Раздел 3. Проектирование баз данных

Тема 3.3.

Построение СУБД - ориентированных моделей

Вопросы лекции:

1. **Особенности этапа внутреннего даталогического проектирования БД.**
2. Содержание внутреннего даталогического проектирования.
3. Физическое проектирование БД.

Как и было сказано в определении внутреннего даталогического уровня моделирования, основной его целью является **детализация** инфологической и концептуальной даталогической моделей БД и **превращение её во внутреннюю схему БД**, на которой ранее выявленные сущности, атрибуты и связи инфологической модели оформляются согласно правилам моделирования **для выбранного вида базы данных** (часто даже с учётом конкретной версии СУБД).

На этом этапе формируется описание модели данных без конкретизации их физического размещения на носителях в соответствии с возможностями данной СУБД.

Описание внутренних даталогических моделей проводится в терминах СУБД и возможно с соблюдением синтаксиса ее языка.

На этом этапе проектирования не прекращается анализ предметной области (скорее всего, появятся новые вопросы, ответы на которые вынудят нас внести изменения в инфологическую модель), но особое внимание на данном уровне стоит уделить уже рассмотренным ранее требованиям, предъявляемым к любой базе данных (адекватность, целостность, минимизация избыточности, производительность и т.д.).

Поскольку именно на даталогическом уровне формируется будущая реальная структура базы данных, все **принятые здесь решения** (и, увы, допущенные ошибки) **будут очень сильно влиять** на степень адекватности базы данных предметной области, на удобство использования базы данных, на её производительность и защищённость её данных.

Перед тем, как создавать внутреннюю даталогическую модель, необходимо ещё раз проанализировать имеющуюся инфологическую модель: на этот раз – с точки зрения «как реализовать это в виде БД»:

- Какие таблицы создать? Почему именно такие? Может быть, есть лучший вариант?
- Какие у этих таблиц будут поля (а также какие у этих полей будут типы данных и иные свойства)?
- Какие ключи использовать?
- Какие связи и как именно устанавливать? (Особенно типа M:M !)
- Может быть, уже видна часть индексов и представлений?
- Достаточно ли наша схема базы данных **нормализована**?

На все эти и многие другие вопросы придётся искать ответы в процессе проектирования базы данных на концептуальном и внутреннем даталогических уровнях.

Что необходимо знать и уметь для того, чтобы успешно проектировать базы данных на **даталогическом уровне**.

В идеале	Как минимум
Глубоко понимать реляционную теорию.	Иметь представление о реляционной теории.
Знать «на уровне инстинктов» теорию нормализации.	Понимать хотя бы 1-3 нормальные формы.
Глубоко знать SQL.	Знать основы SQL.
Знать целевую СУБД на уровне, позволяющем проводить её тонкое администрирование.	Уметь устанавливать и настраивать целевую СУБД.
Уметь использовать средства проектирования.	Иметь много терпения для изучения средств проектирования.

Сложнее всего будет с последними двумя пунктами — и СУБД, и средства проектирования развиваются слишком стремительно, чтобы по ним существовали некие «фундаментальные» книги (они будут устаревать уже к моменту публикации), потому самое надёжное решение — читать официальную документацию по конкретной версии конкретной СУБД и конкретного средства проектирования, которые вы используете.

Проектирование на инфологическом уровне предполагало интенсивное **обсуждение формируемой модели с заказчиком**, и потому там применялись такие средства, которые были бы доступны и понятны «не техническому» человеку (который, например, может быть глубочайшим экспертом в международной логистике или иной предметной области, но совершенно не обязан понимать технические тонкости работы баз данных).

Начиная с даталогического уровня, мы уже в куда большей степени **ориентируемся на технических специалистов**, поэтому используемые здесь решения могут быть непонятны заказчику — **и это нормально**, т.к. можно вносить необходимые правки в инфологическую модель и продолжать говорить с заказчиком «на его языке», не перегружая его техническими подробностями.

И **первое, что стоит сделать** техническим специалистам — это прийти к нескольким ключевым соглашениям, которые в общем случае можно поделить на две группы:

- Соглашения относительно **СУБД**.
- Соглашения относительно **БД**.

Соглашения относительно СУБД могут включать в себя, например, следующие пункты:

- Вид СУБД (на какой тип модели данных ориентирована).
- Конкретный продукт (конкретная СУБД определенной фирмы).
- Минимальная поддерживаемая версия выбранной конкретной СУБД.
- Инфраструктурные особенности выбранной конкретной СУБД.

Например: Соглашения относительно СУБД:

- Вид СУБД: реляционная.
- Конкретный продукт (конкретная СУБД): MySQL.
- Минимальная поддерживаемая версия выбранной конкретной СУБД: 8.0 и новее (т.к. более ранние версии имеют ряд технических ограничений).
- Инфраструктурные особенности выбранной конкретной СУБД: отдельный сервер.

В этом списке **пункты приведены в порядке убываниях их «судьбоносности»** (так, например, переход в середине проекта с реляционной СУБД на иерархическую фактически будет означать начало проекта с нуля; а вот перенос нескольких серверов в облако может пройти почти безболезненно).

Поскольку выбор конкретной СУБД и её версии очень сильно влияет на дальнейшие технические решения, с этими вопросами тоже стоит разобраться до того, как на даталогическом уровне будет сделано много работы.

Если придётся переделывать модель под другую версию выбранной СУБД или даже под другую СУБД, гарантированно потребуется очень много рутинной работы, а также, возможно, и интеллектуальной — если придётся искать новые технологические решения взамен уже созданным.

Соглашения относительно БД:

- Соглашения об именовании структур.
- Соглашения об используемых типах данных.
- Соглашения о поддерживаемых видах связей.
- Соглашения об оформлении SQL-кода.
- Соглашения о комментариях.
- Иные специфические соглашения, зависящие от проекта (например, соглашения об API в виде представлений и хранимых подпрограмм – процедур, триггеров, пакетов и т.п.).

К сожалению, без учета подобных соглашений для СУБД и к БД практически невозможно сформировать даталогическую модель, способную к ее интеграции в СУБД *(автоматической генерации кода для реализации в виде реальной БД)*.

Например: Соглашения относительно БД:

Соглашения об именовании структур:

- При именовании таблиц и полей используется только нижний регистр.
- Слова в именах таблиц и полей отделены друг от друга символом «_».
- В именах таблиц существительные используются в единственном числе (например, «file», а не «files»). В именах полей множественное число допускается, но не рекомендуется.
- Имена полей начинаются с префиксов из первых букв имён таблицы.
- Имена ограничений начинаются с префикса «UNQ_» и содержат имена всех входящих в ограничение полей.
- Имена триггеров начинаются с префикса «TRG_» и содержат в себе имя таблицы и название момента срабатывания.

Соглашения об оформлении SQL-кода:

- Все ключевые слова языка SQL пишутся в верхнем регистре.
- Все имена структур БД обязательно должны быть заключены в обратные апострофы, т.е. символы «`».

Соглашения о комментариях:

- Комментарии оформляются для всех структур БД.

Иные специфические соглашения, зависящие от проекта:

- Все поля, содержащие дату и время, имеют тип INTEGER и хранят UNIXTIME-значения.
- Все поля, содержащие только дату, имеют тип DATE.
- Все первичные ключи — искусственные, автоинкрементируемые, беззнаковые.

С CASE-инструментами проектирования на даталогическом уровне дела обстоят неоднозначно, в отличие от инфологического.

Для даталогического уровня нужны специализированные инструменты, позволяющие оптимальным образом формировать структуру базы данных, многократно её пересматривать и анализировать, а в конечном итоге и **экспортировать в полноценный SQL-код** для импорта в СУБД и создания БД.

Каждый производитель СУБД традиционно предоставляет свои собственные инструменты, например:

- MySQL Workbench для MySQL.
- SQL Server Management Studio для MS SQL Server.
- SQL Developer Data Modeler для Oracle.
- И так далее — таких инструментов очень много.

Это, безусловно, очень мощные инструменты, созданные с учётом всех особенностей целевой СУБД, но созданные в них даталогические модели БД трудно адаптировать под другие СУБД.

На даталогическом этапе практичнее применять универсальные CASE-средства типа ERWin Data Modeler, Sparx Systems Enterprise Architect, DbSchema и т.п.

Преимущества «универсальных» CASE-инструментов:

- Ни один из них не работает напрямую с физической базой данных (что сводит к нулю шансы однажды уничтожить уже работающую базу данных заказчика).
- Эти инструменты поддерживают синтаксис и особенности многих СУБД, что позволяет сравнивать принимаемые решения для разных СУБД, переходить (**конвертировать модели**) с одной СУБД на другую или даже работать в таких необычных ситуациях, когда часть базы данных работает под управлением одной СУБД, а часть — под управлением другой СУБД.
- Создатели этих инструментов учли многие недостатки «узкоспециализированных» аналогов и постарались их исправить.
- Такие инструменты, как правило, обладают гораздо более дружественным интерфейсом, более развитыми средствами совместной (командной) работы и многими иными конкурентными преимуществами.

Раздел 3. Проектирование баз данных

Тема 3.3.

Построение СУБД - ориентированных моделей

Вопросы лекции:

1. Особенности этапа внутреннего даталогического проектирования БД.
2. **Содержание внутреннего даталогического проектирования.**
3. Физическое проектирование БД.

Главное отличие даталогических моделей БД от инфологической состоит в том, что инфологическая модель БД хранит в себе всю информацию о предметной области, необходимую и достаточную для проектирования базы данных, но она **не привязана** к определенной СУБД.

Даталогические модели БД может не отражать в явном виде все сущности, зафиксированные в инфологической модели, но они должны быть непременно привязаны хотя бы к **типу СУБД**, на которой разрабатывается база данных **по поддерживаемой модели данных**.

При проектировании даталогической модели данных должно быть обеспечено **однозначное соответствие** между конструкциями **языка описания данных** и графическими обозначениями информационных единиц и связей между ними.

Спроектировать внутреннюю даталогическую структуру базы данных означает определить все информационные единицы и связи между ними, задать их имена в соответствии с **соглашениями**.

Таким образом внутреннее даталогическое проектирование сводится к следующим **этапам (действиям)**:

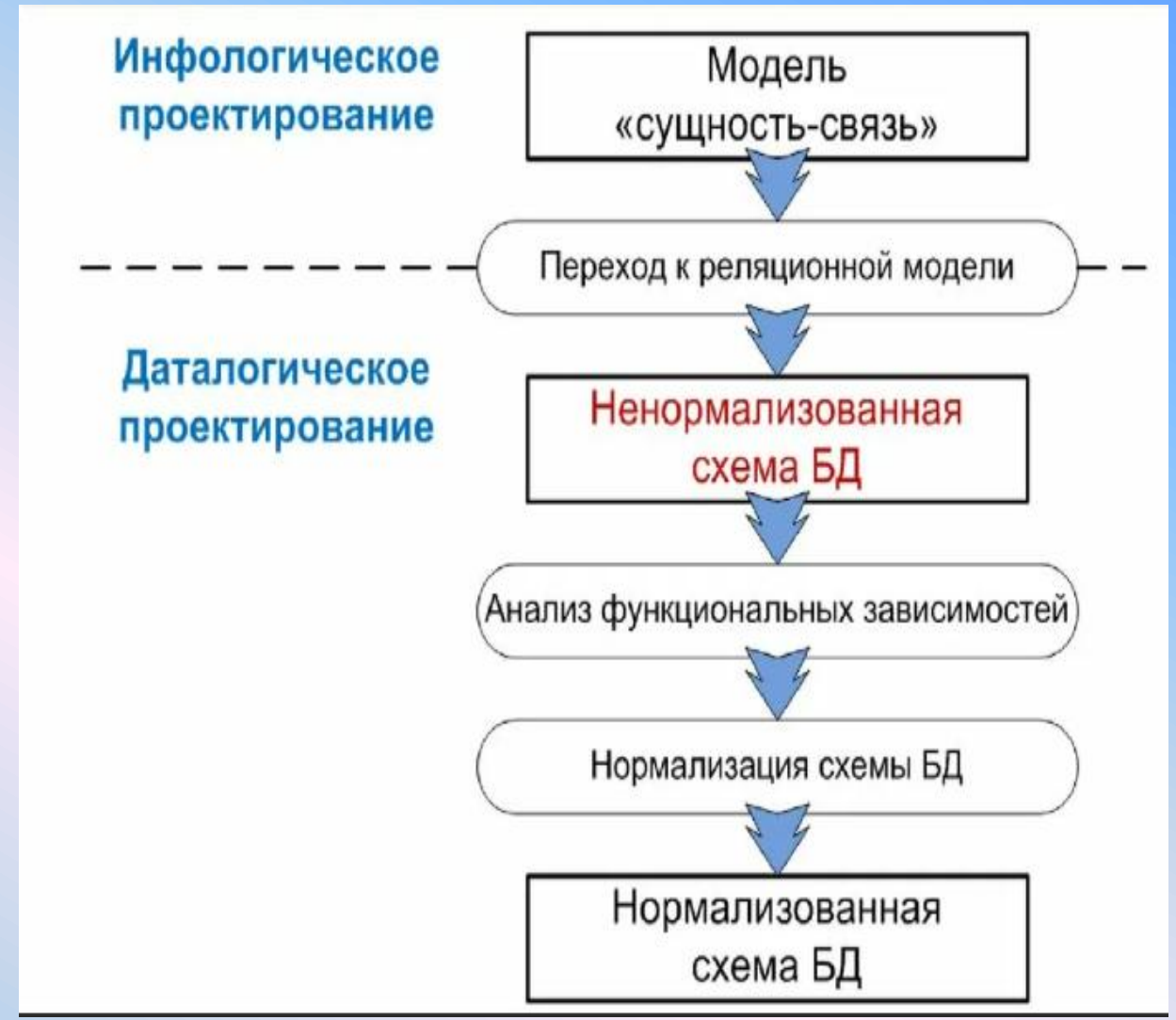
1. Определение таблиц и их однозначное именование по соглашениям БД.
2. Определение полей таблиц и их однозначное именование по соглашениям БД.
3. Определение типов данных и **кодировок** в соответствии с выбранной СУБД.
4. Определение длины каждого поля таблиц.
5. Определение обязательности каждого поля (допускаются ли NULL-значения).
6. Определение индексации каждого поля (создание дополнительных индексов).
7. Определение связей таблиц по соглашениям БД (с возможным их преобразованием).

Особое внимание при построении модели уделяют целостности и отсутствию избыточности данных. Избыточность - это многократное повторение одних и тех же данных. Для этого применяется **нормализация таблиц** исходной БД.

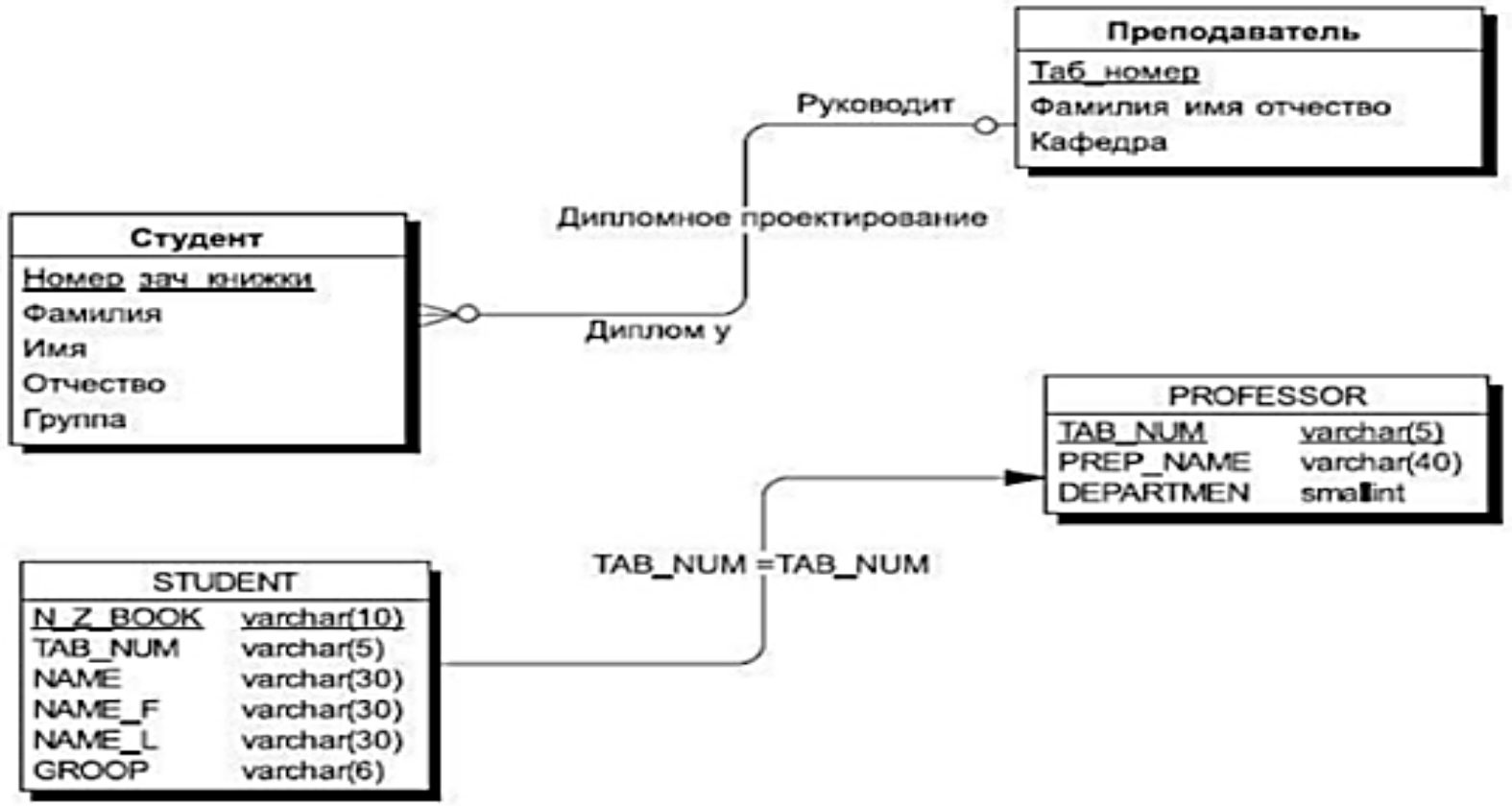
Кроме целостности и не избыточности при проектировании БД учитывают возможность быстрого выполнения запросов к БД и оптимального использования ресурсов, а также разграничение доступа для разных групп пользователей.

Переход от инфологической модели к **даталогической**

Проектирование БД на даталогическом уровне



Переход от инфологической модели к **даталогической**



Уже начиная с этого момента, рекомендуется периодически генерировать SQL-код и импортировать получившийся результат в СУБД, чтобы избежать различных досадных ошибок в конце.

Часто в CASE-средствах этап перехода от концептуального проектирования (логическая модель) к внутреннему даталогическому проектированию БД называют – «Физическая модель БД».

Это **не вполне корректно**. Физический уровень БД – структуры хранения информации БД на внешних носителях в файлах БД определенной структуры.

Раздел 3. Проектирование баз данных

Тема 3.3.

Построение СУБД - ориентированных моделей

Вопросы лекции:

1. Особенности этапа внутреннего даталогического проектирования БД.
2. Содержание внутреннего даталогического проектирования.
3. **Физическое проектирование БД.**

Общие задачи проектирования **физического уровня** баз данных

(этап физического проектирования БД)

Данный этап является частично условным, так как многие его задачи **решены производителем** программного комплекса СУБД.

- выбор типа носителя, способа организации данных, методов доступа (**определение пользователей базы данных, их уровней доступа, разработка и внедрение правил безопасности доступа**),
- определение размеров физического блока, управление размещением данных на внешнем носителе,
- управление свободной памятью, определение целесообразности сжатия данных и используемых методов сжатия,

Общие задачи **физического проектирования** баз данных

- оценка размеров объектов базы (определение размеров табличных пространств и особенностей их размещения на носителях информации,
- определение спецификации носителей информации для промышленной системы (например, тип RAID-массивов, их количество),

RAID-массивы это высокопроизводительные , устойчивые к отказам подсистемы ввода-выводы, это технология для расширения пропускной способности системы ввода/вывода и обеспечения возможности хранения избыточных данных.

- разработка топологии базы данных в случае распределенной базы данных, определение механизмов доступа к удаленным данным.

Общие задачи **физического проектирования** баз данных

- Определение требований к дисковой памяти.
- Разработка пользовательских представлений.
- Анализ необходимости введения контролируемой избыточности.
- Организация мониторинга и настройка функционирования ОС.
- Разработка средств и механизмов защиты.

Для функционирования СУБД во внешней памяти базы данных возникает необходимость хранить следующие разновидности объектов:

- строки отношений - основная часть базы данных, большей частью непосредственно видимая пользователям;
- управляющие структуры - индексы, создаваемые по инициативе пользователя (администратора) или верхнего уровня системы из соображений повышения эффективности выполнения запросов и обычно автоматически поддерживаемые нижним уровнем системы;
- журнальную информацию, поддерживаемую для удовлетворения потребности в падежном хранении данных;
- служебную информацию, поддерживаемую для удовлетворения внутренних потребностей нижнего уровня системы.

Физическое проектирование БД – определение **метода доступа**

Любое упорядочение данных на диске называется структурой хранения.

К сожалению, до сих пор не создана такая идеальная структура хранения, которую можно было бы использовать как оптимальную для любых задач.

Но зато можно организовать самые разные структуры хранения, обладающие различной производительностью, область применения которых определяется тем, насколько они эффективны в том или ином случае.

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются **на физическом уровне**.

Исторически первыми системами хранения и доступа были **файловые структуры и системы управления файлами (СУФ)**, которые фактически являлись частью операционных систем.

СУБД создавала над этими файловыми моделями **свою надстройку**, которая позволяла организовать всю совокупность файлов таким образом, чтобы она работала как единое целое и получала централизованное управление от СУБД.

Однако непосредственный доступ осуществлялся **на уровне файловых команд ОС**, которые СУБД использовала при манипулировании всеми файлами, составляющими хранимые данные одной или нескольких баз данных.

Однако механизмы буферизации и управления файловыми структурами ОС не приспособлены для решения задач собственно СУБД, эти механизмы разрабатывались просто для традиционной обработки файлов, и с ростом объемов хранимых данных они стали неэффективными для использования СУБД.

Тогда постепенно произошел переход от базовых файловых структур к **непосредственному управлению** размещением данных на внешних носителях **самой СУБД**.

И пространство внешней памяти уже выходило из-под владения СУФ ОС и управлялось **непосредственно СУБД**.

При этом механизмы, применяемые в файловых системах, перешли во многом и в новые системы организации данных во внешней памяти, называемые чаще **страничными системами хранения информации**.

В результате:

В настоящее время используются следующие способы физической организации данных в БД (хранения и поиска данных на дисках системы и организации доступа к ним):

- файловые структуры физической организации данных (через обращения к ОС и к ее файловой системе);

- страничные структуры физической организации данных (через реализацию **прямого доступа к дискам** без участия файловых систем).

При страничной организации поиск и предоставление данных пользователю осуществляются с помощью нескольких **программ доступа** данных и включают в себя три основных этапа.

- Определяется **искомая запись**, для извлечения которой запрашивается диспетчер **файлов**.
- Диспетчер файлов определяет **страницу**, на которой находится искомая запись, и для извлечения этой страницы запрашивает диспетчер **дисков**.
- Диспетчер дисков определяет физическое положение искомой страницы на диске и посылает соответствующий запрос на ввод-вывод данных к носителю информации.

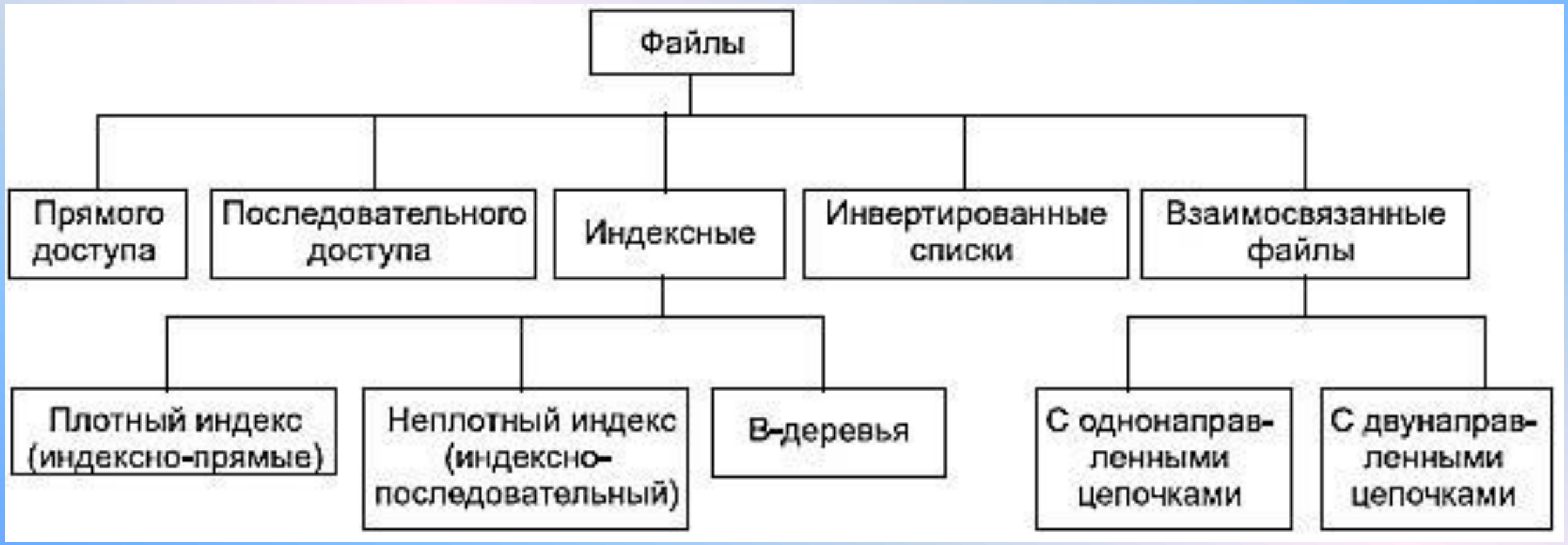
Следует отметить, что организация хранения данных предполагает использование технологии **кластеризации данных**, представляющей собой важнейшее ***условие высокой производительности***.

Эта технология обеспечивает как можно **более близкое** физическое размещение на диске логически связанных между собой и часто используемых данных.

Выделяют **внутрифайловую** и **межфайловую кластеризацию**.

В современных СУБД администратором БД в соответствии с требованиями, предъявляемыми к производительности СУБД, **могут устанавливаться** нескольких различных **типов кластеризации** данных для разных файлов.

В каждой СУБД по-разному организованы хранение и доступ к данным, однако существуют некоторые файловые структуры, которые имеют общепринятые способы организации и широко применяются практически во всех СУБД.



Классификация файлов, используемых в системах баз данных

Спасибо за внимание !