ЛЕКЦИЯ: 2 ВИДЫ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ И ИХ ЦЕЛОСТНОСТЬ

ТРЕБОВАНИЯ К СУБД:

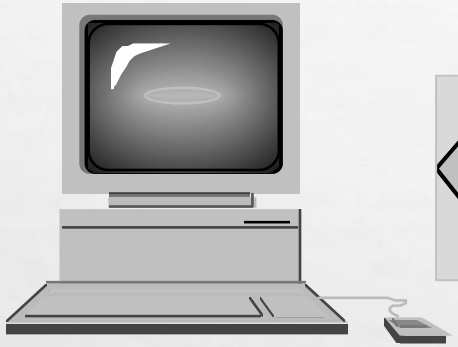
- поддержка целостности данных;
- согласованное хранение независимых наборов данных;
- извлечение данных и управление данными во внешней и оперативной памяти;
- надежное хранение данных несмотря на возможность сбоя в программных или технических средствах;
- одновременный доступ к данным нескольким пользователям;
- управление транзакциями;
- поддержка языков БД;
- масштабирование;
- мультиплатформенность;
- поддержка стандартов.

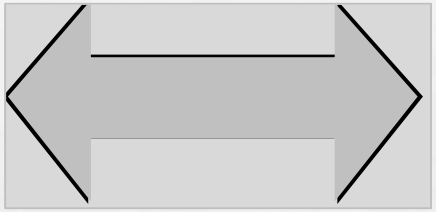
ЦЕЛОСТНОСТЬ ДАННЫХ

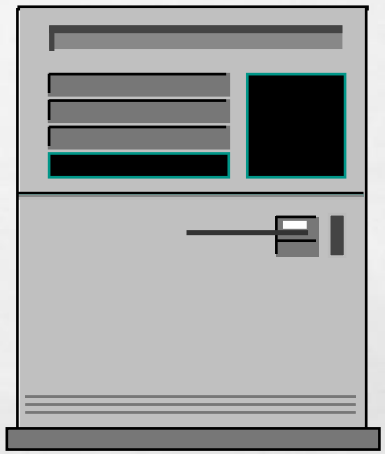
Целостность информации — данные не изменяются при передаче, хранении или отображении.
Целостность базы данных (database integrity) или согласованность, или корректность и непротиворечивость — соответствие имеющейся в БД информации её внутренней логике, структуре и явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется **ограничением целостности** (integrity constraint).
Целостность БД не гарантирует достоверности содержащейся в ней информации, но обеспечивает правдоподобность этой информации, отвергая заведомо невероятные, невозможные значения.
Ссылочная целостность - исключение ошибки связей между первичным и вторичным ключом. Примеры нарушений целостности:

- существование записей-сирот (дочерних записей, не имеющих связи с родительскими записями);
- существование одинаковых первичных ключей.

Независимость от платформ







Операционные системы

- OS/390
- TRU64
- Solaris
- AIX
- HP Unix
- NT
- Linux
- *Другие*

Оконные менеджеры

- MS Windows
- X Motif
- Macintosh
- *Другие*

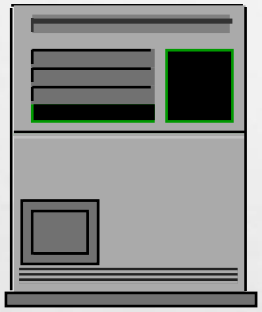
Сетевые протоколы

- TCP/IP
- LU6.2
- SPX/IPX
- OSI
- DECnet
- *Другие*

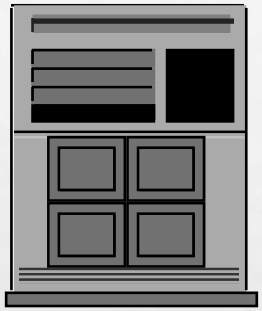
Оборудование

- Compaq
- Sun
- HP
- IBM
- Mac
- *Другие*
- NCR
- Pyramid
- Sequent
- Sun
- Intel

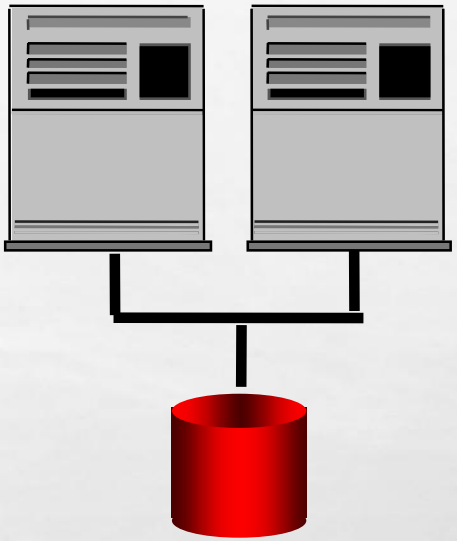
Независимость от архитектуры



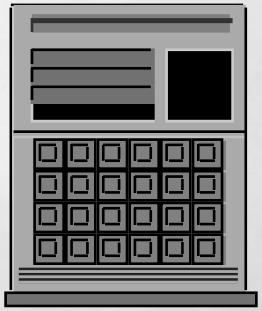
Один процессор



Симметричная многопроцессорная архитектура (SMP)

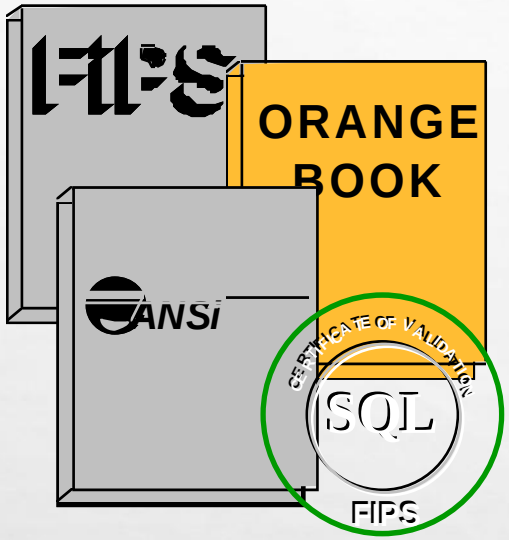


Слабо сцепленные процессоры (кластер)



Массивно параллельный компьютер (MPP)

Поддержка стандартов



Комитеты
- ANSI X3H2
- X3H2.1 RDA
- SQL Access Group
- OMG

Стандарты баз данных
- FIPS 127-2
- ANSI X3-135.1992

Стандарты защиты данных:
- NCSC TDI C2, B1
- ITSEC F-C2/E3, F-B1/E3

Сетевые стандарты
- OSI
- DNSIX (MaxSix)

Межоперабельность
- IDAPI, ODBC
- TSIG
- X/Open
- DCE
- DDE

История PostgreSQL

Свободно распространяемая объектно-реляционная СУБД.

- *1977-1985гг.* **Ingres** - «тренировочный» проект создания классической реляционной системы управления базами данных. Разрабатывался под руководством М. Стоунбрейкера в Калифорнийском университете в Беркли
- *1986-1994гг.* **Postgres** = Post Ingres – команда Стоунбрейкера разрабатывали новую СУБД, при создании которой использовались многие ранее сделанные наработки. Были введены процедуры, правила, пользовательские типы и многие другие компоненты.
- *1995г – по наст. вр.* **PostgreSQL** - разработка разделилась: Стоунбрейкер - создание коммерческой СУБД Illustra (потом Informix), а его студенты - разработка Postgres95, в которой язык запросов POSTQUEL — наследие Ingres — был заменен на SQL. Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. С этого момента СУБД получила имя, под которым она известна и развивается в текущий момент — PostgreSQL.

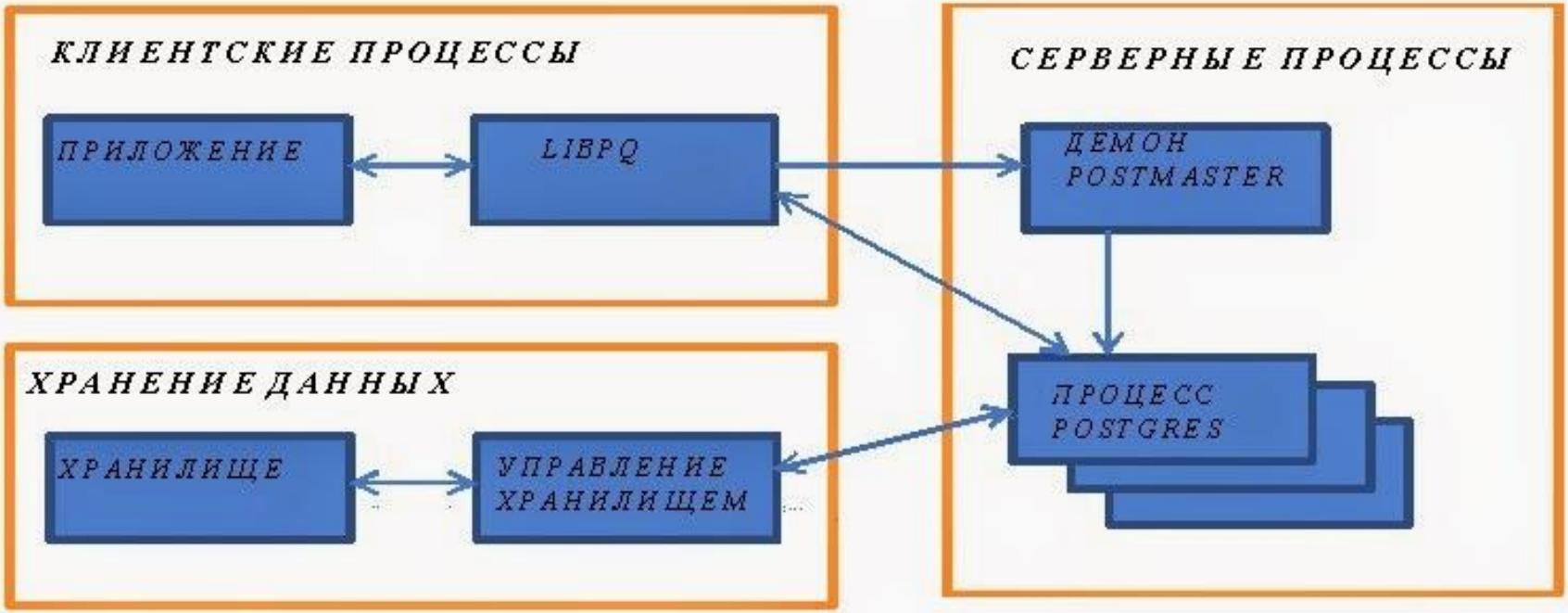
СУБД PostgreSQL

- наиболее развитая СУБД с открыты кодом;
- надежность и устойчивость при больших нагрузках;
- кросс-платформенность: работает в широком диапазоне диалектов UNIX (Linux, FreeBSD, Solaris и т.д.), а также на платформе Microsoft Windows;
- высокий уровень соответствия стандартам;
- существует множество интерфейсов и библиотек взаимодействия для других языков программирования: Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme и Qt;
- расширяемость;
- быстродействие;
- поддержка баз данных практически неограниченного размера.

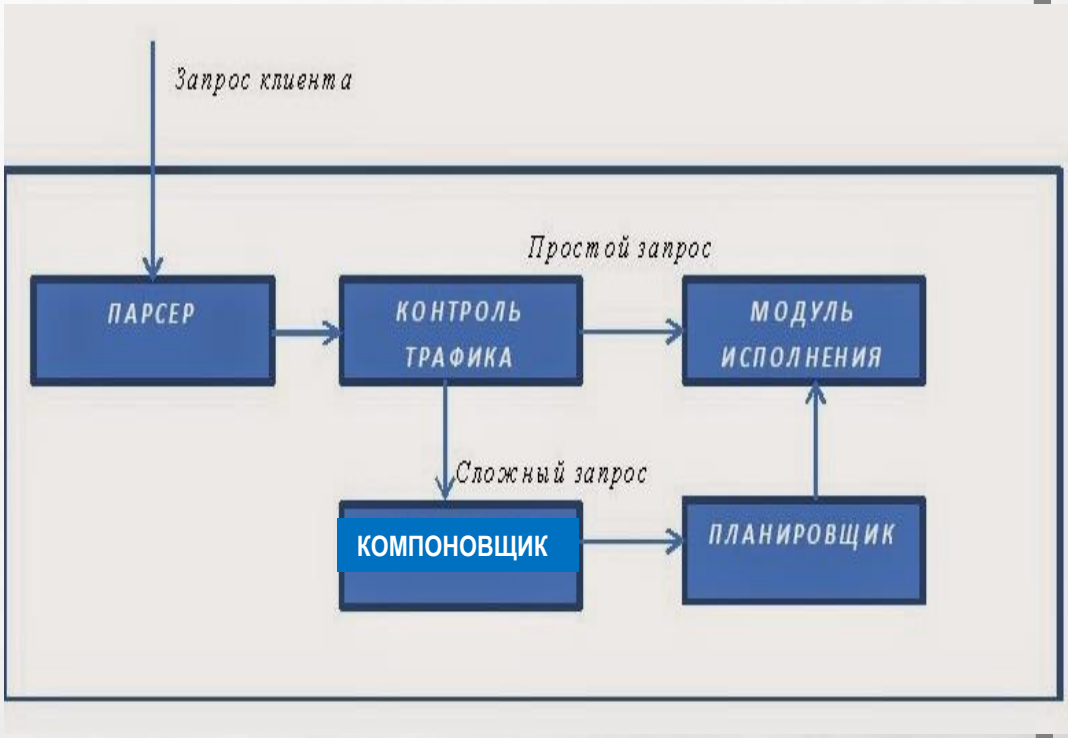
Архитектура PostgreSQL

Архитектура разбита на 3 основные подсистемы:

- **Front End** - клиентская часть системы.
- **Серверная часть** - серверные процессы и контролирующий процесс Postmaster, отвечающий за взаимодействие с клиентами.
- **Back End** - включающий хранилище данных и средства управления хранилищем.



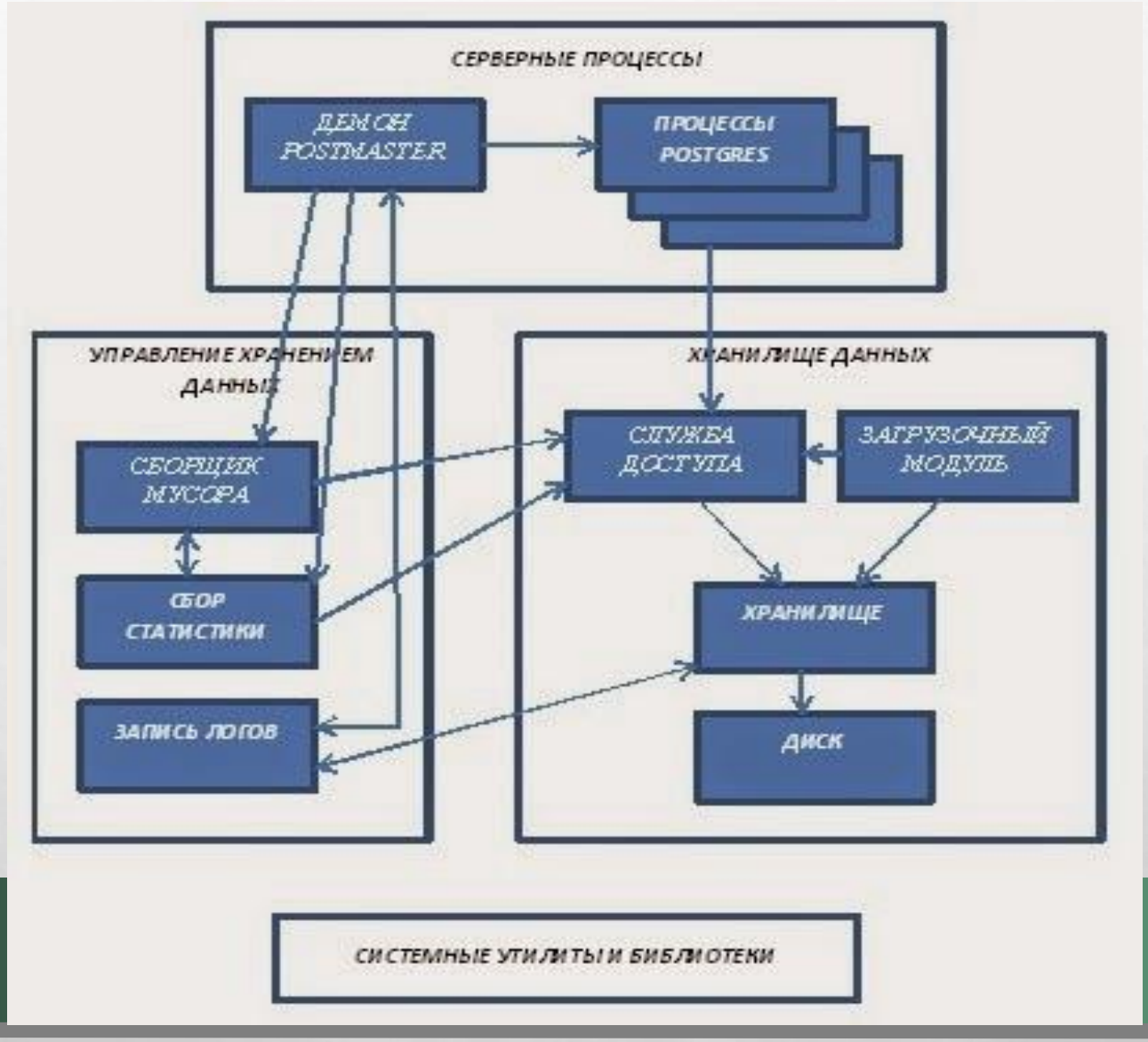
Серверная часть



Обработка запроса :

- **Парсер** принимает запрос и проверяет его синтаксис. В результате формируется дерево разбора или в случае неверного синтаксиса генерируется ошибка.
- **Служба контроля трафика**. Простые запросы исполняются, а сложные передаются компоновщику.
- **Компоновщик** принимает дерево разбора запроса и преобразует его в альтернативную вспомогательную форму.
- **Планировщик** определяет наиболее оптимальный способ выполнения сложного SQL-запроса и передаёт управление модулю исполнения.
- **Модуль исполнения** выполняет запрос.

Средства управления хранилищем



1) доступ к данным
2) управление хранилищем
3) системные утилиты

Средства управления хранилищем

***Взаимодействие с хранилищем*:**

- **Службы доступа -- связь между процессами Postgres и физическим диском. Они обеспечивает семафоры и блокировки файлов, а также отвечают за индексирование, сканирование, поиск, компиляцию и возвращение запрошенных данных.**
- **Образец базы данных создаётся с помощью загрузочного модуля при первом запуске СУБД.**

***Управление хранилищем* включает несколько подсистем:**

- **Модуль сбора статистики - информация о доступе к таблицам и индексам, вызовах серверных функций и командах, выполненных модулем исполнения.**
- **Сборщик мусора Auto-Vacuum —автоматическое освобождение неиспользуемой памяти в таблицах.**
- **Фоновый процесс записи логов – журналирование произведённых операций и информация для резервного восстановления системы в случае сбоя.**

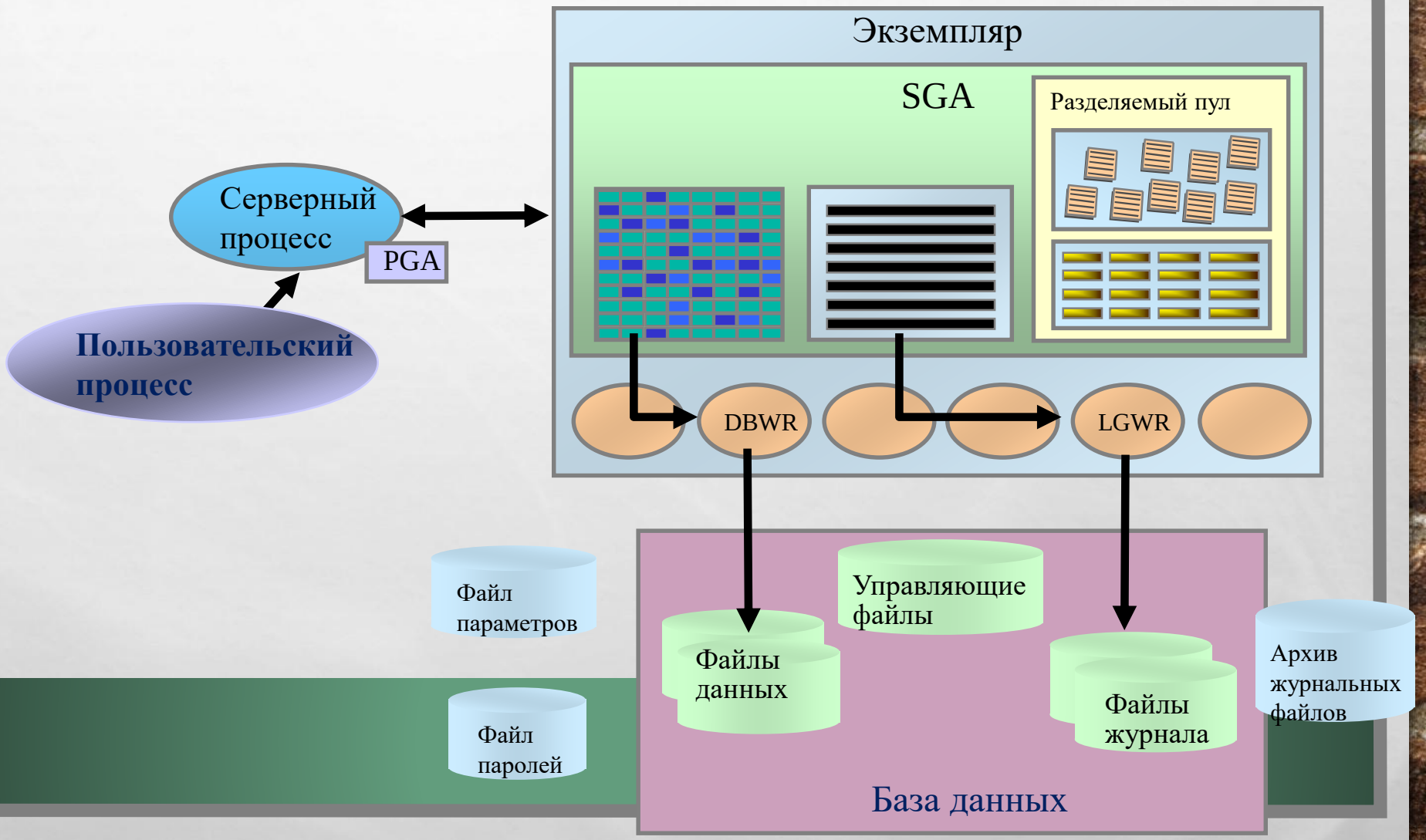
***Системные утилиты* предоставляет некоторые общие функции для процессов серверной части.**

Компоненты архитектуры SQL Server

Файлы. Имеется пять видов файлов, образующих базу данных и поддерживающих экземпляр - файлы параметров, сообщений, данных, временных данных и журналов повторного выполнения.
Структуры памяти, в частности системная глобальная область (System Global Area — SGA), входящие в SGA Java-пул, разделяемый пул и большой пул, а также PGA (Program Global Area) и UGA (User Global Area).
Физические процессы или потоки. Три типа процессов, образующих экземпляр: серверные процессы, фоновые процессы и подчиненные процессы.

База данных — набор физических файлов операционной системы;
Экземпляр — набор процессов Oracle и область SGA
SGA — область памяти, содержащая внутренние структуры данных, доступ к которым необходим всем процессам для кэширования данных с диска, кэширования данных повторного выполнения перед записью на диск, хранения планов выполнения разобранных операторов SQL и т.д.

База данных и instance (экземпляр)

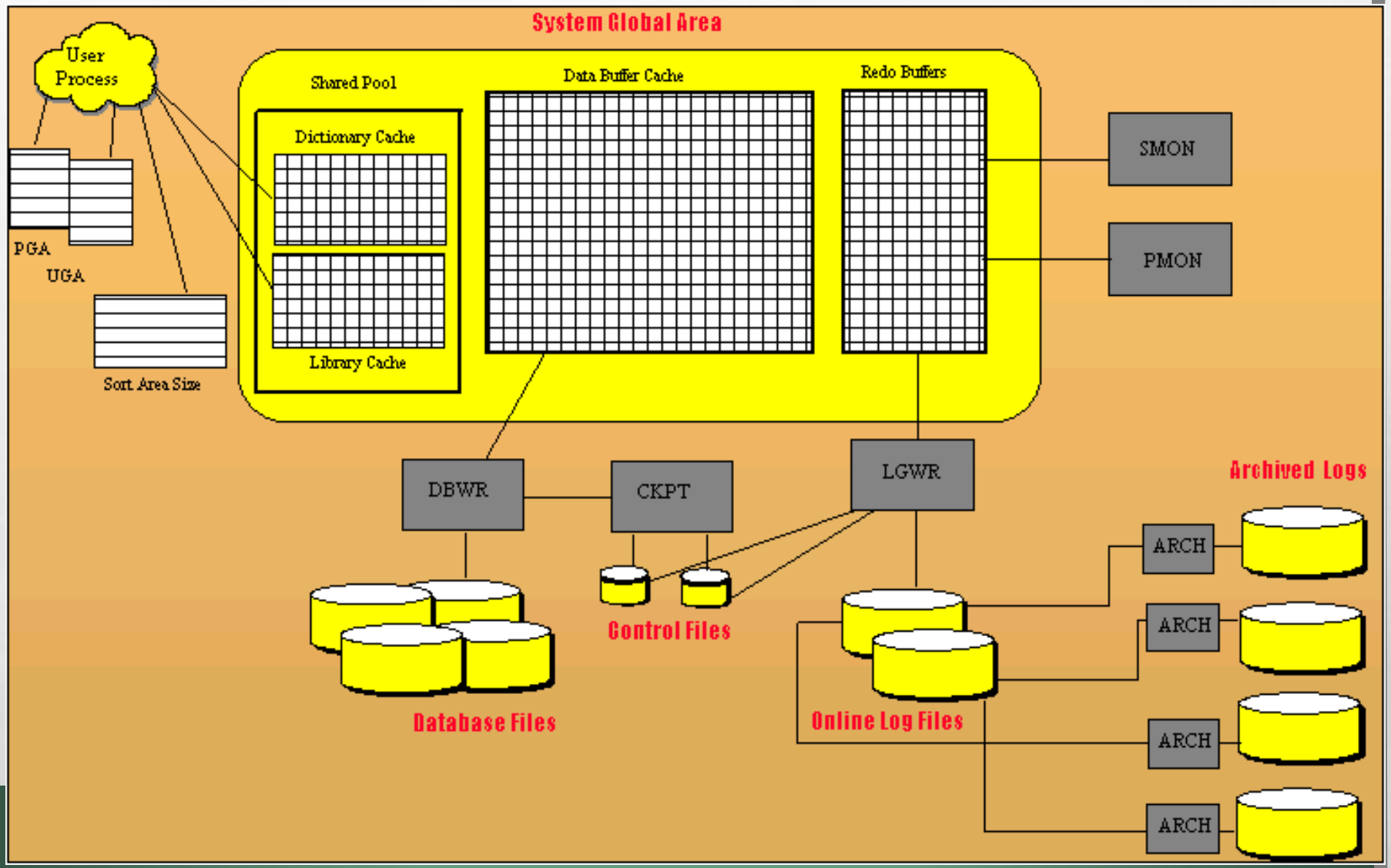


Файлы СУБД

В состав базы данных и экземпляра входит шесть типов файлов.
С экземпляром связаны файлы параметров (1). По этим файлам экземпляр при запуске определяет свои характеристики, например размер структур в памяти и местонахождение управляющих файлов.
Базу данных образуют следующие файлы:

- файлы данных (2). Собственно данные (в этих файлах хранятся таблицы, индексы и все остальные сегменты);
- файлы журнала повторного выполнения (3). Журналы транзакций;
- управляющие файлы (4). Определяют местонахождение файлов данных и содержат другую необходимую информацию о состоянии базы данных;
- временные файлы(5). Используются при сортировке больших объемов данных и для хранения временных объектов;
- файлы паролей (6). Используются для аутентификации пользователей, выполняющих администрирование удаленно, по сети.

Взаимосвязь структур хранения и процессов



Серверные процессы

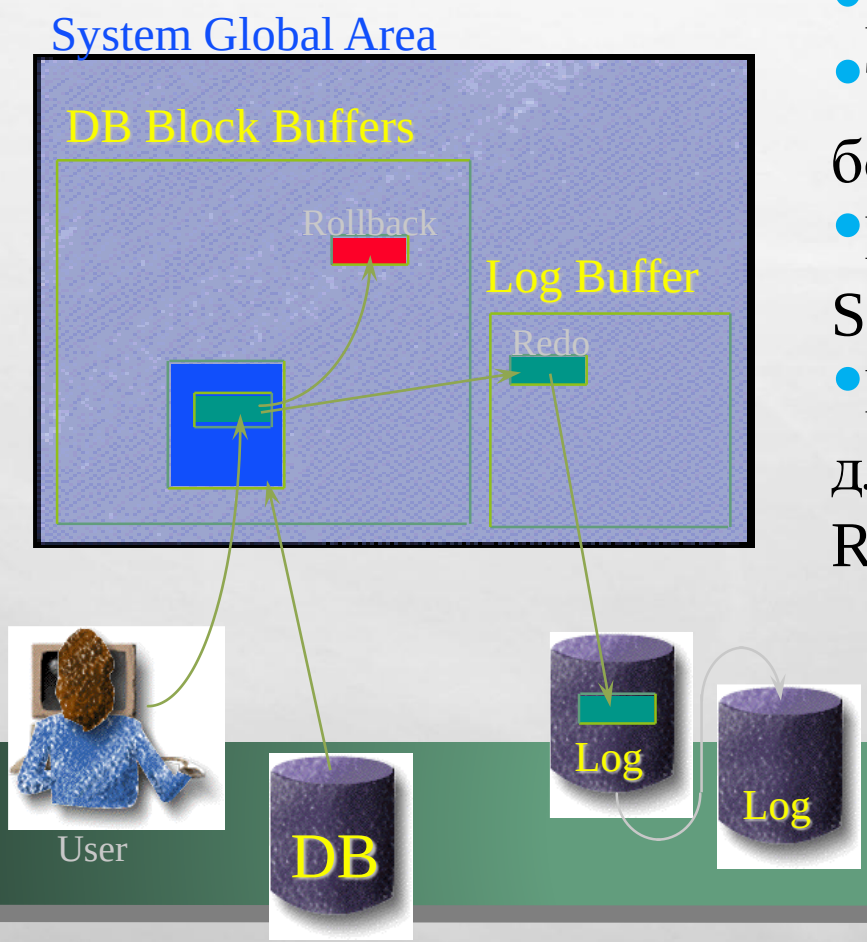
Типовые процессы:

- ***ckpt*** - процесс отвечающий за то, чтобы все изменения данных в памяти были записаны на диск;
- ***pmon*** - обеспечивает наблюдение за пользовательскими процессами и высвобождение ресурсов по их завершении;
- ***smon*** - обеспечивает дефрагментацию места в БД;
- ***reco*** - отвечает за распределенные транзакции;
- ***dbwr*** - отвечает за сохранение измененных данных на диск;
- ***lgwr*** - отвечает за запись в redo log файлы;
- ***arc*** - отвечает за архивирование redo log файлов

Процесс, обеспечивающий подключение по сети: ***listener***

В Oracle 11g, 12c процессов, поддерживающих экземпляр базы данных, больше.

ЖУРНАЛИРОВАНИЕ И ROLLBACK



- Журналы используются только для восстановления БД (redo).
- Rollback отделены от Log.
- Транзакции могут использовать больше одного журнала.
- Rollback хранятся в Rollback Segments (Undo Tablespace).
- Rollback Segments используются для Undo (отката) и Multi-Version Read Consistency.

СТРУКТУРА ХРАНИМЫХ ДАННЫХ

Единица хранения данных в БД – **хранимая запись**.
Хранимая запись состоит из двух частей:

1) ***служебная часть***. Используется для идентификации записи, задания её типа, хранения признака логического удаления, для кодирования значений элементов записи, для установления структурных ассоциаций между записями и проч. Никакие пользовательские программы не имеют доступа к служебной части хранимой записи.
2) ***информационная часть***. Содержит значения элементов данных. Поля хранимой записи могут иметь *фиксированную* или *переменную длину*. Хранение полей переменной длины осуществляется одним из двух способов: размещение полей через разделитель или хранение размера значения поля. Наличие полей переменной длины позволяет не хранить незначащие символы и снижает затраты памяти на хранение данных; но при этом увеличивается время на извлечение записи.

СТРУКТУРА ХРАНИМЫХ ДАННЫХ

Каждой хранимой записи БД система присваивает внутренний идентификатор, называемый по стандарту CODASYL **ключом базы данных** (КБД).

В Oracle используется термин *идентификатор строки*, RowID.

Значение КБД формируется системой при размещении записи и содержит информацию, позволяющую однозначно определить место размещения записи (преобразовать значение КБД в адрес записи).

Примеры:

1) Формат DBF: 1 таблица – 1 файл, записи фиксированной длины – в качестве КБД выступает последовательный номер записи в файле (относительная адресация).

2) СУБД SQL Server – совокупность номера экстента, блока и номера строки в блоке (относительная адресация).

3) Абсолютный адрес в памяти (СУБД Adabas).

УПРАВЛЕНИЕ ПРОСТРАНСТВОМ ПАМЯТИ

- Для обеспечения более эффективного управления ресурсами и/или для технологического удобства всё пространство памяти БД обычно разделяется на части (области, сегменты и др.).
- Области разбиваются на пронумерованные *страницы (блоки)* фиксированного размера. В большинстве систем обработку данных на уровне страниц ведёт операционная система (ОС), а обработку записей внутри страницы обеспечивает только СУБД.
- Страницы представляются в среде ОС блоками внешней памяти или секторами, доступ к которым осуществляется за одно обращение. Некоторые СУБД позволяют управлять размером страницы (блока) для базы данных.
- В каждой *области памяти*, как правило, хранятся данные одного объекта БД (одной таблицы).
- Сведения о месте расположения данных таблицы (ссылка на область хранения) СУБД хранит в словаре-справочнике данных (ССД).

Структура страницы памяти

УПРАВЛЕНИЕ ПРОСТРАНСТВОМ ПАМЯТИ

Способы управления свободным пространством памяти на страницах:

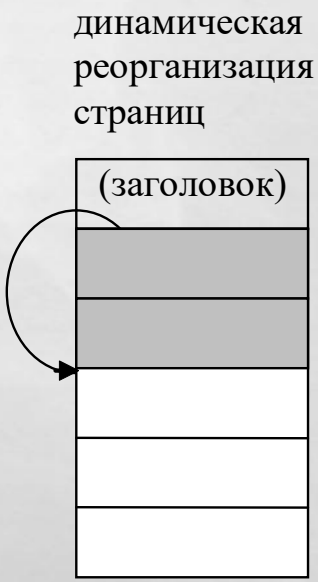
- ведение списков свободных участков;
- динамическая реорганизация страниц.

При **динамической реорганизации страниц** записи БД плотно размещаются вслед за заголовком страницы, а после них расположен свободный участок. Смещение начала свободного участка хранится в заголовке страницы.

Достоинство такого подхода – отсутствие фрагментации.

Недостатки:

- ✓ Адрес записи может быть определён с точностью до адреса страницы, т.к. внутри страницы запись может перемещаться.
- ✓ Поиск места размещения новой записи может занять много времени. Система будет читать страницы одну за другой до тех пор, пока не найдёт странницу, на которой достаточно места для размещения новой записи.

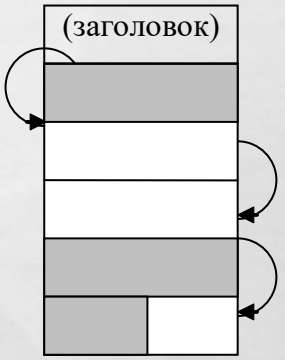


УПРАВЛЕНИЕ ПРОСТРАНСТВОМ ПАМЯТИ

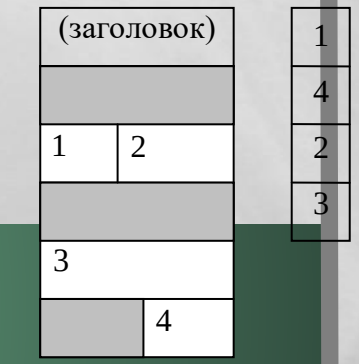
Ведение **списков свободных участков**.
Здесь можно рассмотреть два варианта:

1. Ссылка на первый свободный участок на странице хранится в заголовке страницы, и каждый свободный участок хранит ссылку на следующий (или признак конца списка). Каждый освобождаемый участок включается в список свободных участков на странице.
2. Списки свободных участков реализуются в виде отдельных структур. Эти структуры также хранятся на отдельных *инвентарных страницах*. Каждая инвентарная страница относится к области (или группе страниц) памяти и содержит информацию о свободных участках в этой области. Список ведётся как стек, очередь или упорядоченный список. В последнем случае упорядочение осуществляется по размеру свободного участка, что позволяет при размещении новой записи выбирать для неё наиболее подходящий по размеру участок.

списки свободных участков на странице



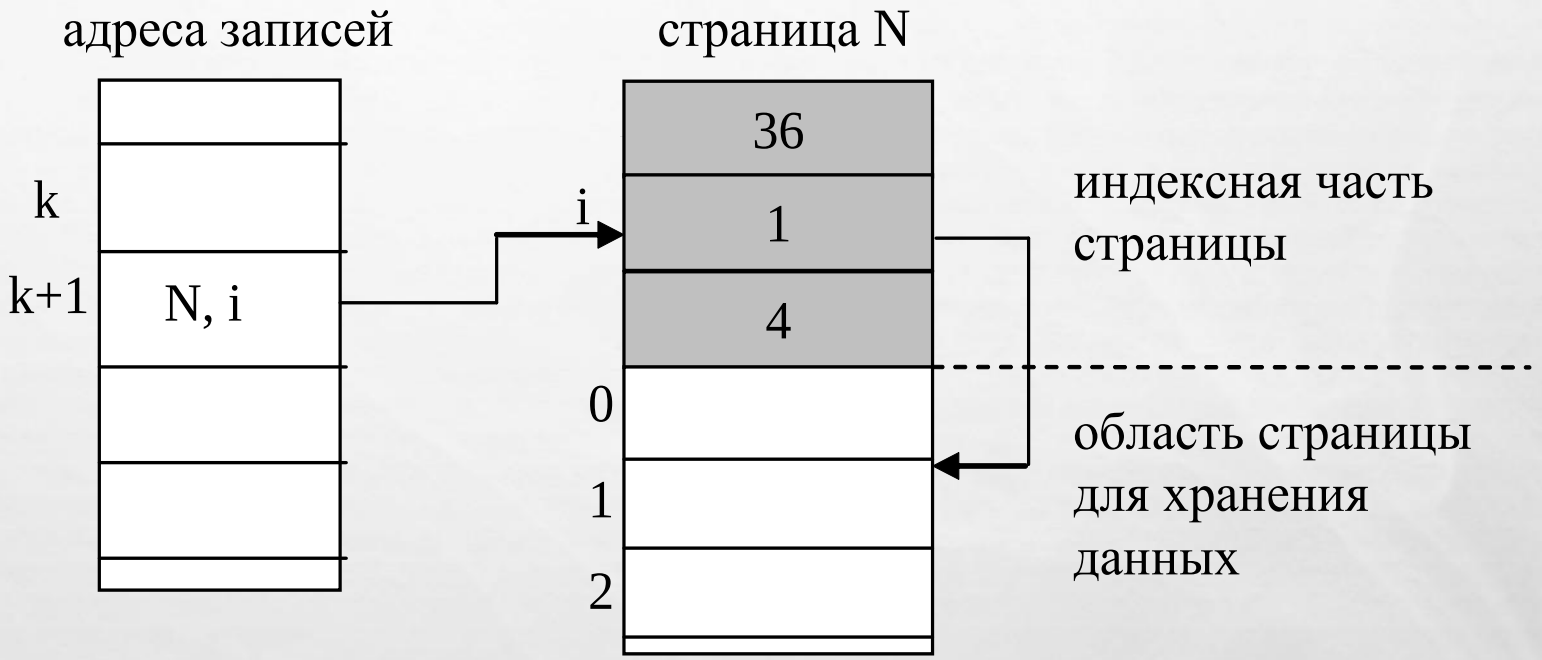
списки свободных участков в виде отдельных структур



ВИДЫ АДРЕСАЦИИ ХРАНИМЫХ ЗАПИСЕЙ:

- **Прямая адресация** предусматривает указание непосредственного местоположения записи в пространстве памяти (например, в системе ADABAS). Недостатки: большой размер адреса; прямая адресация не позволяет перемещать записи в памяти без изменения КБД, что ведёт к фрагментации памяти.
- **Косвенная адресация**. В качестве КБД выступает не сам "адрес записи", а адрес места хранения "адреса записи".
- **Относительная адресация**. Общий принцип относительной адресации заключается в том, что адрес отсчитывается от начала той области памяти, которую занимают данные объекта БД. Если память разбита на страницы (блоки), то адресом может выступать номер страницы (блока) и номер записи на странице (или смещение от начала страницы). В случае относительной адресации перемещение записи приведёт к изменению КБД и необходимости корректировки индексов, если они есть.

ПРИМЕР КОСВЕННОЙ АДРЕСАЦИИ



Часть адресного пространства страницы выделяется под индекс страницы. Число статей (слотов) в нём одинаково для всех страниц. В качестве КБД записи выступает совокупность номера нужной страницы и номера требуемого слота в индексе этой страницы (значения ***N, i***). В ***i***-м слоте на ***N***-й странице хранится собственно адрес записи (смещение от начала страницы).

ОСНОВНЫЕ ФИЗИЧЕСКИЕ СТРУКТУРЫ SQL SERVER

Файлы – это файлы операционной системы, выделенные для хранения базы данных.
Табличная область (TABLESPACE, табличное пространство) – область памяти, предназначенная для хранения всех объектов БД. Табличная область имеет имя и занимает один или более файлов операционной системы. Создается командой CREATE TABLESPACE.
Сегмент (SEGMENT) – область памяти, занимаемая данными одного объекта БД. Имя совпадает с именем объекта.
Экстент (EXTENT) – непрерывная область памяти, относящаяся к одному сегменту.
Блок (BLOCK) – область памяти, которая считывается и записывается на диск за одно физическое чтение.

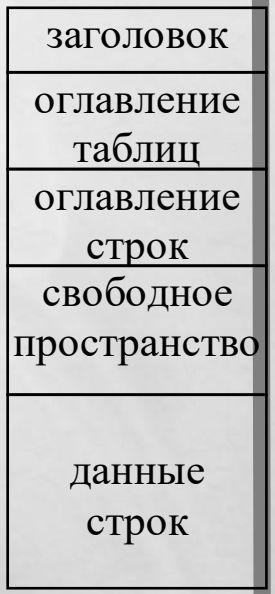
db_block_size – размер блока

ФОРМАТ БЛОКА ДАННЫХ SQL SERVER

- **ЗАГОЛОВОК** (ОБЩИЙ И ПЕРЕМЕННЫЙ) СОДЕРЖИТ ОБЩУЮ ИНФОРМАЦИЮ БЛОКА - АДРЕС БЛОКА И ТИП СЕГМЕНТА (СЕГМЕНТ ДАННЫХ, СЕГМЕНТ ИНДЕКСА ИЛИ СЕГМЕНТ ОТКАТА).
- **ОГЛАВЛЕНИЕ ТАБЛИЦ** - ЧАСТЬ БЛОКА С ИНФОРМАЦИЕЙ О ТОМ, КАКИЕ ТАБЛИЦЫ ИМЕЮТ СТРОКИ В ЭТОМ БЛОКЕ.
- **ОГЛАВЛЕНИЕ СТРОК** - ЭТА ЧАСТЬ БЛОКА СОДЕРЖИТ ИНФОРМАЦИЮ О ДЕЙСТВИТЕЛЬНЫХ СТРОКАХ В БЛОКЕ (ВКЛЮЧАЯ АДРЕСА КАЖДОЙ ПОРЦИИ СТРОКИ В ОБЛАСТИ ДАННЫХ СТРОК).

  ПОСЛЕ ТОГО, КАК В ОГЛАВЛЕНИИ СТРОК РАСПРЕДЕЛЕНО ПРОСТРАНСТВО, ЭТО ПРОСТРАНСТВО НЕ ОСВОБОЖДАЕТСЯ ПРИ УДАЛЕНИИ СТРОКИ. ЭТО ПРОСТРАНСТВО ИСПОЛЬЗУЕТСЯ ПОВТОРНО ЛИШЬ ТОГДА, КОГДА В БЛОК ВСТАВЛЯЮТСЯ НОВЫЕ СТРОКИ.
- **ДАННЫЕ СТРОК** - ЭТА ПОРЦИЯ БЛОКА СОДЕРЖИТ ДАННЫЕ ТАБЛИЦЫ ИЛИ ИНДЕКСА.
- **СВОБОДНОЕ ПРОСТРАНСТВО** В БЛОКЕ ИСПОЛЬЗУЕТСЯ ДЛЯ ВСТАВКИ НОВЫХ СТРОК И ДЛЯ ОБНОВЛЕНИЙ СТРОК, ТРЕБУЮЩИХ ДОПОЛНИТЕЛЬНОГО ПРОСТРАНСТВА.

Структура блока



ОСНОВНЫЕ ОБЪЕКТЫ SQL SERVER

- **КЛАСТЕР** (CLUSTER) – ОБЪЕКТ, ЗАДАЮЩИЙ СПОСОБ СОВМЕСТНОГО ХРАНЕНИЯ ДАННЫХ НЕСКОЛЬКИХ ТАБЛИЦ, СОДЕРЖАЩИХ ИНФОРМАЦИЮ, ОБЫЧНО ОБРАБАТЫВАЕМУЮ СОВМЕСТНО. СОЗДАЕТСЯ КОМАНДОЙ CREATE CLUSTER. ВКЛЮЧАЕТ ТАБЛИЦЫ С ДАННЫМИ.
- **ТАБЛИЦА** (TABLE) ЯВЛЯЕТСЯ БАЗОВОЙ СТРУКТУРОЙ РЕЛЯЦИОННОЙ МОДЕЛИ. ТАБЛИЦА МОЖЕТ БЫТЬ ПУСТОЙ ИЛИ СОСТОЯТЬ ИЗ ОДНОЙ ИЛИ БОЛЕЕ СТРОК ЗНАЧЕНИЙ АТРИБУТОВ. СОЗДАЕТСЯ КОМАНДОЙ CREATE TABLE, МОЖЕТ БЫТЬ СОЗДАНА В КЛАСТЕРЕ.
- **ИНДЕКС** (INDEX) – ЭТО ОБЪЕКТ БАЗЫ ДАННЫХ, СОЗДАВАЕМЫЙ ДЛЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ВЫБОРКИ ДАННЫХ. ИНДЕКС СОЗДАЕТСЯ ДЛЯ СТОЛБЦА (СТОЛБЦОВ) ТАБЛИЦЫ И ОБЕСПЕЧИВАЕТ БОЛЕЕ БЫСТРЫЙ ДОСТУП К ДАННЫМ ЭТОЙ ТАБЛИЦЫ ЗА СЧЕТ УПОРЯДОЧЕНИЯ ДАННЫХ СТОЛБЦА (СТОЛБЦОВ) ПО ЗНАЧЕНИЮ. СОЗДАЕТСЯ КОМАНДОЙ CREATE INDEX.

Кластеры, таблицы и индексы называются *объектами, занимающими память*, т.к. в них хранятся фактографические данные.