# Эволюция баз данных

В истории вычислительной техники можно проследить два основных направления её использования:

1)Численные расчёты

2)Информационные системы (базы данных).

Информационная система - совокупность средств (информационных, технических, программных, организационных), методов и персонала, предназначенных для ввода, хранения, обработки и выдачи информации в интересах поставленной задачи.

Примеры ИС: банковское дело, продажа билетов, управление предприятием.

В первых ИС информация хранилась в виде обычных линейных файлов на магнитных лентах (устройство с последовательным доступом).

Можно предположить, что требования ИС вызвали появление более быстрых устройств хранения информации - накопителей на жестких магнитных дисках (устройств с прямым доступом).

Файлы и файловые системы

Файл - именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.

Файловая система - распределяет внешнюю память таким образом, что отображение имен файлов соответствует адресу во внешней памяти и обеспечивает доступ к данным.

Данные - зарегистрированная информация; представление фактов, понятий или инструкций в форме, приемлемой для общения, интерпретации, или обработки человеком или с помощью автоматических средств.

База данных (Data Base) - поименованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Предметная область - знаний или человеческой деятельности, для которой создается база данных.

Система управления базами данных (СУБД); Data Base Management System (DBMS) - совокупность языковых и программных средств, предназначенных для создания, введения и совместного использования баз данных несколькими пользователями.

В истории развития баз данных выделяют 5 этапов:

1)Базы данных на больших ЭВМ

Первый этап развития связан с организацией баз данных на больших ЭВМ. В это время появились иерархические и сетевые модели. 1968 год - разработка иерархической СУБД Information Management System (IMS). => 1971 год - появление первой сетевой СУБД CODASYL.

2)Эпоха ПК

Второй этап развития связан с развитием ПК. Компьютеры стали ближе и доступнее каждому пользователю. В 1981 году Э.Ф. Кодд получил за создание реляционной модели и реляционной алгебры престижную премию Тьюринга Американской ассоциации по вычислительной технике.

3)Распределённые базы данных

Третий этап развития связан с созданием распределённых баз данных. С развитием вычислительной техники появились распределённые базы данных, позволяющие организовать параллельную обработку информации и поддержку целостности БД.

4)Перспективы развития СУБД

Этот этап характеризуется появлением новой технологии доступа к данным - интернет. Основное отличие этого подхода от технологии клиент-сервер состоит в том, что отпадает необходимость использования специализированного клиентского программного обеспечения. Для работы с удаленной базой данных используется стандартный интернет-браузер.

5)Появление NoSql

NoSQL > Not Only SQL

Причины появления NoSql:

1)Реляционные базы данных не могут справляться с нагрузками актуальными в наше время.

2)Увеличение количества компаний, связанных с использованием больших объемов данных (от нескольких до сотен терабайт).

Преимущества и недостатки СУБД:

1) Контроль за избыточностью данных;

2) Непротиворечивость данных;

3) Больше полезной информации при том же объеме хранимых данных;

4) Совместное использование данных;

5) Поддержка целостности данных;

6) Повышенная безопасность;

7) Применение стандартов;

8) Повышение эффективности с ростом масштабов системы;

9) Возможность нахождения компромисса при противоречивых требованиях;

10) Повышение доступности данных и их готовности к работе;

11) Улучшение показателей производимости;

12) Упрощение сопровождения системы за счет независимости от данных;

13) Улучшение управления параллельной работой;

14) Развитые службы резервного копирования и восстановления.

Недостатки СУБД:

1)Сложность;

2)Размер;

3)Стоимость СУБД;

4)Дополнительные затраты на аппаратное обеспечение;

5)Затраты на преобразование;

6)Производительность;

7)Более серьёзные последствия при выходе системы из строя.

# Архитектура удаленных баз данных

Классификация баз данных:

1)По технологии обработки данных

базы данных подразделяются на:

1. Централизованные (централизованные базы данных находятся на 1 машине вычислительной сети);
2. Распределенные (распределенные базы дынных состоят из нескольких частей, возможно, дублирующих друг друга, которые находятся на нескольких компьютерах вычислительной сети.

Сервер - компьютер, предоставляющий ресурс.

Клиент - компьютер, пользующийся этим ресурсом.

2)По способу доступа к данным

базы данных подразделяются:

1. С локальным доступом;
2. С удаленным доступом.

Существует 2 архитектуры удаления баз данных:

1. Файл-сервер;
2. Клиент-сервер.

Файл-сервер - выделенный сервер, оптимизированный для выполнения файловых операций ввода-вывода.

Предназначен для хранения файлов любого типа. Как правило, обладает большим объемом дискового пространства и оборудован RAID контроллером для обеспечения быстрой записи и чтения данных.

Файл-серверные приложения - приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения данных.

Функции сервера: хранения данных и кода программы.

Функции клиента: обработка данных происходит исключительно на стороне клиента.

Преимущества и недостатки файл-сервера

Плюсы:

1. Низкая стоимость разработки;
2. Высокая скорость разработки;
3. Невысокая стоимость обновления и изменения ПО;
4. Многопользовательность.

Минусы:

1. Низкая производительность;
2. Ненадежная система;
3. Слабая защита данных (средствами ОС);
4. Высокий сетевой трафик.

Клиент-сервер - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами.

Клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и ПО.

Многоуровневая архитектура клиент-сервер - разновидностьархитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов.

Частные случаи многоуровневой архитектуры: трехуровневая архитектура.

Преимущества и недостатки клиент сервера

Плюсы:

1. Распределение функций вычислительной системы между несколькими независимыми компьютерами в сети;
2. Все данные хранятся на сервере, который защищен лучше большинства клиентов;
3. Высокая безопасность;
4. Высокая надежность.

Минусы:

1. Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;
2. Поддержка работы данной системы требует отдельного специалиста - системного администратора;
3. Высокая стоимость оборудования.

Трехуровневая архитектура

Трехуровневая архитектура, синоним трехзвенная архитектура предполагает наличие следующих компонентов приложения: клиентское, подключенное к серверу приложений, который в свою очередь подключен к серверу базы данных.

Первый уровень: клиент (компонент представления);

Второй уровень: сервер приложения (прикладной компонент);

Третий уровень: сервер базы данных (компонент доступа к ресурсам);

База данных

Достоинства:

1. Независимость уровней;
2. Легкость в поддержке;
3. Отдельные компоненты можно использовать в других задачах;
4. Задача разработки хорошо делится и поэтому может быть быстрее решена (уровни можно разрабатывать параллельно);
5. Web дизайнер делает уровень представления.
6. Инженер (Software Engineer) делает логику
7. Администратор БД делает модель данных
8. Клиентское ПО не нуждается в администрировании;
9. Масштабируемость;
10. Конфигурируемость:
11. Достаточно высокая безопасность и надежность;
12. Низкие требования к скорости канала между терминалами и сервером приложений;
13. Низкие требования к производительности и техническим характеристикам терминалов.

Недостатки:

1. Сложность администрирования и обслуживания;
2. Более высокая сложность создания приложений;
3. Высокие требования к производительности серверов приложений и сервера базы данных;
4. Высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

# Модели данных

Модель данных - совокупность структур данных, операций их обработки и ограничения целостности, накладываемых на данные.

Модели данных делятся на: иерархическую, сетевую, реляционную.

Иерархическая модель

Одной из наиболее популярных иерархических СУБД была Information Management System (IMS) разработанная в 1968 фирмой IBM.

Появление иерархических СУБД было обусловлено необходимостью решения задачи разузлования.

Главный тип именуется исходным типом, а подчиненные - порожденными. У подчиненных типов могут быть в свою очередь подчиненные типы.

Задача разделения какой-то главной части на составные части называется разузлованием.

Достоинства:

1. Простота освоения и использования;
2. Высокое быстродействие.

Недостатки:

1. Невозможность связи “многие-ко-многим”;
2. Сложность понимания структуры при большом объеме данных для пользователя;
3. Зависимость программ от данных.

Сетевая модель

В структуре сетевой модели понятия “корень”, “узел” и указатель идентичны иерархической модели.

Сетевая модель схожа с иерархической моделью, но в отличие от неё соседние узлы имеют связь “многие-ко-многим”.

Появление сетевых СУБД было обусловлено необходимостью решения задачи автоматизированного обслуживания заказов.

Достоинства:

1. Высокое быстродействие;
2. Гибкость;
3. Универсальность;
4. Возможность доступа к данным через значения нескольких отношений.

Недостатки:

1. Зависимость программ от данных;
2. Сложная структура.

Реляционная модель

Создателем реляционной модели является сотрудник фирмы IBM доктор Э. Ф. Кодд в 1970г.

В реляционной модели таблица описывает некий информационный объект: люди, предметы и т.д.

|  |  |
| --- | --- |
| **Реляционный термин** | **Определение** |
| База данных | Набор объектов |
| Схема базы данных | Набор таблиц и связей между ними |
| Отношение | Таблица с определёнными свойствами |
| Схема отношения | Совокупность заголовков столбцов |
| Тело отношения | Совокупность записей |
| Атрибут отношения | Наименование столбца таблицы |
| Кортеж отношения | Строка, запись |
| Степень отношения | Количество столбцов таблицы |
| Мощность отношения | Количество строк таблицы |
| Домен | Совокупность допустимых значений атрибута |

Первичный ключ - минимальный набор атрибутов, который единственным образом идентифицирует кортеж данного отношения.

Первичный ключ бывает простой (состоит из одного атрибута) и составной (состоит из нескольких атрибутов).

Свойства первичного ключа:

1) Уникальность - в каждом кортеже отношения R значение ключа K единственным образом идентифицирует данный кортеж.

2) Минимальность - никакое допустимое подмножество ключа К не обладает свойством уникальности.

Внешний ключ - атрибут или множество атрибутов внутри отношения, которое соответствует первичному ключу некоторого отношения.

Свойства отношения:

1. Отношение имеет имя, которое отличается от имен всех других отношений в реляционной схеме.
2. Каждая ячейка отношения содержит только одно элементарное (неделимое) значение.
3. Каждый атрибут имеет уникальное имя.
4. Каждый кортеж является уникальным, т.е. дубликатов кортежей быть не может.
5. Порядок следования атрибутов не имеет значения.
6. Теоретически порядок следования кортежей в отношении не имеет значения.

Достоинства:

1. Простота и доступность понимания конечным пользователем;
2. При проектировании реляционной БД применяются строгие правила, базирующие на математическом аппарате;
3. Полная независимость программ от данных;
4. Для построения запросов и написания прикладных программ нет необходимости знания конкретной организации БД во внешней памяти.

Недостатки:

1. Относительно низкая скорость доступа и большой объем внешней памяти.
2. Трудность понимания структуры данных из-за появления большого кол-ва таблиц в результате логического проектирования.
3. Далеко не всегда предметную область можно представить в виде совокупности таблиц.

# Операции над отношениями

Кодд предложил не только реляционную модель, но и реляционные языки для обработки работы с таблицами алгебры и реляционного исчисления.

Операции:

1. Объединение (union)
2. Пересечение (intersect)
3. Разность (except)
4. Декартово произведение (cartesian product)
5. Выборка (select)
6. Проекция (project)
7. Деление (division)
8. Соединение (join)

# Типы связей в реляционной модели

В реляционной модели существуют такие связи, как:

1. Один к одному (1:1)
2. Один ко многим (1:М)
3. Многие ко многим (М:М)

Одной строке таблицы соответствует не более одной строки в другой таблице - связь 1:1;

Одной строке таблицы соответствует 0,1 или более строк в другой таблице - связь 1:М;

Одна строка первой таблицы связана с 0,1 или более строками второй таблицы и наоборот - М:М.

Определение типа связей:

1. Выбрать главную таблицу (в ней должен быть первичный ключ);
2. Выбрать зависимую таблицу (в ней должен быть внешний ключ);
3. Выбрать одну запись из главной таблицы и посмотреть, сколько потенциально ей может соответствовать записей в зависимой таблице;
4. Если не более 1, то связь 1:1;
5. Если 0,1 и более, то связь 1:М;
6. Связь М:М реализуется через промежуточную таблицу.

# Целостность данных

Типы соединений между таблицами:

1. Симметричное
2. Левое внешнее
3. Правое внешнее

Целостность данных - непротиворечивость данных.

Целостность данных:

1. Категорная (никакой первичный ключ не может содержать пустого значения);
2. Доменная (каждый атрибут определён на своём домене);
3. Ссылочная

Правило ссылочной целостности: значение внешнего ключа должно соответствовать значению первичного ключа или должно быть пустым.

Каскадное обновление связанных записей (при изменении значений первичного ключа произойдёт изменение внешнего ключа в соответствующих записях зависимой таблицы).

Каскадное удаление связанных записей (при удалении записей из главной таблицы будут удалены из зависимой таблицы все записи с внешним ключом, равным первичному ключу в главной таблице).

# Этапы проектирования баз данных

Жизненный цикл базы данных - процесс проектирования, реализации и управления базой данных.

Анализ:

1. Планирование разработки;
2. Определение требований;
3. Сбор и анализ требований.

Проектирование:

1. Концептуальная модель;
2. Логическая модель;
3. Физическая модель.

Реализация

Тестирование

Эксплуатация и сопровождение.

Планирование разработки:

1. Объем работ;
2. Требуемые ресурсы;
3. Сроки разработки;
4. Стоимость проекта.

Решаются вопросы:

1. Технологической осуществимости;
2. Операционной осуществимости;
3. Экономической целесообразности.

Определение требований к системе

Цели информационной системы: определение архитектуры, определение требований к техническому и программному обеспечению, информационные потребности отделов, кол-во пользователей.

Сбор и анализ требований

Определение потребностей пользователей, на основании которых составляется спецификация требований пользователей.

Концептуальное проектирование базы данных - процесс создания модели используемой на предприятии информации без физических аспектов её представления.

В результате изучения предметной области определяются информационные объекты, их атрибуты, связи между ними, ключевые поля.

ER - модель Чена

Эта модель была разработана Ченом в 1976 году с целью упрощения концептуального проектирования баз данных.

Сущность представляет собой различимое множество объектов (экземпляров сущности) реального мира с одинаковым набором атрибутов.

Полнота связи

Полнота - возможность существования в сущности экземпляра, не связанного ни с одним экземпляром другой сущности.

Кардинальные числа:

1. Нижнее кардинальное число. Отвечает за полноту связи и может иметь значения 0 или 1. Nmin = {0;1};
2. Верхнее кардинальное число. Отвечает за мощность связи. Принимает значения 1 или N. Nmax = {1;N}.

Логическое проектирование базы данных - процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов.

Для построения логической модели базы данных используются средства автоматизации проектирования (CASE средства).

Проектирование базы данных

Этапы концептуального и логического проектирования больших систем следует отделять от этапов физического проектирования. На это есть несколько причин.

1. Они связаны с совершенно разными аспектами системы, поскольку отвечают на вопрос, что делать, а не как делать.
2. Они выполняются в разное время, поскольку понять, что надо сделать, следует прежде, чем решить, как это сделать.
3. Они требуют совершенно разных навыков и опыта, поэтому требуют привлечения специалистов различного профиля.

Проектирования базы данных - итерационный процесс, который имеет свое начало, но не имеет конца и состоит из бесконечного ряда уточнений. Его следует рассматривать прежде всего как процесс познания.

Задача физического этапа проектирования – выбор рациональной структуры хранения данных и методов доступа к ним.

Физическое проектирование базы данных - процесс подготовки описания и реализации базы данных средствами конкретной СУБД.

В случае реляционной модели данных под этим подразумевается следующее:

1. Создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;
2. Определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность СУБД;
3. Разработка средств защиты создаваемой системы.

Реализация - разработка программы, позволяющей работать с БД.

Тестирование

Приложение базы данных тестируется с целью обнаружения ошибок, а также его проверки на соответствие всем требованиям, выдвинутым пользователем.

Эксплуатация и сопровождение

База данных считается полностью разработанной и реализованной. Система наблюдается и поддерживается.

Подходы к проектированию базы данных: нисходящий и восходящий.

При восходящем подходе работа начинается с самого нижнего уровня атрибутов (т.е. свойств сущностей и связей), которые на основе анализа существующих между ними связей группируются в отношения, представляющие типы сущностей и связи между ними.

Более подходящей стратегией проектирования сложных баз данных является использование нисходящего подхода. Начинается этот подход с выбора высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

Обязательные(mandatory) и необязательные (optional)отношения  
В обязательных (mandatory) отношениях каждый экземпляр одной сущности должен участвовать во взаимосвязи с другой сущностью.

В необязательной (optional) взаимосвязи любой экземпляр одной сущности может участвовать в взаимосвязи с другой сущностью, но это не обязательно.

# Нормализация отношений

Цели теории нормализации: устранение дублирования и несогласованности данных;

Виды аномалий:

1. Аномалии вставки;
2. Аномалии удаления;
3. Аномалии модификации.

Аномалии вставки

В отношении Сотрудники невозможно ввести строку о новом отделе компании, содержащую значение NULL в атрибуте ТабельныйНом

Аномалии удаления

При удалении из отношения Сотрудники строки с информацией о последнем сотруднике некоторого отдела компании, сведения об этом отделе будут полностью удалены из базы данных.

Аномалии модификации

При попытке изменения значения одного из атрибутов для некоторого отдела компании в отношении (например, названия отдела) необходимо обновить соответствующие значения в строках для всех сотрудников этого отделения. Если такой модификации будут подвергнуты не все требуемые строки отношения, база данных будет содержать противоречивые сведения.

Нормализация отношений - формальный аппарат ограничений на формирование отношений, который позволяет свести к минимуму дублирование данных, обеспечивает непротиворечивость данных, уменьшает трудозатраты на ведение баз данных.

Полная декомпозиция таблицы – это набор произвольного числа ее проекций, соединение которых идентично исходной таблице.

Процесс нормализации заключается в переходе от исходной таблицы к ее полной декомпозиции.

Первая нормальная форма

Таблица находится в 1НФ тогда и только тогда, когда в каждом ее поле (на пересечении строки и столбца) находится ровно одно значение (не более одного и не ноль значений).

Отношение называется нормализованным или приведённым к первой нормальной форме, если все его атрибуты просты, далее неделимы.

Вторая нормальная форма

Отношение находится во второй нормальной форме, если оно соответствует первой нормальной форме, и каждый не ключевой атрибут функционально зависит от первичного ключа.

Таблица находится в 2НФ тогда и только тогда, когда каждое не ключевое поле таблицы связано полной функциональной зависимостью с первичным ключом.

Третья нормальная форма

Отношение соответствует третьей нормальной форме и только тогда, когда оно находится во второй нормальной форме, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

# Организация запросов к базе данных на SQL языке

Возможности языка для работы с базами данных

Любой язык работы с базами данных должен предоставлять пользователю

следующие возможности:

1. Создавать базы данных и таблицы с полным описанием их структуры;
2. Выполнять основные операции манипулирования данными, такие как вставка, модификация и удаление данных из таблиц;
3. Выполнять простые и сложные запросы.

История языка SQL

Structured Query Language (SQL)Был разработан фирмой IBM в 1970 году как язык управления реляционными базами данных.

Команды языка SQL

DDL (Data Definition Language) - группа операторов определения данных

DML (Data Manipulation Language) - группа операторов для выборки и обновления

DCL – (Data Control Language) -группа операторов определения доступа к данным

TCL –(Transaction Control Language) - группа операторов для управления транзакциями

1.Data Definition Language (DDL)

Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы, т.е. создаем, изменяем и удаляем их.

CREATE – используется для создания объектов базы данных;

ALTER – используется для изменения объектов базы данных;

DROP – используется для удаления объектов базы данных.

2.Data Manipulation Language (DML) - группа операторов для манипуляции данными.

SELECT – осуществляет выборку данных;

INSERT – добавляет новые данные;

UPDATE – изменяет существующие данные;

DELETE – удаляет данные.

3.Data Control Language (DCL) - группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций над объектами базы данных.

Сюда входят:

GRANT – предоставляет пользователю или группе разрешения на определённые операции с объектом;

REVOKE – отзывает выданные разрешения;

DENY– задаёт запрет, имеющий приоритет над разрешением.

4.Transaction Control Language (TCL) - группа операторов для управления транзакциями

BEGIN TRANSACTION – служит для определения начала транзакции;

COMMIT TRANSACTION – применяет транзакцию;

ROLLBACK TRANSACTION – откатывает все изменения, сделанные в контексте текущей транзакции;

SAVE TRANSACTION – устанавливает промежуточную точку сохранения внутри транзакции.

Синтаксис SELECT

SELECT [предикат] список полей  
 FROM таблица1 [, ...]   
 [WHERE условия отбора]  
 [GROUP BY... ]  
 [HAVING условия отбора.. ]  
 [ORDER BY... ]

Виды операторов:

Арифметические (+, -, \*, /)

Сравнения (=, >=, <=, <>)

Логические (OR, AND, NOT)

Строковые +(конкатенация)

Вспомогательные (IN, BETWEEN, LIKE)

Оператор LIKE

Используется для сравнения строки с шаблоном

|  |  |
| --- | --- |
| Символ | Значение |
| % | Любая последовательность символов |
| - | один символ |
| [ ] | один символ из списка |
| [ ^ ] | один символ не из списка |

Оператор IN

Определяет, совпадает ли указанное значение с одним из значений, содержащихся во вложенном запросе или списке.

Оператор BETWEEN

Определяет диапазон для проверки.

Предикаты используются для ограничения числа возвращаемых записей.  
 Предикаты:  
1) ALL - выбираются все записи.

2) DISTINCT - устраняются дубликаты по выбранным полям.

3) DISTINCTROW - устраняются дубликаты по всем полям.

4) TOP позволяет выбрать первые n отсортированных записей. В инструкции SELECT всегда указывайте ORDER BY вместе с предложением TOP.

Изменение данных в таблице  
UPDATE таблица

SET поле1=новое значение1 [поле2=новое значение2…]

[WHERE условия отбора]

Запрос на создание таблицы

SELECT [предикат] список полей

INTO новая\_таблица  
FROM исходная\_таблица

Удаление записей  
DELETE FROM таблица   
[WHERE условия отбора]

Оглавление

[1. Эволюция баз данных 1](#_Toc106729333)

[2. Архитектура удаленных баз данных 3](#_Toc106729334)

[3. Модели данных 6](#_Toc106729335)

[4. Операции над отношениями 9](#_Toc106729336)

[5. Типы связей в реляционной модели 10](#_Toc106729337)

[6. Целостность данных 10](#_Toc106729338)

[7. Этапы проектирования баз данных 11](#_Toc106729339)

[8. Нормализация отношений 14](#_Toc106729340)

[9. Организация запросов к базе данных на SQL языке 16](#_Toc106729341)