AND Bonus <= 15000)

CREATE TABLE Платеж\_Ведомость (

10 INTEGER,

Имя VARCHAR (25),

Оклад DECIMAL (9, 2),

Премия Bonus);

GRANT USAGE

ON DOMIAN Bonus;

TO PersonalManager, President;

Право предоставлять права

GRANT UPDATE, INSERT

ON сотрудники

TO SalesManager

WITH GRANT OPTION;

GRANT UPDATE, INSERT

ON сотрудники

TO AssistantManager;

Отмена привилегий.

REVOKE [GRANT OPTION FOR] списокПрав

ON объект

FROM СписокПользователей [RESTRICT | CASCADE];

CASCADE – убираются права и у дочерных сотрудников.

Иногда может возникнуть случай, когда список предоставленных прав предоставляется большему перечню пользователей.

GRANT SELECT

ON Клиенты

TO PUBLIC;

GRANT INSEERT, DELETE

ON Клиенты TO Иванов, Петров;

GRANT UPDATE

ON Клиенты (Имя, Адрес, Телефон, Сумма\_заказов)

TO Иванов, Петров;

Иной способ.

GRANT SELECT

ON Клиенты

TO PuBLIC;

GRANT INSERT, DELETE, UPDATE

ON Клиенты

TO Иванов, Петров;

REVOKE UPDATE

ON Клиенты (ID – клиента) TO Иванов, Петров;

Управляющие серверы

INTERBASE. Структура

**INTERBASE – это SQL-сервер, СУБД, отличающаяся маленькими требованиями к ресурсам системы и при этом довольно серьезной производительностью.** Разумеется, речь идет в основном про управление БД объемом до нескольких гб. В таких ситуациях InterBase является, пожалуй, наилучшим выбором из некоторого числа других серверов.

**Достоинства:**

— кроссплатформенность то есть, переносимость с одной ОС на другую;

— отсутствие блокировок по чтению и быстрое восстановление БД при сбоях;

— каскадные триггеры;

— BLOB-поля.

BLOB поля – это массив двоичных данных, специальный тип данных, предназначенный, в первую очередь, для хранения изображений, аудио и видео, а также компилированного программного кода.

Триггеры представляют собой особый вид хранимых процедур, привязанных к таблицам и представлениям и автоматически срабатывающие при добавлении, удалении или изменении данных.

Применение InterBase

* Моторола;
* Нокия;
* Боинг;
* Бостонская фондовая биржа;
* Абрамс;
* немецкая железная дорога.

В 1976 г. Джим Старки, сотрудник компании, DEC разрабатывает БД.

В 1983 г. появляется компания Borland.

В 1985 г. Джим Старки вмести с коллегами организовывает компанию Groton Database Systems.

В 1986 г. создана Interbase.

Дисковая структура БД INTERBASE.

БД состоит из одного или нескольких файлов. Файл БД в свою очередь состоит из набора страниц фиксированной длины, размер которых указывается при создании БД и в дальнейшим каждый файл состоит из страниц одинакового размера.

* Страница-заголовок (Header page);
* Страница указателей (Pointer page);
* Страница учёта транзакций (Transaction page);
* Страница распределения места (Space inventory page);
* Страница генераторов (Generator page);
* Страница основного индекса (Index root page);
* Страница индексов (Index page);
* Страница данных (Data page).

**Страница заголовок (Header page)** – это первый блок в первом файле БД. Когда InterBase присоединяется к БД он считывает первый килобайт файла. Заголовок содержит в себе критическую информацию о БД. Она включает в себя: номер версии дисковой структуры и размер страницы. После этого InterBase проверяет поддерживается ли сервером версия дисковой структуры. При положительном ответе он перечитывает страницу заголовок, используя правильный размер страницы и узнает необходимую информацию такую как, имена и диапазон страниц, содержащийся во вторичных файлах БД, следующую доступную транзакцию и последнюю интересующую нас транзакцию. На следующем шаге ищется ядро системных таблиц и строится внутреннее представление БД.

**Страница указателей (Pointer page)**

Страница заголовок содержит также информацию о местоположения первой страницы указателей для системной таблицы. Данная таблица InterBase найти критичные страницы, включающие страницы данных для таблиц. Система читает первую страницу указателей и использует её для поиска первой страницы данных, а также индексов и других системных таблиц. Страница указателей имеет простой заголовок, который включает в себя тип страницы, номер следующей страницы указателей данной таблицы. Остальная часть страницы заполнена массивом из четырехбайтовых чисел, который является номерами страниц и составляют данную таблицу.

**Страница учёта транзакций (Transaction page)**

Подобно странице указателей страница учёта транзакций также состоит из простого заголовка, который включает в себя тип страницы и номер следующей страницы учёта транзакций. Оставшаяся часть страницы заполняется массивом двухбитовых записей, которые отражают состояние транзакций в системе. 0 указывает, что транзакция не была начата, или была активна и «погибла» без подтверждения или отката. 1 указывает, что транзакция была подтверждена. Двойка указывает на то, что транзакция была опровергнута. 11 указывает на то, что транзакция находится в состоянии «лимбо» (неопределенное состояние, которое существует в середине двухфазового подтверждения транзакции).

Чтобы определить состояние транзакций InterBase использует номер транзакции как индекс в массиве транзакций и просматривает состояние соответствующих битов. Алгоритм сложный, т.к. зависит от заголовка страниц. В случае, если одна транзакция проверяет состояние другой транзакции и обнаруживает, что она обозначена как активная, а фактически является «мёртвой», то меняет состояние «мёртвой» транзакции с активной на «откаченную».

**Страница распределения места** показывает какие страницы являются распределенными (зарезервированными) и, если свободна страница, то насколько она свободна. Остальная часть, кроме заголовков, таблицы заполнена массивом однобитых кластеров, которые соответствуют страницам БД.

При создании каждая новая таблица добавляется в таблицу распределения места и отражает состояние страницы.

**Страница генераторов** состоит из заголовка и четырехбайтовых записей, которые представляют состояние генераторов. Индексом в этом массиве служит порядковый номер генератора.

**Страница основного индекса**. Любые таблицы, включая те, у которых нет индексов, имеют страницу основного индекса. Страница основного индекса отождествляется с вершиной каждого индекса, определенного для таблицы. На страницы должны быть перечислены поля индексации и указана полезность индекса, которая может быть рассчитана как отношение числа, различающихся индексных полей внутри индекса и среднего количества записей. Среднее количество записей – это число страниц БД, занятых данной таблицей, деленной на максимальное количество записей на страницу. Полезность индекса – чрезвычайно важный показатель, на основании которого InterBase оптимизирует выполнения запроса.

**Страница индексов**

Также состоит из заголовка, который включает собственный индекс и следующей страницы, а остальная часть содержит записи, указывающую на точки следующих уравнений индексов страниц, т.к. индексы представляют собой древовидную структуру.

**Страница-блок**

Страницы-блок полностью заняты двоичными данными, которые при определенных условиях можно выделить в отдельный тип страниц. Для каждого созданного блок-поля создается запись, содержащая информацию о расположении данных поля и данные о содержимом, которые могут быть полезны при чтении. Механизм хранения таких полей определяется их размером и бывает трёх типов (0, 1, 2).

**Механизм 0.** Поле умещается на одной страницы БД вместе с записью.

**Механизм 1.** Поле не умещается на одной страницы БД вместе с записью, тогда поле записывается в специализированные страницы, а в поле двоичных данных помещается массив указателей на занятые полем страницы.

**Механизм 2.** Когда на начальной странице не хватает места даже для того, чтобы записать туда массив указателей, тогда InterBase создает страницы указателей на страницы с полем 0.

Понятие доменов

Домен – это базовый тип данных, к которому относятся элементы этого домена, к понятию домена относится его понимание, как допустимого множественного значения в БД. Домен имеет семантическую нагрузку. Данные считаются сравнимыми только в том случае, когда они относятся к одному домену.

Краеугольным камнем грамотного описания структуры данных в InterBase является понятие домена (domain), абстрактно определяющего столбец таблицы БД.

Домены чем-то напоминают шаблоны, по которым описываются столбцы конкретных таблиц. Причем эти шаблоны могут включать проверки not NULL и другие ограничения.

Поддержка доменов в InterBase соответствует стандарту ANSI SQL – 92 и по сравнению с другими серверами БД является более чем уникальной.

Введенные в язык описания данных DDL севера InterBase ещё в 1989 г., домены до сих пор полностью соответствуют концепции, предложенной Коддом.

Пример описания домена

CREATE DOMAIN CUSTNO AS INTEGER DEFAULT 9999 CHECK (VALUE > 1000);

В БД InterBase не полностью реализовано понятие доменов. Домены служат не для связи таблиц по первичному и внешнему ключу, а для описания типов поля определенного пользователя.

Пример.

**CREATE DOMAIN** IZMER\_NUM **INTEGER NOT NULL**;

**CREATE DOMAIN** ACTIVITIES\_NUM **INTEGER NOT NULL**;

…

**CREATE DOMAIN** NAMES\_TYPE **VARCHAR**(45) **COLLATE** PXW\_CYRCL; #Конструкция PXW\_CYRCL позволяет правильно вести сортировку строк таблицы по полю типа этого домена

**CRATE DOMAIN** FLOAT\_TYPE **DOUBLE PRECISION**; #Тип данных: двоичная точность при округлении.

**CREATE DOMAIN** BOOL\_TYPE **CHAR**(1) **DEFAULT** “F”; #Эта конструкция проверяет выход значения поля за заданные границы

**CHECK** (**VALUE** = “T’ **OR VALUE** = “F”);

**CREATE DOMAIN** FORMULA FORMULA\_TYPE **BLOB SUB\_TYPE** 1 **SEGMENT SIZE** 256 **CHARACTER SET** WIN1251; #Эта конструкция создает домен типа «блок», то есть набор байтов, который рассматривается как текст; в страничке файла БД для этого текста выделяется по 256 байт сразу, при чем текст записывается в кодировке WIN1251

**CREATE DOMAIN** INTEGER\_TYPE **INTEGER**;

…

**CREATE DOMAIN** BY\_USER VARCHAR(30) DEFAULT USER;

**CREATE DOMAIN** BY\_DAT **TIMESTAMP DEFAULT** “now”;

**CREATE TABLE** IZMER\_NAMES

(

ID\_NUM IZMER\_NUM,

NAME NAMES\_TYPE,

USER\_NAME BY\_USER,

CHANGE\_DATE BY\_DATE,

**PRIMARY KEY** (ID\_NUM)

);

**CREATE TABLE** ACTIVITIES

(

ID\_NUM ACTIVITIES\_NUM,

ID\_IZMER\_NAMES IZMER\_NUM,

POZITION INTEGER\_TYPE,

NAME NAMES\_TYPE,

NAME NAMES\_TYPE,

IS\_DECCIPHERAD\_INFO BOOL\_TYPE,

USER\_NAME BY\_USER,

CHANGE\_DATE BY\_DATE,

**PRIMARY KEY**(ID\_NUM),

**FOREIGN KEY**(ID\_IZMER\_NAMES) **REFERENCES**

IZMER\_NAMES(ID\_NUM)

);

Правила создания таблиц

Сначала создаются справочники, т.е. таблицы, которые не имеют внешний связей.

Затем создаются таблицы, которые имеют связи со справочниками. Необходимо помнить, что связей должно быть меньше на 1 ед., чем количество самих таблиц.

Не используйте визуальные программы, вводите всё в виде SQL запросов.

Суррогатные ключи

Есть два типа ключевых полей: естественный и суррогатный.

Суррогатный ключ – это поле целого типа, которое имеет уникальное значение, образующее домен с другими таблицами.

Например, естественный ключ – это номер медицинской карты.

К медицинской карте «цепляются» талоны, у которых естественными ключами будут отчётный год и номер талона. К талонам «прицепляется» таблица посещения.

Естественными ключами посещения будут дата посещения, отчётный год, номер медицинской карты.

Мы видим, что размер естественного ключа увеличивается с каждой новой таблицей. Соответственно растёт вычислительная нагрузка на сервер БД. Противостоять такому разрастанию помогают суррогатные ключи.

Например, для таблицы с талонами вводится уникальное поле целого типа с последовательностью чисел и объявляется первичным ключом.

Механизм поддержки уникальности значений суррогатного ключа.

**CREATE GENERATOR** GET\_IZMER\_NAMES\_NUM;

**SET GENERATOR** GET\_IZMER\_NAMES\_NUM TO 50;

**SET TERM** !!;

**CREATE PROCEDURE** SET\_IZMER\_NAMES\_NUM

RETURNS(NUM INTEGER)

**AS**

**BEGIN**

NUM = GEN\_ID(GET\_IZMER\_NAMES\_NUM, 1);

**END**!!

**SET TERM**; !!

«Деревянные» списки

Бывают случаи, когда отношения «главный-подчиненный» присущи записям одной и той же таблицы. Например, отношения между отделами организации или структуры госаппарата. Одна запись может быть главной для нескольких структур, которые, в свою очередь, могут быть главными для следующих. Первая запись – это главный узел, от которого идут ветви подчиненных записей. Если подчиненные записи имеют свои вложенные или подчиненные записи, то они образуют узлы по иерархическому списку.

Проще всего это представить в пространстве в виде слоёв записей. Несмотря на сложность, реализация такой структуры довольна проста. Для этого нужно иметь, как минимум, два столбика таблицы. Первый – это суррогатный первичный ключ, а второй – это ссылка на первый столбик со значением первичного ключа записи владельца.

**CREATE TABLE** ACTIVITIES

(

ID\_NUM ACTIVITIES\_NUM,

ID\_OWNER ACTIVITIES\_NUM,

ID\_IZMER\_NAMES IZMER\_NUM,

POZITION INTGE\_TYPE,

NAME NAMES\_TYPE,

USER\_NAME BY\_USER,

CHANGE\_DATE BY\_DATE,

**PRIMARY KEY**(ID\_NUM),

**FOREIGN KEY**(ID\_IZMER\_NAMES) REFERENCES

IZMER\_NAMES(ID\_NUM)

);

Таблица содержит первичный ключ ID\_NUM. Содержит ссылку на главную запись поля ID\_OWNER. Поле POZITION целого типа определяет позицию записи для возможности передвижения вверх и вниз.

**CREATE GENERATOR** GET\_ACTIVITIES\_NUM;

**SET GENERATOR** GET\_ACTIVITIES\_NUM TO 50;

**SET TERM** !!;

**CREATE PROCEDURE** SET\_ACTIVITIES\_NUM

**RETURNS**(NUM INTEGER)

**AS**

**BEGIN**

NUM = GEN\_ID(GET\_ACTIVITIES\_NUM, 1);

**END**!!

**SET TERM**;!!

Далее создадим индекс для сортировки по полю POZITION.

**CREATE UNIQUE INDEX** ACTIVITIES\_POSITON ON

ACTIVITIES(ID\_OWNER, POZITION);

#ТРИГГЕР UPDATE\_ACTIVITIES

**SET TERM** !!;

**CREATE TRIGGER** UPDATE\_ACTIVITIES FOR ACTIVITIES

BEFORE UPDATE **AS**

**BEGIN**

NEW.USER\_NAME = USER;

NEW.CHANGE\_DATE=’now’

**END**!!

**SET TERM**; !!

Теперь добавим внешний индекс таблицы на саму себя. Раньше этого сделать нельзя, т.к. ни самой таблицы, ни полей ID\_OWNER, ID\_NUM не существовало.

**ALTER TABLE** ACTIVITIES

**ADD**

**FOREIGN KEY** (ID\_OWNER) **REFERENCES** ACTIVITIES(ID\_NUM)

**ON DELETE CASCADE**;

Следующий запрос перемещает записи в слое вверх-вниз. Подразумевается, что в слое не больше 2млрд записей.

**SET TERM** !!;

**CREATE PROCEDURE** SET\_ACTIVITIES\_POSITION(OWNER\_NUM INTEGER, OLD\_POSITION INTEGER, NEW\_POSITION ITNEGER)

**AS**

**BEGIN**

**UPDATE** ACTIVITIES

**SET**

POZITION = 2147483647

**WHERE**

POZITION=:NEW\_POSITION **AND**

ID\_OWNER=:OWNER\_NUM;

**UPDATE** ACTIVITIES

**SET**

POZITION=:NEW\_POSITION

**WHERE**

POZITION=:OLD\_POSITION **AND**

ID\_OWNER=:OWNER\_NUM;

**UPDATE** ACTIVITIES

**SET**

POZITION=:OLD\_POSITION

**WHERE**

POZITION=2147483647 **AND** ID\_OWNER=:OWNER\_NUM;

**END**!!

**SET TERM**; !!