МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КУРС ЛЕКЦИЙ

по дисциплине

«Технология разработки и защиты баз данных»

для студентов направления подготовки бакалавров 01.03.02 – Прикладная математика и информатика

Махачкала 2023

УДК 004.65

Курс лекций «Технология разработки и защиты баз данных» для студентов направления подготовки бакалавров 01.03.02 – Прикладная математика и информатика. Махачкала: ДГТУ, 2023.-90 с.

В рамках дисциплины студенты должны получить навыки использования базовых понятий и определений, основ проектирования баз данных, ориентироваться в тенденциях развития современных средств проектирования баз данных и уровнях представления данных, модели данных и методы обработки моделей представления данных. На этапе проектирования студенты должны уметь выполнять нормализацию схем отношений и строить команды манипуляции данными на языке запросов SQL.

Цель конспекта лекций: изучение теоретических основ проектирования баз данных, компонентов банков данных, характеристик современных СУБД, современных технологий организации БД, приобретение навыков работы в среде конкретных СУБД.

Задачи: в результате изучения лекционного материала у студентов должны быть сформированы системное базовое представление, умения и навыки по основам построения систем управления базами данных; студенты должны иметь представление о роли и месте баз данных в автоматизированных системах, о назначении и основных характеристиках различных систем управления базами данных, их функциональных возможностях.

При изучении лекционного материала дисциплины «Технология разработки и защиты баз данных» у студентов приобретают теоретические знания, формируются новые понятия, выясняются логические связки между ними, развивается интуиция и логика мышления.

Составители: доцент кафедры ПМиИ к.т.н. Мирземагомедова М.М.

Зав. кафедрой ПМиИ, к.ф.-м.н., доцент Исабекова Т.И.

Рецензенты:

Зав кафедрой ИТиПИвЭ ДГТУ, к.э.н., доцент Мурадов М.М.

Директор НИИ РПИ ДГНУХ,
д.э.н. Савзиханова С.Э.

Печатается согласно постановлению Ученого Совета Дагестанского Государственного Технического Университета. от «_____»_____2023г.

ОГЛАВЛЕНИЕ

Лекция 1. Основные положения теории баз данных, хранилищ данных, баз знаний. 6
Основные положения 6
Классификация видов данных 8
Контрольные вопросы 9
Лекция 2. Основные принципы построения концептуальной, логической и физической модели данных 10
Концептуальный уровень моделирования 10
Логический уровень моделирования 11
Физический уровень моделирования 12
Контрольные вопросы 13
Лекция 3. Структуры данных СУБД, общий подход к организации представлений, таблиц, индексов и кластеров. 13
Структура данных СУБД 13
Домены, типы атрибутов 14
Контрольные вопросы 22
Лекция 4. Основные принципы структуризации и нормализации базы данных. 22
Принципы построения баз данных 23
Основные этапы проектирования баз данных 23
Контрольные вопросы 25
Лекция 5. Методы описания схем баз данных в современных СУБД. Структуры данных СУБД. 25
Основная цель СУБД 25
Структура СУБД 28
Контрольные вопросы 29
Лекция 6. Методы организации целостности данных. Модели и структуры информационных систем. 29
Методы обеспечения целостности системы защиты 29
Модели и методы организации данных 31
Организация данных 34
Контрольные вопросы 35
Лекция 7. Современные инструментальные средства проектирования схемы базы данных 35
CASE-средства проектирования баз данных 36
Проектирование баз данных с помощью CASE-средств 37
Классификация CASE-средств 37
Контрольные вопросы 38

Лекция 8. Технологии передачи и обмена данными в компьютерных сетях 38
Основные понятия 38
Современные технологи и методы передачи данных 39
Контрольные вопросы 44
Лекция 9. Введение в SQL и его инструментарий 44
SQL - язык манипулирования данными в реляционной базе данных 44
Описание основных операторов SQL 45
Контрольные вопросы 50
Лекция 10. Установка и настройка SQL-сервера 50
Служба SQL Server Agent: назначение, автоматический запуск от имени доменной учетной записи, роль базы данных MSDB 50
Автоматизация административных операций средствами SQL Server Agent 51
Контрольные вопросы 52
Лекция 11. Импорт и экспорт данных 52
Средства для массового импорта и экспорта данных 52
Повышение производительности передачи данных 54
Минимизация ведение журнала транзакций 55
Отключение и перестроение индексов 55
Отключение и включение ограничений 56
Контрольные вопросы 59
Лекция 12. Автоматизация управления SQL 60
Компоненты агента SQL Server 60
Агент SQL Server как система планирования заданий 60
Контрольные вопросы 64
Лекция 13. Выполнение мониторинга SQL Server с использование оповещений и предупреждений 64
Механизм оповещений 64
Настройка профиля компоненты Database Mail 65
Добавление оператора оповещений 67
Настройка почты агента SQL Server 68
Включение триггеров и задач оповещения 69
Проверка работоспособности оповещений 70
Лекция 14. Настройка текущего обслуживания баз данных 71
Надежное обслуживание баз MS SQL Server 71
Основные возможности QMB 72

Архитектура 72
Контрольные вопросы 78
Лекция 15. Поиск и решение типичных ошибок, связанных с администрированием 78
Ошибки администрирования БД 79
Базовая модель поиска ошибок 79
Стратегии определения ошибок 81
Технологии работы NMS 81
Контрольные вопросы 82
Лекция 16. Способы контроля доступа к данным и управления привилегиями 82
Двухуровневая защита данных 83
Резервное копирование 83
Контрольные вопросы 84
Лекция 17. Алгоритм проведения процедуры резервного копирования 84
Резервное копирование данных 84
Полное резервирование 85
Контрольные вопросы 87
ЛИТЕРАТУРА 88

Лекция 1.
Основные положения теории баз данных, хранилищ данных, баз знаний.

Цель: изучить основные положения теории баз данных, хранилищ данных, баз знаний

План занятия:

1. Рассмотреть основные положения теории БД
2. Рассмотреть классификацию видов данных

Основные положения

Для понимания организации данных в базе данных необходимо знание основных положений теории баз данных. Рассмотрим некоторые положения этой теории.

База данных (Database) - это особым образом организованные и хранимые в электронном виде данные.

Особым образом организованные означает, что данные организованы неким конкретным способом, способным облегчить их поиск и доступ к ним для одного или нескольких приложений. Также такая организация данных предусматривает наличие минимальной избыточности данных.

Базы данных являются одной из разновидностей информационных технологий, а также формой хранения данных.

Целью создания баз данных является построение такой системы данных, которая бы не зависела от программного обеспечения, применяемых технических средств и физического расположения данных в ЭВМ. Построение такой системы данных должно обеспечивать непротиворечивую и целостную информацию. При проектировании базы данных предполагается многоцелевое ее использование.

База данных в простейшем случае представляется в виде системы двумерных таблиц.

Схема данных - описание логической структуры данных, специфицированное на языке описания данных и обрабатываемое СУБД.

Схема пользователя - зафиксированный для конкретного пользователя один вариант порядка полей таблицы.

Системы управления базами данных, СУБД

Система управления базой данных - это программное обеспечение, контролирующее организацию, хранение, целостность, внесение изменений, чтение и безопасность информации в базе данных.

СУБД (Database Management System, DBMS) представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная база данных.

Система управления реляционными базами данных (Relational Database Management System) - это СУБД, основанная на реляционной модели данных.

В реляционной модели данных любое представление данных сводится к совокупности реляционных таблиц (двумерных таблиц особого типа). Системы управления реляционными базами данных используются для построения хранилищ данных.

СУБД имеет программные, технические и организационные составляющие.

Программные средства включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование базы данных. Внутренними языками программирования СУБД являются языки четвертого поколения (C, C++, Pascal, Object Pascal). С помощью языков БД создаются приложения, базы данных и интерфейс пользователя, включающий экранные формы, меню, отчеты.

Аналитику при необходимости работы с конкретной СУБД, в частности, при экспорте данных в среду инструмента Data Mining, следует изучить особенности этой СУБД. Так, например, в базе данных СУБД FoxPro все таблицы и представления базы данных физически хранятся в отдельных файлах, которые объединяются в одном проекте. В СУБД Access все таблицы базы данных хранятся в одном файле.

Для работы с конкретной базой данных, в том числе с целью анализа, аналитику желательно знать описание всех таблиц и их структур (атрибутов, типов данных), количество записей в таблице, а также связи между таблицами. Иногда для этих целей используется словарь данных.

К базам данных, а также к СУБД предъявляются такие требования:

- высокое быстродействие;
- простота обновления данных;
- независимость данных;
- возможность многопользовательского использования данных;
- безопасность данных;
- стандартизация построения и эксплуатации БД (фактически СУБД);
- адекватность отображения данных соответствующей предметной области;
- дружелюбный интерфейс пользователя.

Высокое быстродействие предусматривает малое время отклика, т.е. малый промежуток времени от момента запроса к базе данных до момента реального получения данных.

Независимость данных - это возможность изменения логической и физической структуры базы данных без изменения представлений пользователей.

Независимость данных обеспечивает минимальные изменения структуры базы данных при изменениях стратегии доступа к данным и структуры самих исходных данных. Эти изменения должны быть предусмотрены на этапах концептуального и логического проектирования базы данных с обеспечением минимальных изменений на этапе физического ее проектирования.

Безопасность данных - это защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения. Безопасность включает два компонента: целостность и защиту данных от несанкционированного доступа.

Целостность данных - устойчивость хранимых данных к разрушению и уничтожению, связанным с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей.

Целостность данных - точность и валидность данных. Целостность данных предполагает: отсутствие неточно введенных данных, защиту от ошибок при обновлении баз данных; невозможность удаления (или каскадное удаление) связанных данных разных таблиц; сохранность данных при сбоях техники (возможность восстановления данных) и др.

Защита данных от несанкционированного доступа предполагает ограничение доступа к определенным данным базы и достигается введением мер безопасности: разграничение прав доступа к данным различных пользователей в зависимости от выполняемых ими функций и/или должностных обязанностей; введением защиты в виде паролей; использованием представлений, т.е. таблиц, которые являются производными от исходных и предназначены для работы конкретных пользователей для решения конкретных задач.

Стандартизация обеспечивает преемственность поколений конкретной СУБД, упрощает взаимодействие баз данных одного поколения СУБД с одинаковыми и различными моделями данных.

СУБД отвечает за обработку запросов к базе данных и получение ответа. Способы хранения данных могут быть различными: модель данных может быть, как реляционной, так и многомерной, сетевой или иерархической.

Предметной областью называют определенную часть реального мира, представляющую интерес для конкретного исследования или планируемых действий и соответственно для использования и отображения в информационной системе (в банке данных или знаний).

Банк данных – это автоматизированная система, включающая базу данных, лингвистические, программные, технические, организационно-методические средства, обеспечивающие централизованное накопление и коллективное многоцелевое использование информации в различных областях деятельности пользователей.

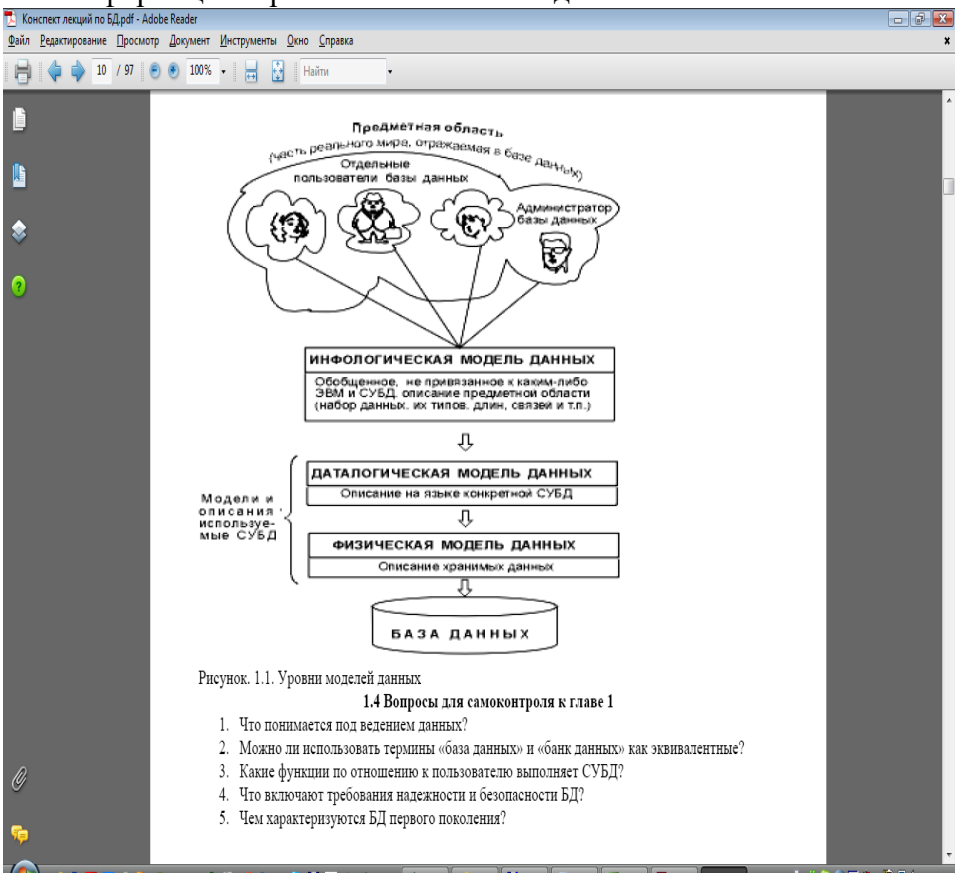


Рисунок. 1.1. Уровни моделей данных

1.4 Вопросы для самоконтроля к главе 1

1. Что понимается под ведением данных?
2. Можно ли использовать термины «база данных» и «банк данных» как эквивалентные?
3. Какие функции по отношению к пользователю выполняет СУБД?
4. Что включают требования надежности и безопасности БД?
5. Чем характеризуются БД первого поколения?

Классификация видов данных

Какими могут быть данные? Ниже приведено несколько классификаций.

Реляционные данные - это данные из реляционных баз (таблиц).

Многомерные данные - это данные, представленные в кубах OLAP.

Измерение (dimension) или ось - в многомерных данных - это собрание данных одного и того же типа, что позволяет структурировать многомерную базу данных.

По критерию постоянства своих значений в ходе решения задачи данные могут быть:

- переменными;
- постоянными;
- условно-постоянными.

Переменные данные - это такие данные, которые изменяют свои значения в процессе решения задачи.

Постоянные данные - это такие данные, которые сохраняют свои значения в процессе решения задачи (математические константы, координаты неподвижных объектов) и не зависят от внешних факторов.

Условно-постоянные данные - это такие данные, которые могут иногда изменять свои значения, но эти изменения не зависят от процесса решения задачи, а определяются внешними факторами.

Данные, в зависимости от тех функций, которые они выполняют, могут быть справочными, оперативными, архивными.

Следует различать данные за период и точечные данные. Эти различия важны при проектировании системы сбора информации, а также в процессе измерений.

- данные за период;
- точечные данные.

Данные за период характеризуют некоторый период времени. Примером данных за период могут быть: прибыль предприятия за месяц, средняя температура за месяц.

Точечные данные представляют значение некоторой переменной в конкретный момент времени. Пример точечных данных: остаток на счете на первое число месяца, температура в восемь часов утра.

Данные бывают первичными и вторичными. Вторичные данные - это данные, которые являются результатом определенных вычислений, примененных к первичным данным. Вторичные данные, как правило, приводят к ускоренному получению ответа на запрос пользователя за счет увеличения объема хранимой информации.

Метаданные

В завершение лекции о данных рассмотрим понятие метаданных.

Метаданные (Metadata) - это данные о данных.

В состав метаданных могут входить: каталоги, справочники, реестры.

Метаданные содержат сведения о составе данных, содержании, статусе, происхождении, местонахождении, качестве, форматах и формах представления, условиях доступа, приобретения и использования, авторских, имущественных и смежных с ними правах на данные и др.

Метаданные - важное понятие в управлении хранилищем данных.

Метаданные, применяемые при управлении хранилищем, содержат информацию, необходимую для его настройки и использования. Различают бизнес-метаданные и оперативные метаданные.

Бизнес-метаданные содержат бизнес-термины и определения, принадлежность данных и правила оплаты услуг хранилища.

Оперативные метаданные - это информация, собранная во время работы хранилища данных:

- происхождение перенесенных и преобразованных данных;
- статус использования данных (активные, архивированные или удаленные);
- данные мониторинга, такие как статистика использования, сообщения об ошибках и т.д.

Метаданные хранилища обычно размещаются в репозитории. Это позволяет использовать метаданные совместно различным инструментам, а также процессам при проектировании, установке, эксплуатации и администрировании хранилища.

Контрольные вопросы:

1. Дайте определения понятиям: база данных, схема данных, СУБД, целостность данных
2. Перечислите виды данных с описанием

Лекция 2.
Основные принципы построения концептуальной, логической и физической модели данных

Цель: рассмотреть основные принципы построения концептуальной, логической и физической модели данных

План занятия:

1. Рассмотреть уровни моделирования данных
2. Изучить концептуальный уровень: базовые элементы, пример, фокус моделирования
3. Изучить логический уровень: базовые элементы, пример, фокус моделирования
4. Изучить физический уровень: базовые элементы, пример, фокус моделирования

Существует три уровня моделирования данных:

- Концептуальный -► концептуальная модель
- Логический -► логическая модель
- Физический -► физическая модель

Концептуальный уровень моделирования

На концептуальном уровне моделирования мы определяем основные понятия предметной области и их взаимосвязь. Иногда также используют термин "построение онтологиии предметной области".

Базовый элемент концептуальной модели: бизнес-сущность или бизнес-объект.

Примеры бизнес-сущностей:

- клиент (как вариант, клиент-ФЛ, клиент-ЮЛ),
- продукт (как вариант, товар, услуга),
- сделка (как вариант, заказ),
- контракт.

Концептуальная модель, как правило, представлена в документах класса ГЛОССАРИЙ (примечание: это может быть как местная wiki, так и разрозненные главы типа "Глоссарий" в различных документах компании). Схематически (графически) наиболее удобным способом отображение концептуальной модели является диаграмма типа "диаграмма классов". Связи между бизнес-сущностями полезно «окрашивать» действиями из предметной области, например, клиент заключает сделку, заказ состоит из товаров. Наилучшая нотация для концептуального уровня – подборка элементов из слоя Business и Motivation методологии Archimate. В данной нотации рекомендуется использоваться объект Meaning, а также бизнес-сущности могут быть привязаны к ряду других элементов (например, value или capability), что улучшает их окрашивание или лучше структурирует (делает более наглядной) моделируемую предметную область.

Фокус моделирования:

- понятийная/смысловая модель, выработка глоссария
- разработка онтологии домена (онтики)
- создание/проработка представления о понятии/явлении, как об информационном объекте
- выделение ключевых атрибутов, характеризующих ту или иную бизнес-сущность

Сложные концептуальные модели, содержащие десятки бизнес-сущностей, разбиваются на домены. Домен – группировка «родственных» сущностей, образующих модель отдельного фрагмента моделируемой предметной области. Иногда концептуальная

модель становится существенной частью логической модели. Это бывает в следующих случаях:

- при создании BI-систем
- при создании модели данных интеграционной платформы.

В отдельных случаях приходится поддерживать концептуальную модель двух видов:

- текущую концепт.модель, являющуюся мостиком между логическими моделями данных конкретных приложений.
- текущую концепт.модель, отражающую текущее понимание нами онтологии предметной области. Эта модель может меняться почти каждый день.

Логический уровень моделирования

Логическая модель является уточнением и детализацией концептуальной модели. Но это лишь, с одной стороны. На построение логической модели также влияет:

- тип планируемой СУБД, которая будет воплощать модель
- класс проектируемой системы: операционная (транзакционная) или аналитическая (BI)
- исторически сложившияся трактовка предметной области вендором системы

Логический уровень моделирования – это уровень логики организации данных, то есть какие данные и как сгруппированы и связаны друг с другом. Концептуальный уровень больше заботится о смысловых связях, логический – о реальных связях между объектами системы (ссылки объектов друг на друга, отношения объектов). Концептуальный уровень оперирует бизнес-сущностями, логический – сущностями будущей или фактически имеющейся информационной системы (например, базы данных). В компаниях с большой историей логический уровень задан фактически развернутыми системами конкретных вендоров.

Важное замечание. В простейших случаях концептуальные объекты (бизнес-сущности) совпадают с объектами логического уровня. В таких случаях фаза концептуального моделирования может совсем не требоваться или совпадать с фазой построения логической модели. Очень часто молодые системные аналитики пытаются построить сложную логическую модель и проваливают работу, не подозревая о необходимости фазы концептуального моделирования, которая должна выполняться с участием бизнес-аналитиков и самого бизнеса. В настоящее время - время диджитал - концептуализация и цифровизация должны рассматриваться как синонимы.

Тип объектов логического уровня соответствует типам объектов избранной СУБД. Для реляционных и объектно-ориентированных баз данных – это ENTITY (сущность). Сущности логического уровня – это сущности, которыми оперирует информационная система (база данных или сервер приложений). К сущностям логического уровня подвязываются методы работы с этими сущностями.

Базовый элемент логической модели: сущность (в ООП - класс).

Для проектирования реляционных баз данных используется нотация ERD. Рамками этой нотации фактически и задаётся логический уровень моделирования. Однако для систем, имеющих на верхнем уровне объектную модель, логический уровень описывается диаграммной классов (из нотации UML). Детализирующие (вспомогательные) и часто используемые диаграммы логического уровня моделирования – это диаграммы состояний.

Примеры сущностей (классов):

- клиент -► party + customer + customer_profile + person
- продукт -► продукт + предложение продукта + price plan
- заказ -► order, order_item

Фокус моделирования:

- выделение элементарных сущностей, элементарных логических единиц (классов), имеющих самостоятельный смысл в моделируемой предметной области.
- детальное (точное) уточнение (установка) взаимосвязей между сущностями

- перечисление всех (!) значимых для бизнеса атрибутов сущности
- Разделение атрибутов на простые атрибуты и перечисления (будущие справочники)
- выделение сущностей не столько как контейнеров для атрибутов, сколько как объектов поведения

Немаловажное влияние на сущности логического уровня и их взаимосвязи оказывает тип проектируемой системы. Если проектируемая система относится к BI-классу, то следует понимать назначение BI-системы, ожидаемые от нее витрины, срезы, аналитики, метод моделирования времени (динамики изменения данных) и т.п.

Если в ходе проектирования разрабатывается логическая модель не одной, а нескольких систем, что часто имеет место быть в крупных компаниях, внедряющих пятую, десятую или сто двадцатую систему, то при разработке логической модели указывают к какой системе принадлежит (или будет принадлежать) та или иная сущность. Распределение сущностей по различным информационным системам даёт возможность грубо наметить (спрогнозировать) будущие информационные потоки (и точки интеграции) между системами.

Важное замечание. При разработке новой информационной системы, являющейся частью комплекса унаследованных систем, часто приходится использовать как проектирование сверху вниз (от концептуального уровня к физическому), так и обратное – снизу-вверх: от уже существующих физических моделей к логической и далее мапирование в концептуальную. Это на порядок или даже на 2 порядка усложняет проектирование даже если нужно создать/автоматизировать один сквозной процесс, протекающий через ряд информационных систем.

Иногда при проработке логического уровня возникают сущности, которые трудно подвязать к бизнес-сущностям концептуального уровня и тогда возникает вопрос: нужно ли на концептуальном уровне завести новую бизнес-сущность? Ответ таков:

- с одной стороны, не стоит перегружать концептуальный уровень.
- с другой стороны, если на концептуальном уровне появляется новая сущность, она должна быть отражена в глоссарии, как принципиально новое понятие, существенно отличное от других понятий данной предметной области.
- не ленитесь использоваться для организации сущностей отношения наследования-специализации.

Каждый вендор информационной системы имеет логическую модель своей системы даже если он и не раскрывает ее своим клиентам. При интеграции нескольких систем приходится сначала увязывать их логические модели на концептуальном уровне моделирования, а лишь потом строить связи между логическими моделями разных систем. Например, в системе автоматизации продаж клиент может быть представлен как PARTY, в ERP-системе той же компании как КОНТРАГЕНТ, а системе биллинга той же компании, как АБОНЕНТ.

Физический уровень моделирования

Физический уровень – это уровень таблиц для реляционных моделей данных.

Базовый элемент физической модели: таблица.

Примеры таблиц:

- клиент -► table_1
- продукт -► table_2
- сделка -► table_3

Не исключается, что одна таблица физического уровня может участвовать в моделировании сразу нескольких логических сущностей.

Фокус моделирования:

- Выделение отдельных таблиц, в том числе как результат нормализации данных.
- Выделение таблиц-справочников.

- Определение ключей.
- Разделение атрибутов на простые атрибуты и перечисления (будущие справочники)

В простейших случаях логические объекты (сущности) совпадают с объектами физического уровня. Это характерно для самых простых баз данных реляционного типа.

Контрольные вопросы:

1. Что относится к базовым элементам концептуального, логического и физического уровней
2. Приведите примеры концептуального, логического и физического уровней
3. В чем заключается фокус моделирования концептуального, логического и физического уровней

Лекция 3.
Структуры данных СУБД, общий подход к организации представлений, таблиц, индексов и кластеров.

Цель: рассмотреть структуры данных СУБД, изучить общий подход к организации представлений, таблиц, индексов и кластеров

План занятия:

1. Структура данных СУБД
2. Общий подход к организации представлений, таблиц, индексов и кластеров

Структура данных СУБД

Пользователи нуждаются в описании схемы на некотором понятийном для всех уровне, которое называется моделью данных.

Модель данных – интегрированный набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные в некоторой организации.

Модель данных является абстрактным представлением объектов и событий «реального мира», а также существующих между ними связей. В этой абстракции акцент делается на самых важных и неотъемлемых аспектах деятельности организации, а все второстепенные свойства игнорируются. Таким образом, можно сказать, что модель данных представляет саму организацию.

Модель данных можно рассматривать как сочетание трех указанных ниже компонентов:

- структурной части – набора правил, по которым может быть построена база данных;
- управляющей части, определяющей типы допустимых операций с данными (операции обновления и извлечения данных, а также операции изменения структуры базы данных);
- набора ограничений поддержки целостности данных, гарантирующих корректность используемых данных.

Цель построения модели данных заключается в понятном представлении данных, которое можно будет легко применить при проектировании базы данных.

Для трехуровневой архитектуры существуют следующие три связанные модели данных (объектные модели):

- **Внешняя модель данных**, которую иногда называют моделью предметной области (она отображает представления каждого типа пользователей организации).

Silverrun BPM (функциональная модель, модель потоков данных).
Внешние объекты, процессы, потоки, накопители.
Описание структур данных.

- **Концептуальная модель данных**, которая отображает логическое (или обобщенное) представление о данных, не зависимое от типа выбранной СУБД.

Silverrun ERX (ER – модель).
Сущности, связи, атрибуты.

- **Внутренняя модель данных**, которая отображает концептуальную схему для конкретной СУБД.

Silverrun RDM (логическая модель, для конкретной СУБД – реляционная модель для реляционной СУБД).

Домены, типы атрибутов.

Кроме объектных моделей существуют модели данных на основе записей. В зависимости от способа записи данных база данных состоит из нескольких (множества) записей фиксированного формата, которые могут иметь разные типы.
Каждый тип записи определяет фиксированное количество полей, каждое из которых имеет фиксированную длину.
Существует три основных типа записей, определяющие три типа моделей:

- реляционная модель данных (relational data model);
- сетевая модель данных (network data model);
- иерархическая модель данных (hierarchical data model).

Реляционная модель данных основана на понятии математических отношений. В реляционной модели данные и связи представлены в виде таблиц (см. рис.7), каждая из которых имеет несколько столбцов с уникальными именами.
Сущность «Работник»

StaffNo	FName	LName	DOB	Salary	BranchNo
2152	Иван	Егоров	20.02.1935	3500	02
2876	Петр	Сивохин	2.12.1981	2500	11

Сущность «Отдел»

BranchNo	Name	№ комнаты
02	Отдел сбыта	21
11	Плановый отдел	18

Рис.7 Описание данных в реляционной модели

В реляционной модели данных единственное требование состоит в том, чтобы база данных с точки зрения пользователя выглядела как набор таблиц. Это требование относится только к внешнему и концептуальному уровням архитектуры ANSI/SPARC.

В сетевой модели данные представлены в виде коллекций записей, а связи – в виде наборов (см. рис.8). Сетевую модель можно представить как граф с записями в виде узлов (вершин) графа, и наборов данных в виде его ребер.

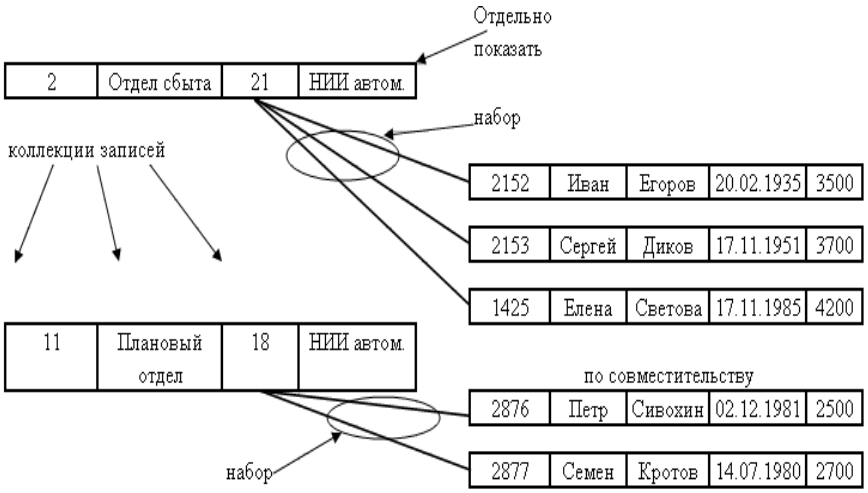


Рис.8. Сетевая модель данных

Иерархическая модель данных является подтипом сетевой модели. В ней данные представлены как коллекции записей, а связи – как наборы (см. рис.9). Однако узел может иметь только одного родителя

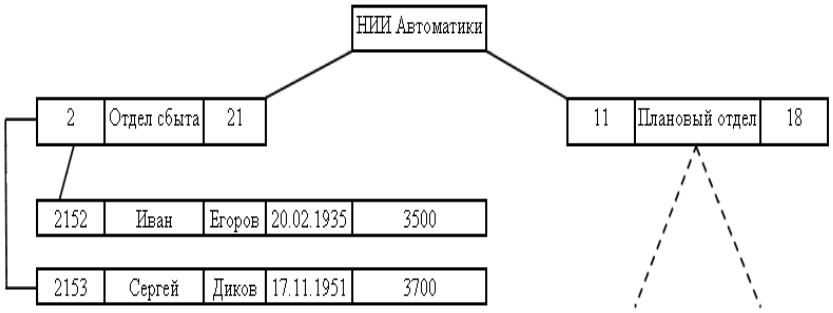


Рис.9. Иерархическая модель данных

Наиболее важные термины описания структуры данных представлены на рисунке 10.

Отношение – плоская таблица, состоящая из столбцов и строк.

Атрибут – поименованный столбец отношения.

Домен – набор допустимых значений для одного или нескольких атрибутов.

Кортеж – строка отношения.

Степень – количество атрибутов, содержащихся в отношении.

Кардинальность – количество кортежей, которое содержит отношение.

Первичный ключ – атрибут или множество атрибутов, которые выбраны для уникальной идентификации кортежей отношения.

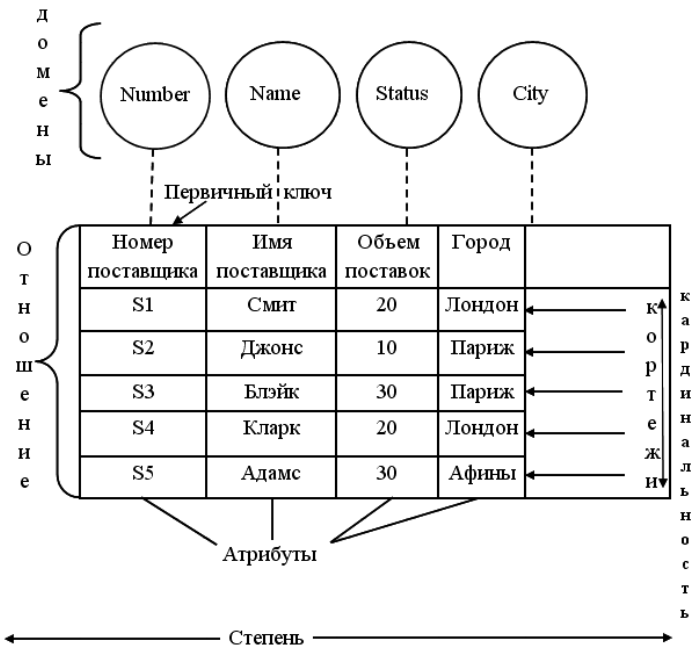


Рис.10. Структура данных

Реляционная база данных – набор отношений (нормализованных).

Каждый кортеж отношения представляет собой определенное высказывание об объекте.

Точное определение термина отношение:

Пусть задано множество доменов Ti (I = 1, 2,..n), все из которых необязательно должны быть различными. Тогда r будет отношением, определенным на этих доменах, если оно состоит из двух частей – заголовка и тела, где

а) заголовок – это множество из n атрибутов вида Ai : Ti; здесь Ai – имена атрибутов, а Ti – соответствующие им имена типов.

б) тело – это множество из m кортежей t; здесь t, в свою очередь, является множеством компонентов вида Ai : Vi, в которых Vi – значение типа Ti, то есть значение атрибута Ai в кортеже t (I = 1, 2,..n).

Подходы к определению понятия отношения могут быть различными. Математически отношение может быть определено как множество кортежей, являющееся подмножеством декартова произведения фиксированного числа областей (доменов). В результате получаем, что в каждом кортеже должно быть одинаковое число компонентов (атрибутов) и значение каждого из них выбирается из некоторого определенного домена.

В терминах таблицы, заголовок – это строка, состоящая из названий столбцов и соответствующих имен типов, а тело – это множество строк данных.

Отношения обладают следующими свойствами:

1. в них нет одинаковых кортежей;
2. кортежи отношения не имеют упорядоченности в направлении сверху вниз;
3. атрибуты в кортежах не упорядочены слева направо;
4. каждый кортеж содержит ровно одно значение для каждого атрибута.

Свойство 1: Свойство следует из того факта, что тело отношения – это математическое множество (кортежей), а в математике множества по определению не содержат одинаковых элементов.

Таблица, в общем случае, может содержать одинаковые строки (при отсутствии правил, запрещающих это).

Свойство 2: Свойство следует из того, что тело отношения – это математическое множество, а простые множества в математике не упорядочены.

В таблице строки упорядочены сверху вниз.

Свойство 3: Свойство следует из того факта, что заголовок отношения также определен, как множество (атрибутов). Атрибут всегда определяется по имени, а не по расположению.

В таблице столбцы могут быть упорядочены.

Свойство 4: Свойство следует из определения кортежа: кортеж является множеством из n компонентов. Отношение, удовлетворяющее этому свойству, называется нормализованным или представленным в первой нормальной форме (1НФ).

В частности, возможно существование типа, значениями которого будут отношения и, следовательно, существование отношения с атрибутами, значениями которых также являются отношения.

Дополнительные свойства:

- Отношение имеет имя, которое отличается от имен всех других отношений.
- Каждый атрибут имеет уникальное имя.
- Значения атрибута берутся из одного и того же домена.
- Все кортежи одного отношения должны иметь одно и то же количество атрибутов.

Необходимо иметь возможность уникальной идентификации каждого кортежа отношения с помощью атрибутов.

Атрибут или множество атрибутов, которое единственным образом идентифицирует кортеж данного отношения, называется суперключом.

Потенциальный ключ – это суперключ, который не содержит подмножества, также являющегося суперключом данного отношения.

Такой ключ обладает двумя свойствами:

- уникальность – в каждом кортеже отношения значение ключа единственным образом идентифицирует этот кортеж;
- неприводимость – никакое допустимое подмножество ключа не обладает свойством уникальности.

Отношение может иметь несколько потенциальных ключей. Если ключ состоит из нескольких атрибутов, то он называется составным ключом.

Первичный ключ – это потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения. Потенциальные ключи, которые не выбраны в качестве первичного ключа, называются альтернативными ключами.

Отделение

Первичный ключ

Номер отделен.	Адрес	Район	Город	Почт. код	Номер телефона	Номер факса
B5	Красная 40	Первом.	Пенза	440017	36-82-38	33-45-61
B3	Московск. 2	Ленинск.	Пенза	———	48-54-11	45-06-15
B2	Clover Street	———	London	NW106EU	181-963-103	181-453-799

Штат

Внешний ключ

№ работн.	Фамилия	Адрес	Телефон	Должн.	Пол	Дата рождения	№ отделен
0021	Джонсон	Попова 10	34-34-10	менеджер	М	01.10.1945	B5

Рис.11 Пример отношений с выбранными ключами

Атрибут «Город» не может быть выбран в качестве потенциального ключа. Каждое значение атрибута «Номер отделения» имеет уникальное значение, поэтому этот атрибут является потенциальным ключом. Атрибуты «Номер телефона» и «Номер факса» также являются потенциальными ключами (рис.11).

«Номер отдела» - первичный ключ.

Внешний ключ – это атрибут или множество атрибутов, которое соответствует какому-либо потенциальному ключу некоторого (может быть, того же самого) отношения.

Внешние ключи отражают определенную связь между кортежами этих отношений.

Схема отношения обозначается с помощью имени отношения, за которым (в скобках) перечисляются имена атрибутов. При этом первичный ключ подчеркивается.

Отделение (Номер отдела, Адрес, Район, Город, Почт. код, Номер телефона, Номер факса).

Как уже говорилось ранее для управления отношениями в реляционных СУБД можно использовать процедурные и непроцедурные языки. Коддом были предложены формальные (а не дружественные пользователю) языки – реляционная алгебра и реляционное исчисление.

Реляционная алгебра представляет процедурный язык, который может быть использован, чтобы сообщить СУБД как следует построить требуемое отношение на базе одного или нескольких существующих в базе данных отношений.

Реляционное исчисление представляет собой непроцедурный язык, который можно использовать для определения того, каким будет некоторое отношение, созданное на основе одного или нескольких других отношений базы данных.

Реляционная алгебра и реляционное исчисление эквивалентны друг другу. Для каждого выражения алгебры существует эквивалентное выражение в реляционном исчислении (и наоборот).

Следует различать сами отношения (то есть значения отношений) и переменные-отношения, так как отношения являются переменными (в математическом смысле), над которыми можно осуществлять определенные действия – конструировать новые отношения, изменять их значения и т.д.

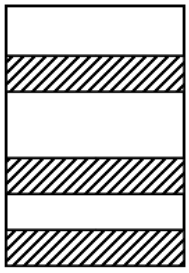
Кодд определил восемь операторов реляционной алгебры, использующих отношения в качестве операндов и возвращающих отношения в качестве результата.

Восемь операторов составляют две группы по четыре оператора:

1. Традиционные операции над множествами – объединение, пересечение, разность и декартово произведение (с учетом того, что их операндами являются отношения, а не произвольные множества).

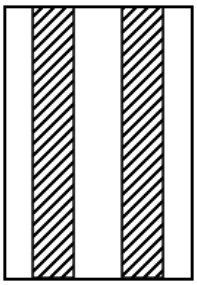
2. Специальные реляционные операции – выборка, проекция, соединение и деление.

Выборка



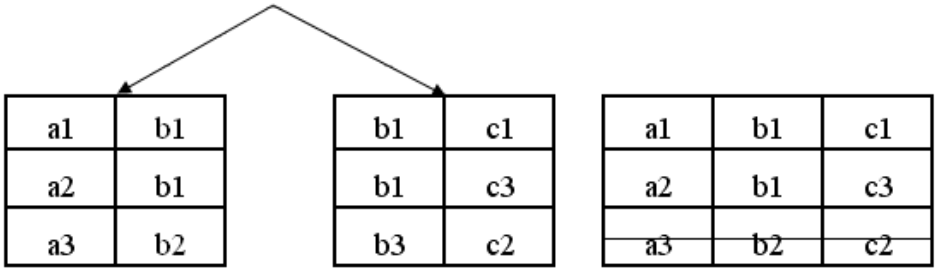
Возвращает отношение, содержащее все кортежи из заданного отношения, которые удовлетворяют определенным условиям.

Проекция



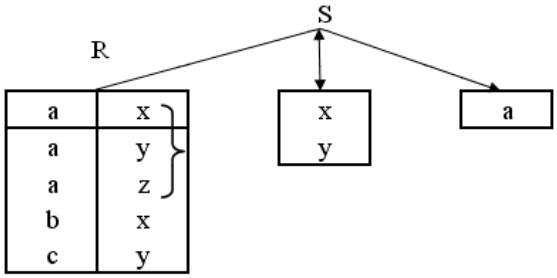
Возвращает отношение, содержащее кортежи заданного отношения, состоящие из определенных атрибутов (остальные атрибуты исключены).

Соединение



Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей, принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах (причем эти общие значения в результирующем кортеже появляются один раз, а не дважды).

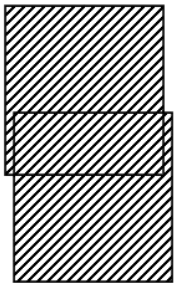
Деление



Для заданных двух отношений (унарных) и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

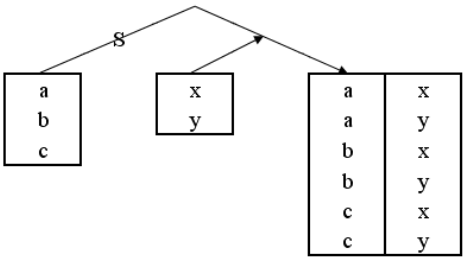
Результатом оператора деления R на S является набор кортежей отношения R, определенных на множестве атрибутов не входящих в S, которые соответствуют комбинации всех кортежей отношения S.

Объединение



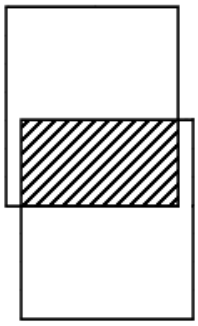
Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим.

Произведение



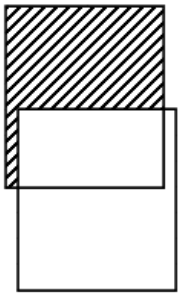
Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям.

Пересечение



Возвращает отношение, содержащее все кортежи, которые принадлежат одновременно двум заданным отношениям.

Разность



Возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух заданных отношений и не принадлежат второму.

Так как результатом выполнения любой операции является отношение, то можно записывать вложенные реляционные выражения, то есть выражения, в которых операнды сами представлены реляционными выражениями, причем произвольной сложности.

Основная цель алгебры – обеспечить запись реляционных выражений. Выражения можно преобразовывать в соответствии с правилами преобразования. Таким образом, реляционная алгебра может служить хорошим обоснованием для построения оптимизатора запросов СУБД.

Реляционная алгебра в явном виде представляет набор операций, которые можно использовать, чтобы сообщить системе, как в базе данных из определенных отношений построить некоторое требуемое отношение. Реляционное исчисление просто представляет систему обозначений для определения требуемого отношения в терминах данных отношений.

Рассмотрим запрос: Выбрать номера поставщиков и названия городов, в которых находятся поставщики детали с номером P2.

Поставщики (S)

Номер поставщика	Имя	Рейтинг	Город
S1	Смит	20	Лондон
S2	Кларк	20	Лондон
S3	Джоунс	10	Париж
S4	Блэик	30	Рим

Поставки (P)

Номер поставки	Название товара	Цвет	Вес	Номер поставщика
P1	Болты	Красный	12.0	S1
P2	Гайки	Белый	20.0	S3
P3	Диски	Голубой	10.5	S2
P2	Гайки	Белый	11.0	S1

- сначала выполнить соединение отношения поставщиков и отношения поставок по атрибуту «Номер поставщика»;

№ поставщика	Имя	Рейтинг	Город	№ поставки	Товар	Цвет	Вес
S1	Смит	20	Лондон	P1	Болты	Красный	12.0
S1	Смит	20	Лондон	P2	Гайки	Белый	11.0
S2	Кларк	20	Лондон	P3	Диски	Голубой	10.5
S3	Джоунс	10	Париж	P2	Гайки	Белый	20.0

- Выбрать из результата кортежи с номером детали P2.

№ поставщика	Имя	Рейтинг	Город	№ поставки	Товар	Цвет	ВВес
S1	Смит	20	Лондон	P2	Гайки	Белый	11.0
S3	Джоунс	10	Париж	P2	Гайки	Белый	20.0

Выполнить для результата выборки операцию проекции по атрибутам “Номер поставщика” и “Город”.

№ поставщика	Город
S1	Лондон
S3	Париж

Этот же запрос в терминах реляционного исчисления формулируется приблизительно так:

получить атрибуты “№ поставщика” и “Город” для таких поставщиков, у которых в отношении “Поставки” существует запрос о поставке с тем же значением атрибута “№ поставщика” и со значением атрибута “Номер поставки”, равным P2.

Select P.”Номер поставки”, S.”Город”

From “Поставщики” S, “Поставки” P

Where P.“Номер поставщика” = S.” Номер поставщика ”

And P.” Номер поставки ”=P2

Индекс (Index) — это объект базы данных, создаваемый для повышения производительности выборки данных и контроля уникальности первичного ключа.

Функция (Function) - это объект базы данных, представляющий поименованный набор команд SQL и/или операторов специализированных языков обработки

программирования базы данных, который при выполнении возвращает значение — результат вычислений.

Для обработки данных специальным образом или для реализации поддержки ссылочной целостности базы данных используются объекты: хранимая процедура, триггер.

Хранимая процедура (Stored procedure) — это объект базы данных, представляющий поименованный набор команд SQL.

Триггер (Trigger) - это объект базы данных, который представляет собой специальную хранимую процедуру. Эта процедура запускается автоматически, когда происходит связанное с триггером событие (например, до вставки строки в таблицу).

Данные объекты реляционной базы данных представляют собой программы, т.е. исполняемый код. Этого код обычно называют серверным кодом (server-side code), поскольку он выполняется компьютером, на котором установлена СУБД. Планирование и разработка такого кода является одной из задач проектировщика реляционной базы данных.

Для эффективного управления разграничением доступа к данным в Interbase/Firebird поддерживается объект роль.

Роль (Role) — это объект базы данных, представляющий собой поименованную совокупность привилегий, которые могут назначаться пользователям, категориям пользователей или другим ролям.

Контрольные вопросы:

1. Из чего состоит структура данных СУБД
2. Опишите общий подход к организации представлений, таблиц, индексов и кластеров

Лекция 4.
Основные принципы структуризации и нормализации базы данных.

Цель: изучить основные принципы структуризации и нормализации баз данных

План занятия:

1. Изучить основные принципы структурирования БД
2. Рассмотреть этапы нормализации БД

1) В настоящее время успешное функционирование различных фирм, организаций и предприятий просто невозможно без развитой информационной системы, которая позволяет автоматизировать сбор и обработку данных. Обычно для хранения и доступа к данным, содержащим сведения о некоторой предметной области, создается база данных.

База данных (БД) - именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области. Под предметной областью понимается некоторая область человеческой деятельности или область реального мира, на основе которой создается БД и её структура.

Система управления базами данных (СУБД) - совокупность языковых и программных средств, предназначенных для создания, наполнения, обновления и удаления баз данных.

Принципы построения баз данных

К современным базам данных, а, следовательно, и к СУБД, на которых они строятся, предъявляются следующие основные требования:

- ·Высокое быстродействие (малое время отклика на запрос). Время отклика - промежуток времени от момента запроса к БД до фактического получения данных.
- ·Простота обновления данных.
- ·Независимость данных - возможность изменения логической и физической структуры БД без изменения представлений пользователей.
- ·Совместное использование данных многими пользователями.
- ·Безопасность данных - защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения.
- ·Стандартизация построения и эксплуатации БД (фактически СУБД).
- ·Адекватность отображения данных соответствующей предметной области.
- ·Простой интерфейс пользователя.

Важнейшими являются первые два противоречивых требования: повышение быстродействия требует упрощения структуры БД, что, в свою очередь, затрудняет процедуру обновления данных, увеличивает их избыточность.

Безопасность данных включает их целостность и защиту. **Целостность данных** - устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей. Она предполагает:

- отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте;
- защиту от ошибок при обновлении БД;
- невозможность удаления (или каскадное удаление) связанных данных разных таблиц;
- неискажение данных при работе в многопользовательском режиме и в распределенных базах данных;
- сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается триггерами целостности - специальными приложениями-программами, работающими при определенных условиях. **Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:**

- · введением системы паролей;
- · получением разрешений от администратора базы данных (АБД);
- · запретом от АБД на доступ к данным;
- · формирование видов - таблиц, производных от исходных и предназначенных конкретным пользователям.

Стандартизация обеспечивает преемственность поколений СУБД, упрощает взаимодействие БД одного поколения СУБД с одинаковыми и различными моделями данных. При этом может быть осуществлен как локальный, так и удаленный доступ к данным (технология клиент/сервер или сетевой вариант).

Проектирование баз данных - процесс решения класса задач, связанных с созданием баз данных.

Основные задачи проектирования баз данных:

- · Обеспечение хранения в БД всей необходимой информации.
- · Обеспечение возможности получения данных по всем необходимым запросам.
- · Сокращение избыточности и дублирования данных.
- ·Обеспечение целостности данных (правильности их содержания): исключение противоречий в содержании данных, исключение их потери и т.д.

Основные этапы проектирования баз данных:

Концептуальное (инфологическое) проектирование – построение формализованной модели предметной области. Такая модель строится с использованием стандартных языковых средств, обычно графических, например ER-диаграмм (диаграмм «Сущность-связь»). Такая модель строится без ориентации на какую-либо конкретную СУБД.

Основные элементы данной модели:

- · Описание объектов предметной области и связей между ними.
- Описание информационных потребностей пользователей (описание основных запросов к БД).
- · Описание алгоритмических зависимостей между данными.
- · Описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

Логическое (даталогическое) проектирование – отображение инфологической модели на модель данных, используемую в конкретной СУБД, например на реляционную модель данных. Для реляционных СУБД даталогическая модель – набор таблиц, обычно с указанием ключевых полей, связей между таблицами. Если инфологическая модель построена в виде ER-диаграмм (или других формализованных средств), то даталогическое проектирование представляет собой построение таблиц по определённым формализованным правилам, а также нормализацию этих таблиц. Этот этап может быть в значительной степени автоматизирован.

Физическое проектирование – реализация даталогической модели средствами конкретной СУБД, а также выбор решений, связанных с физической средой хранения данных: выбор методов управления дисковой памятью, методов доступа к данным, методов сжатия данных и т.д. – эти задачи решаются в основном средствами СУБД и скрыты от разработчика БД.

На этапе инфологического проектирования в ходе сбора информации о предметной области требуется выяснить:

- · основные объекты предметной области (объекты, о которых должна храниться информация в БД);
- · атрибуты объектов;
- · связи между объектами;
- основные запросы к БД.

2) Проектирование реляционной БД заключается в разработке структуры данных, т.е. в определении состава таблиц и связей между ними. При этом структура должна быть эффективной и обеспечивать: быстрый доступ к данным; отсутствие дублирования (повторения) данных; целостность данных.

Проектирование БД можно представить следующим образом:

- Сбор всей информации об объектах решаемой задачи в рамках одной таблицы (одного отношения)
- Разбиение полученной таблицы на несколько взаимосвязанных таблиц на основе принципа нормализации отношений.

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. **Цель нормализации** сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

первая нормальная форма (1NF); вторая нормальная форма (2NF); третья нормальная форма (3NF); нормальная форма Бойса-Кодда (BCNF); четвертая

нормальная форма (4NF); пятая нормальная форма, или нормальная форма проекции-соединения (5NF или PJ/NF).

Основные свойства нормальных форм: каждая следующая нормальная форма в некотором смысле лучше предыдущей; при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы.

Основными считаются первые 3 нормальные формы:

1НФ - таблица находится в первой нормальной форме (1НФ) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто. (Любое поле таблицы содержит неделимую информацию и в таблице определен первичный ключ)

2НФ - Таблица находится во второй нормальной форме (2NF) в том и только в том случае, когда находится в 1NF, и каждый ее неключевой атрибут полностью зависит от первичного ключа. (Таблица должна удовлетворять 1NF и любое неключевое поле должно однозначно идентифицироваться ключевыми полями.)

3НФ – Таблица находится в третьей нормальной форме (3NF) в том и только в том случае, если находится в 2NF и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. (Таблица должна удовлетворять 2NF и ни одно из неключевых полей не должно однозначно идентифицироваться значением другого неключевого поля (полей))

Контрольные вопросы:

1. Перечислите основные принципы структурирования БД
2. Перечислите этапы нормализации БД

Лекция 5.
Методы описания схем баз данных в современных СУБД. Структуры данных СУБД.

Цель: рассмотреть методы описания схем баз данных в современных СУБД, изучить структуру данных СУБД

План занятия:

1. Основная цель СУБД
2. Методы описания схем БД в современных СУБД
3. Структура данных СУБД

Основная цель СУБД

Основная цель СУБД заключается в том, чтобы предложить пользователю абстрактное представление данных, скрыв от него конкретные особенности хранения и управления ими.

При этом, поскольку БД является общим ресурсом, то каждому пользователю может потребоваться свое, отличное от других, представление об информации, хранимой в БД.

Архитектура большинства современных СУБД строится на базе так называемой архитектуры ANSI – SPARC (American National Standard Institute Standards Planning and Requirements Committee). Хотя модель ANSI/SPARC не стала стандартом, тем не менее,

она представляет собой основу для понимания некоторых функциональных особенностей СУБД.

Наиболее важным моментом этой модели является определение трех уровней абстракции, то есть трех различных уровней описания элементов данных.

В модели определены три уровня – внешний, концептуальный и внутренний (см. рис.5).

Причины, по которым желательно выполнять такое разделение:

- Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое собственное представление о них.
- Пользователи не должны иметь дело с подробностями физического хранения данных в базе.
- АБД должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательское представление.

Внешний уровень – представление базы данных с точки зрения пользователей. Этот уровень описывает ту часть базы данных, которая относится к каждому пользователю. С точки зрения пользователя определение данных представляется в контексте предметной области.

Рис. 5. Архитектура современных СУБД

Каждый пользователь имеет дело с представлением «реального мира», выраженным в наиболее удобной для него форме.

Интерес представляют следующие понятия:

1. Сущность – объект «реального мира», такой как Работник, Отдел, Договор.
2. Атрибуты – свойства или качества каждой сущности (например, Имя, Адрес, Зарплата для сущности Работник).
3. Связи – взаимоотношения между сущностями (например, Работник работает в Отделе).

Внешнее представление пользователя содержит только те сущности, атрибуты и связи «реального мира», которые интересны этому пользователю. Другие сущности, атрибуты и связи, которые ему не интересны, также могут быть представлены в базе данных, но они важны для другого пользователя.

(Адрес – для отдела кадров, а бухгалтерия может им не пользоваться).

Концептуальный уровень – обобщающее представление базы данных. Этот уровень описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними. Этот уровень обобщает представления всех пользователей – фактически, это полное представление требований к данным со стороны организации, в которой работают пользователи. Это представление не зависит от способа хранения этих данных.

На концептуальном уровне представлены следующие компоненты:

- все сущности, их атрибуты и связи;
- накладываемые на данные ограничения;
- семантическая информация о данных;
- информация о безопасности и целостности данных.

Описание сущности должно содержать сведения о типах данных атрибутов (целочисленный, действительный, символьный) и их длине (количество значащих цифр или максимальное количество символов), не должно включать сведений об объеме занятого пространства в байтах.

Внутренний уровень – физическое представление базы данных в компьютере. Этот уровень описывает, как информация хранится в базе данных. Этот уровень содержит описание структур данных и организации отдельных файлов, используемых для хранения данных в запоминающих устройствах. На физическом уровне определяются методы взаимодействия СУБД с операционной системой компьютера.

Общее описание БД называется схемой базы данных. Существует три различных типа схем базы данных, которые соответствуют трем уровням абстракции.

На самом верхнем уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных (рис. 6). На концептуальном уровне описание базы данных называют концептуальной схемой, а на самом нижнем уровне абстракции – внутренней схемой.

Внешнее представление 1

SNo	FName	LName	Age	Salary
Номер сотрудника	Имя	Фамилия	Возраст	Зарплата

Внешнее представление 2

StNo	Lname	BNo
Личный № сотрудника	Фамилия	Номер отдела

Рис.6 Внешнее представление

Концептуальный уровень

StaffNo	FName	LName	DOB Дата рождения	Salary	BranchNo

Внутренний уровень
Struct STAFF
Staff No integer (идентификатор)
Branch No integer
FName char (15)
LName char (15)
DOB date
Salary double–precision

Важно различать описание базы данных и саму базу данных. Описанием базы данных является схема БД, которая создается при проектировании и меняется достаточно

редко. Данные, содержащиеся в БД, могут меняться часто (например, при добавлении сведений о новом сотруднике).

Схема создается с помощью некоторого языка определения данных конкретной СУБД.

Основным назначением трехуровневой архитектуры является обеспечение независимости от данных, которая означает, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

Логическая независимость от данных – означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему.

Такие изменения концептуальной схемы как добавление или удаление новых сущностей, атрибутов или связей должны осуществляться без необходимости внесения изменений в уже существующие внешние схемы. Об этих изменениях должны знать только те, для которых они предназначены.

Физическая независимость от данных – означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему.

Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы.

Структура СУБД

В структуре типичной СУБД выделяются: ядро СУБД, компилятор языка базы данных (обычно SQL), подсистему поддержки времени выполнения, набор утилит.

Ядро СУБД является основной резидентной частью СУБД и предназначено для управления данными во внешней памяти, управления буферами оперативной памяти, управления транзакциями и журнализации. Выполнение названных функций обеспечивается входящими в состав ядра СУБД компонентами: менеджером данных, менеджером буферов, менеджером транзакций и менеджером журнала.

Менеджер данных осуществляет управление данными во внешней памяти. Эта функция поддерживает необходимые структуры внешней памяти для хранения данных, непосредственно входящих в базу данных, и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы).

Менеджер буферов управляет буферами оперативной памяти. СУБД обычно работают с базами данных, размеры которых существенно больше доступного объема оперативной памяти. Если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти.

Буферизация данных в оперативной памяти позволяет временно содержать данные в процессе их получения, передачи, чтения или записи в специальной области памяти. Эта процедура сглаживает скоростные и временные характеристики устройств.

Менеджер транзакций управляет объединением элементарных операций в транзакции для обеспечения целостности базы данных, управляет параллельно выполняющимися транзакциями и т.д. ***Транзакция*** – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Если транзакция успешно выполняется, СУБД фиксирует изменения БД, произведенные этой транзакцией во внешней памяти. В противном случае ни одно из этих изменений никак не отражается на состоянии БД.

Менеджер журнала управляет журнализацией. Журнал – это особая часть БД, недоступная пользователям СУБД, в которую поступают записи обо всех изменениях основной части БД.

При журнализации используется стратегия "упреждающей" записи в журнал (протокол Write Ahead Log - WAL). Стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. При соблюдении протокола WAL с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую. Интерфейс используется в программах, формируемых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах.

Компилятор языка БД преобразует операторы языка БД в выполняемую программу. Результат компиляции – выполняемая программа, представляется в машинных кодах или в выполняемом внутреннем машинно-независимом коде.

Подсистема поддержки времени выполнения используется для интерпретации внутреннего машинно-независимого кода при выполнении операторов программы.

Утилиты предназначены для таких процедур, которые неэффективно выполнять с использованием языка БД. К таким операциям относятся: загрузка и выгрузка БД, сбор статистики, глобальная проверка целостности БД и другие.

Контрольные вопросы:

1. В чем состоит основная цель СУБД
2. Методы описания схем БД в современных СУБД
3. Перечислите структуру данных СУБД

Лекция 6.
Методы организации целостности данных. Модели и структуры информационных систем.

Цель: изучить методы организации целостности данных, модели и структуры информационных систем

План занятия:

1. Методы обеспечения целостности системы защиты
2. Модели и методы организации данных
3. Организация данных

Методы обеспечения целостности системы защиты

Под целостностью подразумевается актуальность и непротиворечивость информации, ее защищенность от разрушения и несанкционированного изменения.

Методами обеспечения целостности являются:

- использование отказоустойчивых устройств;
- резервирование (дублирование) данных;
- разграничение прав доступа к программам и данным;
- организация антивирусной защиты;
- контроль целостности программ и данных.

Использование отказоустойчивых устройств.

Под надежностью понимается свойство системы выполнять возложенные на нее задачи в определенных условиях эксплуатации. При наступлении отказа компьютерная

система не может выполнять все предусмотренные документацией задачи, т.е. переходит из исправного состояния в неисправное. Если при наступлении отказа компьютерная система способна выполнять заданные функции, сохраняя значения основных характеристик в пределах, установленных технической документацией, то она находится в работоспособном состоянии.

Резервирование (дублирование) данных.

По времени восстановления информации методы дублирования могут быть разделены на:
- оперативные;
- неоперативные.

К оперативным методам относятся методы дублирования информации, которые позволяют использовать дублирующую информацию в реальном масштабе времени. Это означает, что переход к использованию дублирующей информации осуществляется за время, которое позволяет выполнить запрос на использование информации в режиме реального времени для данной КС. Все методы, не обеспечивающие выполнения этого условия, относят к неоперативным методам дублирования.

По используемым для целей дублирования средствам методы дублирования можно разделить на методы, использующие:
- дополнительные внешние запоминающие устройства (блоки);
- специально выделенные области памяти на несъемных машинных носителях;
- съемные носители информации.

По числу копий методы дублирования делятся:
- на одноуровневые;
- многоуровневые.

Как правило, число уровней не превышает трех.

По степени пространственной удаленности носителей основной и дублирующей информации методы дублирования могут быть разделены на следующие методы:
- сосредоточенного дублирования;
- рассредоточенного дублирования.

Для определенности целесообразно считать методами сосредоточенного дублирования такие методы, для которых носители с основной и дублирующей информацией находятся в одном помещении. Все другие методы относятся к рассредоточенным.

В соответствии с процедурой дублирования различают методы:
- полного копирования;
- зеркального копирования;
- частичного копирования;
- комбинированного копирования.

При полном копировании дублируются все файлы.

При зеркальном копировании любые изменения основной информации сопровождаются такими же изменениями дублирующей информации. При таком дублировании основная информация и дубль всегда идентичны.

Частичное копирование предполагает создание дублей определенных файлов, например, файлов пользователя. Одним из видов частичного копирования, получившим название инкрементного копирования, является метод создания дублей файлов, измененных со времени последнего копирования.

Комбинированное копирование допускает комбинации, например, полного и частичного копирования с различной периодичностью их проведения.

Наконец, по виду дублирующей информации методы дублирования разделяются:
- на методы со сжатием информации;
- методы без сжатия информации.

Модели и методы организации данных

Информационные модели

Информационная модель представляет собой формализованное описание на языке информатики части реального мира (предметной области), подлежащей изучению для организации управления и автоматизации социально-экономических процессов.

Информационная модель должна удовлетворять ряду требований: управление потоками событий; идентификация сообщений; обработка ошибок; возможность расширения; простота использования и управления (включая транспортабельность передачи сообщений); «мягкий» отказ; возможность расширения.

Информационный объект – это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных информационных элементов (атрибут/реквизит, отношение).

Информационный объект имеет множество реализаций (экземпляров), каждый из которых представлен совокупностью конкретных значений атрибутов и идентифицируется значением ключевого атрибута (ключа). Остальные атрибуты являются описательными.

Информационная модель позволяет зафиксировать концептуальное устройство предметной области, иерархию понятий, свойств и структуру объектов.

Информационные объекты одного реквизитного состава и структуры образуют классы объектов, которым также присваивается уникальное имя («Студент», «Сессия», «Стипендия»).

Модель данных – это совокупность структурированных данных и операций их обработки.

Построение информационной модели базируется на описании документооборота и алгоритмов решения прикладных задач. Моделирование возможно, если создано формализованное описание, учитывающее основные закономерности процессов и действующие факторы.

Решение по конфигурации модели принимается с учетом объемных и временных характеристик потоков информации, рациональных маршрутов ее движения.

Локализация информации осуществляется с учетом:

- класса задач, решаемых с использованием этой информации;
- круга соответствующих пользователей;
- места хранения.

Концептуальная модель предметной области служит для описания ее объектов и отношений между ними:

Мк =

где А – множество объектов;

R – множество отношений между объектами.

Информационно-логическая (инфологическая) модель предметной области отражает предметную область в виде совокупности информационных объектов и их структурных связей.

Этапы построения информационной модели:

- идентификация пользователей и сопряженных организаций;
- идентификация областей принятия решений;
- определение области принятия решений;
- разработка описательной системы модели;
- разработка нормативной системы модели;
- разработка согласованной модели системы;
- построение и описание алгоритма принятия решений;
- определение информационных потребностей.

Технологию информационного моделирования можно представить следующим образом.

Первый шаг. Агрегированный структурный анализ:

Назначение и цели организации.

Операционная часть.

Структурная конфигурация.

Второй шаг. Функциональный анализ:

Основные функциональные стратегии, цели и показатели работы.

Основные свойства, используемые для интеграции (планирование и контроль, инструменты связи, система принятия решений).

Третий шаг. Детализированный анализ организационных функций:

Функциональные цели и показатели эффективности функционирования для поддерживаемых целей.

Функциональные единицы и структуры.

Функциональные системы.

Четвертый шаг. Анализ управленческих функций, поддерживаемых системой:

Категории видов управленческой деятельности.

Роли руководителей по основным видам деятельности: описать обязанности в пределах организационных функций или процессов; определить прямые обязанности.

Показатели эффективности для управленческих функций.

Идентифицировать действия, которые нужно поддерживать по каждому виду управленческой деятельности: определить цели и природу действий; определить природу проблем и природу процесса решения проблем; описать технику и ресурсы, необходимые для действий; описать влияние человеческих факторов на действия.

Пятый шаг.Определить характеристики для поддержки управленческих функций:

- информационные характеристики и содержание;
- вид необходимого преобразования информации;
- характеристики сообщений с точки зрения руководителей.

Уровни моделирования

Организационный уровень заключается в разработке организационных мероприятий и нормативных документов, обеспечивающих функционирование системы.

Информационная модель системы организационного уровня должна удовлетворять следующим требованиям:

- многократное использование любых наборов данных, содержащихся в динамической модели любого узла, всеми пользователями системы;
- однократный ввод оперативных данных;
- минимальная избыточность за счет развитой системы идентификации содержания информации и связей между узлами системы;
- физическая независимость данных, обеспечивающая возможность изменения способов физического хранения данных, а также замены внешних запоминающих устройств без значительной модификации программного обеспечения;
- логическая независимость данных, предусматривающая возможность добавления новых элементов данных и расширения общих логических структур информации без модификации программного обеспечения;
- простота использования модели, позволяющая применять языки запросов высокого уровня, которые обеспечивают возможность получения данных пользователями системы без необходимости разработки ими специальных программ;
- пользователи должны иметь возможность легкого получения информации о том, какие данные имеются в их рассмотрении, используя словари данных, определяющие элементы хранимой информации и методы ее получения, а также различные средства помощи;
- модель должна обеспечивать требуемую скорость удовлетворения запросов пользователей на запрашиваемые данные с помощью совершенных систем адресации, механизмов доступа и поиска данных;

- модель должна обеспечивать требуемый уровень контроля достоверности и целостности хранимой и используемой информации;
- модель должна обеспечивать требуемый уровень сохранности и защищенности данных от физического разрушения, несанкционированного доступа и использования, а также средства эффективного и своевременного восстановления работоспособности при сбоях и отказах;
- при подготовке и использовании информации должны применяться методы счетного и логического контроля, автоматического обнаружения и исправления ошибок.

Концептуальный уровень соответствует логическому аспекту представления об информации предметной области в интегрированном виде.

Функциональный уровень обеспечивает решение прикладных задач, требующих предварительного анализа информации.

Информационный уровень обеспечивает формирование информационных объектов (количественное и качественное описание), между которыми установлены связи, позволяющие осуществлять поиск и выбор требуемой информации в соответствии с реализуемыми функциями.

В основе моделирования лежит специфицирование (формализованное описание) предметной области. Специфицирование состоит в описании входной и выходной информации, планово-отчетных показателей, нормативно-справочной информации, оперативной информации, процедур, предназначенных для принятия решений руководителями всех уровней, словарей и каталогов пользователей.

Схема специфицирования предметной области:
- обозначение работы (код, наименование, идентификатор);
- проблемно или пооперационно организованная функция;
- обеспечивающая процедура.

При моделировании используется блочный принцип.

Функциональные блоки описывают функции в соответствии с этапами работы и подразделениями-исполнителями (функциональный граф).

Информационные блоки являются информационными описаниями выполняемых функций (информационный граф). Они состоят из функционально-ориентированных наборов данных, являющихся основой построения информационных баз.

Спецификация функционально-ориентированных наборов данных:
- обозначение функционально-ориентированных наборов данных;
- объемные характеристики (атрибуты, уровень, хранение, объемы);
- психологические характеристики (обработка, потребление, пользователь, режим работы, стоимость).

Между информационными блоками существуют логические связи, позволяющие осуществлять поиск и выбор требуемой информации в соответствии с реализуемыми функциями.

В прикладной системе можно выделить следующие уровни организации информации: модели процессов, документов, вычислений и данных, а также экземпляры хранимых записей.

Монитор управления обеспечивает интерфейс между моделями процессов и моделями документов. Модели процессов задаются с помощью функций и предикатов, что позволяет синхронизировать процедуры обработки информации, инициируемые пользователем.

Блок общения обеспечивает интерфейс между моделями документов, вычислений и данных. Модели документов имеют иерархическую структуру и содержат наряду с именами информационных областей (доменов) связи, существующие между ними для организации логического контроля и обработки вводимых и выводимых данных.

Блок управления вычислениями обеспечивает интерфейс между моделью вычислений и моделью данных. Модель вычислений задает информационные связи,

существующие между процедурами обработки данных, позволяя автоматически генерировать управляющие программы для преобразования одних данных в другие.

Блок управления данными обеспечивает интерфейс между записями и моделью данных, которая представляет собой перечень имен документов с указанием функциональных связей между ними и позволяет организовать ввод, корректировку и поиск хранимых данных. Необходимый интерфейс между физическими записями и уровнем хранимых записей обеспечивается использованием методов доступа, которые позволяют представить структуры памяти в виде совокупности хранимых файлов.

Организация данных

Модель данных – это совокупность структурированных данных и операций их обработки.

Модели данных классифицируют по структурам.

Простые списковые.

Содержат списки индексов для множества записей. Индекс включает ключ записи и соответствующий адрес (поэтому эти структуры еще называют адресными списками).

Цепные.

Каждая запись, кроме собственного адреса, содержит адрес следующей за ней записи (ссылку). Могут быть незамкнутыми и замкнутыми (кольцевыми).

Иерархические.

Объединяют наборы разнотипных записей, допускающих всевозможные сочетания между собой. Описываются с помощью служебных записей для вершин (имя, дуги) и для дуг (имя, источник, приемник, прочие дуги источника). Могут быть древовидными и сетевыми.

Реляционные.

Объединяют наборы однотипных записей, описываемых с помощью двумерных таблиц (строка-кортеж, столбец-домен).

Модели данных используют различные методы доступа:

- последовательный;
- прямой (индексный);
- индексно-последовательный.

Файл – это совокупность экземпляров записей одной структуры. Через файл осуществляется обращение к данным во внутреннем (машинном) представлении.

Объект характеризуется записью. Запись характеризуется полем (атрибут может иметь несколько полей). Поле характеризуется описанием (реквизитом). Поле, каждое значение которого однозначно определяет соответствующую запись, называется ключевым полем (первичный или простой ключ).

В правильно построенной реляционной базе данных в каждой таблице есть один или несколько столбцов, значения в которых во всех строках разные. Этот столбец называется первичным ключом таблицы.

Если запись однозначно определяется значениями нескольких полей, то используется составной ключ (или вторичный).

Первичный ключ для каждой строки таблицы является уникальным, поэтому в таблице с первичным ключом нет двух одинаковых строк.

Таблица, в которой все строки отличаются друг от друга, в математических терминах называется отношением. Именно этому термину реляционные базы данных и обязаны своим названием, поскольку в их основе лежат отношения (таблицы с отличающимися друг от друга строками).

Столбец одной таблицы, значения в котором совпадают со значениями столбца, являющегося первичным ключом другой таблицы, называется внешним ключом.

Внешние ключи выполняют роль поисковых или группировочных признаков.

Внешний ключ, как и первичный ключ, тоже может представлять собой комбинацию столбцов. На практике внешний ключ всегда будет составным (состоящим из нескольких столбцов), если он ссылается на составной первичный ключ в другой таблице. Очевидно, что количество столбцов и их типы данных в первичном и внешнем ключах совпадают.

Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей.

Контрольные вопросы:

1. Опишите методы обеспечения целостности системы защиты
2. Что относится к моделям и методам организации данных
3. Принцип организации данных

Лекция 7.
Современные инструментальные средства проектирования схемы базы данных

Цель: рассмотреть современные инструментальные средства проектирования схемы базы данных

План занятия:

1. CASE-средства проектирования баз данных
2. Проектирование баз данных с помощью CASE-средств
3. Классификация CASE-средств

При выборе технологии построения информационной системы, содержащей в своем составе базу данных, нужно тщательно оценивать и прогнозировать ее потенциальные потребности в средствах управления данными. Конечно, любую информационную систему можно основывать на использовании промышленной, большой и мощной СУБД (такой, как например Oracle). Но вполне может оказаться так, что в действительности приложение будет использовать доли процентов общих возможностей СУБД. Накладные расходы (затраты на дополнительную аппаратуру, лицензирование дорогостоящего программного продукта, увеличение общего времени выполнения операций) могут оказаться неоправданными.

При разработке информационных систем для локальных вычислительных сетей с использованием технологии клиент/сервер оправданно и целесообразно в качестве СУБД применять свободно распространяемую СУБД FireBird, которую можно устанавливать практически на компьютеры с любой платформой (Unix, Linux, Windows и пр.). Эта СУБД, для своей установки, не требует покупки специального сервера (так например, СУБД MS SQL Server требует для своей установки сервер Windows Server 2003) и обладает большим количеством других преимуществ.

Обслуживание СУБД FireBird осуществляется с использованием инструментов администрирования IBExpert, которые для граждан и предприятий, использующих русскоязычную операционную систему Windows, также являются бесплатными.

Методология проектирования информационных систем, включающих базы данных, предусматривает проектирование базы данных и приложения для работы с ней. Данное методическое пособие посвящено вопросам разработки баз данных. Проектирование баз данных таких систем предполагает разбиение всего процесса на

несколько фаз, каждая из которых может состоять из нескольких этапов. На каждом этапе разработчик использует набор технических приемов позволяющих решать задачи данной стадии разработки.

Весь процесс разработки разделяется на три основные фазы: концептуальное, логическое и физическое проектирование.

Концептуальное проектирование БД необходимо для создания информационной модели предприятия (предметной области), не зависящей от каких- либо физических условий реализации. К последним относятся: тип СУБД, --- программ приложения, используемый язык программирования, конкретная вычислительная платформа и другие физические особенности реализации.

Логическое проектирование БД необходимо для создания информационной модели предприятия на основе разработанного концептуальной модели с учетом используемого типа СУБД (но не конкретной СУБД и прочих физических условий реализации).

Физическое проектирование БД - это процесс создания описания конкретной реализации БД с учетом особенностей выбранной СУБД. Эта фаза заканчивается созданием конкретной БД для создаваемого приложения, на основании разработанной ранее логической модели.

Проектирование базы данных может предусматривать выбор наиболее подходящего инструмента автоматизированного проектирования - CASE-инструмента (Computer-Aided Software Engineering).

В самом широком смысле термин CASE- инструмент применим к любым средствам автоматизированного проектирования и создания программ.

CASE- инструменты могут включать следующие компоненты:

- словарь данных, предназначенный для хранения информации в данных, используемых в создаваемом приложении;
- инструменты проектирования, обеспечивающие проведение анализа данных;
- инструменты разработки модели данных предприятия (модели бизнес-процесса), а также концептуальных и логических моделей данных;
- инструменты, позволяющие создавать прототипы приложений.

Использование CASE-инструментов позволяет существенно повысить производительность труда при разработке приложений баз данных.

CASE-средства проектирования баз данных

CASE-средства (Computer - Aided Software Engineering) - это методы и технологии, которые позволяют проектировать различные информационные системы (в частности, базы данных) и автоматизировать их создание. О проектировании баз данных, видах CASE-средств и об особенностях их применения.

CASE-средства--это автоматизированные средства, основанные на CASE-технологиях, позволяющие автоматизировать отдельные этапы жизненного цикла программного обеспечения.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее один или несколько процессов жизненного цикла ПО и обладающее следующими основными характерными особенностями:

- мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;
- интеграция отдельных компонент CASE-средств, обеспечивающая управляемость процессом разработки ИС;
- использование специальным образом организованного хранилища проектных метаданных (репозитория).

На сегодняшний день рынок программного обеспечения СНГ располагает следующими наиболее развитыми CASE-средствами:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE.Аналитик.

Проектирование баз данных с помощью CASE-средств

К ключевым понятиям проектирования баз данных относятся:

- ***CASE-технологии*** - программная основа CASE-средств, применяемая для разработки и поддержки процессов жизненных циклов ПО, используемых в моделировании данных и генерации схем баз данных. Чаще всего программные коды в CASE-технологиях пишутся на языке SQL;
- ***концептуальное проектирование*** - построение обобщенной, не имеющей конкретики, модели базы данных с описанием ее объектов и связей между ними;
- ***логическое проектирование*** - создание схемы базы данных с учетом специфики конкретной модели данных (но не конкретной СУБД). Например, для реляционной модели данных логическая схема БД будет содержать определенный набор таблиц и связей между ними;
- ***физическое проектирование*** - построение схемы базы данных под конкретную СУБД. При таком проектировании учитываются ограничения на именование объектов базы данных, ограничения на определенные типы данных, физические условия хранения данных в БД (разделение по файлам и устройствам), возможность доступа к БД.

При проектировании баз данных с помощью CASE-средств выделяются и анализируются определенные бизнес-процессы, для которых создается БД, определяются взаимосвязи их элементов, оптимизируется их инфраструктура. CASE-средства позволяют существенно сократить время на разработку БД и уменьшить количество ошибок в них.

Для создания баз данных под наиболее распространенные СУБД чаще всего используются следующие CASE-средства:

- ***ERwin (Logic Works)*** - CASE-инструмент для создания концептуальных и логических схем баз данных. Он позволяет редактировать различные наборы данных, представляя их в виде электронных таблиц, разрабатывать структуры баз данных, синхронизировать модели, скрипты и БД, настраивать шаблоны, выводить рабочую информацию в виде отчетов, строить удобные и понятные диаграммы, отображающие различные процессы в системе и взаимосвязи между ними;
- ***S-Designor (SDP)*** - графический CASE-инструмент для проектирования структуры реляционных БД. Он создает модели баз данных в два этапа - выстраивая концептуальную модель и затем преобразуя ее в физическую, причем в данном процессе разработки возможен как прямой, так и обратный переход между моделями. Данный инструмент позволяет проектировать базы данных под различные СУБД, в том числе под Oracle и MySQL;
- ***DataBase Designer (ORACLE)*** - интегрированная CASE-среда, которая позволяет анализировать предметную область создания БД, выполнять программирование и проектирование, проводить оценку и тестирование, осуществлять сопровождение, обеспечивать качество, управлять конфигурацией и проектом, разрабатывать и анализировать требования к информационной системе.

Классификация CASE-средств

В зависимости от того, на каком этапе проектирования баз данных используются CASE-средства, их относят к:

- ***CASE-средствам верхнего уровня***. Их задействуют на начальных этапах проектирования, когда требуется выполнить анализ поставленной задачи, поставить цели и определить приоритеты, представить необходимую информацию в виде диаграмм и деревьев решений;
- ***CASE-средствам нижнего уровня***. С помощью этих средств выполняются заключительные этапы проектирования БД, проводятся собственно проектирование, написание кода, тестирование и внедрение программного обеспечения поддержки информационных систем.
- ***интегрированным CASE-средствам***, которые дают возможность выполнять все этапы проектирования БД благодаря наличию функций верхнего и нижнего уровней.

Контрольные вопросы:

1. CASE-средства проектирования баз данных
2. Проектирование баз данных с помощью CASE-средств
3. Классификация CASE-средств

Лекция 8.
Технологии передачи и обмена данными в компьютерных сетях

Цель: изучить технологии передачи и обмена данными в компьютерных сетях

План занятия:

1. Основные понятия
2. Современные технологи и методы передачи данных

Основные понятия

Технологии передачи данных в своей работе используют (в зависимости от конкретной их реализации) различные физические интерфейсы.

Примечание: интерфейс это - физическая (или логическая) граница при взаимодействии нескольких независимых объектов - своеобразная прослойка между ними. Интерфейсы делятся на две категории:
- физические интерфейсы
- интерфейсы логические

Физический интерфейс это - конечный порт подключения (разъем с группой электрических контактов). Например - интерфейс сетевой карты компьютера. А пара портов, соединенная с помощью разъемов и кабеля называется линией (каналом) передачи данных.

Логический интерфейс - это набор правил (протокол), который определяет саму логику обмена данными между связанными линией (сетью) устройствами.

Организация передачи данных в компьютерной сети происходит в тесном взаимодействии этих двух интерфейсов: физический компонент (сетевая карта) и логический (ее драйвер).

Обязательным условием для успешной реализации любой из технологий передачи данных является присутствие в потоке данных дополнительного компонента - протокола передачи.

Протокол передачи на логическом уровне представляет собой набор правил, которые определяют обмен данными между различными приложениями или

устройствами. Эти правила задают единый способ передачи сообщений и обработки ошибок передачи. На физическом уровне протокол это - набор служебных данных, прикрепляющихся к основным пакетам (кадрам) информации, без которых просто невозможно эффективное взаимодействие в сети.

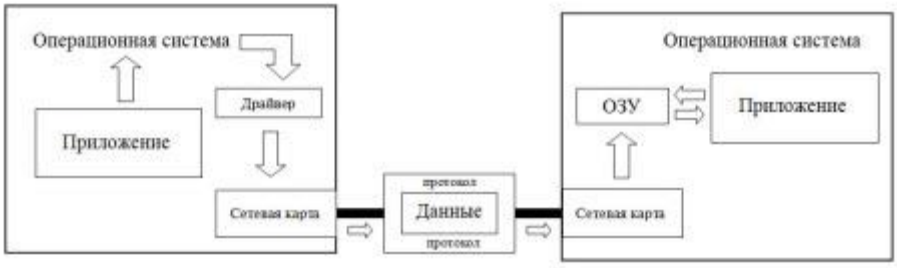
Протокол должен абстрагироваться (игнорировать) конкретную среду передачи, его задача - обеспечивать надежную связь между узлами в коммутационном облаке.

Давайте рассмотрим сам процесс организации передачи данных более подробно!

Сначала происходит вот что: приложение (программа) обращается к ОС за разрешением для сетевого взаимодействия с другим устройством (принтером, удаленным компьютером, камерой наблюдения и т.д.) Операционная система дает команду драйверу сетевой карты, который загружает в буфер карты первую порцию данных и инициирует работу интерфейса на передачу

На другом конце линии (сети) удаленное устройство принимает в буфер своей сетевой карты поступающие данные. После окончания передачи протокол проверяет нет ли в передаваемых частях (пакетах) данных ошибок (если надо запрашивает их повторную передачу) и загружает принятые данные из буфера карты в заранее зарезервированное пространство оперативной памяти. Оттуда уже конечное приложение (программа) извлекает информацию и работает с ней.

Вот - схема, для наглядности (кликабельно): СХЕМУ ЗАРИСОВАТЬ



На основании всего сказано выше, можно сделать такой вывод: технологии построения сети сводятся к тому, чтобы связать между собой удаленные устройства электрически и информационно! Т.е. - создать физическую среду передачи (кабель, беспроводная связь) и обеспечить общий протокол передачи данных по сети.

Современные технологи и методы передачи данных

Современные технологи и методы передачи данных, в большинстве случаев, основаны на клиент-серверном взаимодействии.

Клиент это - модуль (программа, служба, отдельный компьютер), служащий для формирования и передачи сообщений (запросов) к ресурсам удаленного устройства (серверу), с последующим приемом результатов от него и передачей их соответствующим приложениям на клиенте.

Сервер это - модуль (программа, служба...), который постоянно ожидает прихода из сети запросов от клиентов и обслуживающий (с участием локальной ОС) эти запросы.

Один сервер может обслуживать сразу множество клиентов. Например - веб-сервер «Apache» (Апач), который обеспечивает множественные подключения за день к нашему сайту «sebeadmin.ru» по протоколу «http». Вот - еще пример: база данных, с которой работают клиенты. На них установлены клиентские модули программ, которые подключаются к базе и поддерживают только графический интерфейс работы с ней. Все вычисления и обработка, при этом, происходят на сервере и с использованием его ресурсов.

Познакомимся еще с одним определением! Клиент-серверная составляющая, которая предоставляет доступ к какому-то ресурсу компьютера через сеть называется сетевой службой. Причем, каждая служба связана с определенным типом сетевых ресурсов.

Например: служба печати позволяет нам распечатывать документы на сетевом принтере, а файловая служба - получать доступ к данным, находящимся на удаленных компьютерах. Для серфинга по Интернету есть своя веб-служба, которая состоит из серверной части (веб-сервера) и клиентской (веб-браузера) пользователя (IE, Opera, Firefox и т.д.)

В свете всего сказанного выше, технологии передачи данных должны опираться не просто на операционные системы, а на сетевые ОС, которые предоставляют пользователю доступ к информационным и аппаратным ресурсам других компьютеров. Причем эти операционные системы, согласно изложенным выше определениям, также делятся на два больших класса: серверные и клиентские ОС.

Клиентские системы обращаются, в основном, с запросами к серверным компонентам других компьютеров а серверные компоненты серверной ОС предоставляют эти услуги. Конечно, на данный момент, практически любая современная ОС способна выполнять как роль клиента, так и сервера. Серверные системы просто изначально созданы из расчета обслуживания ими максимального количества обращений и обладают лучшей отказоустойчивостью (надежностью).

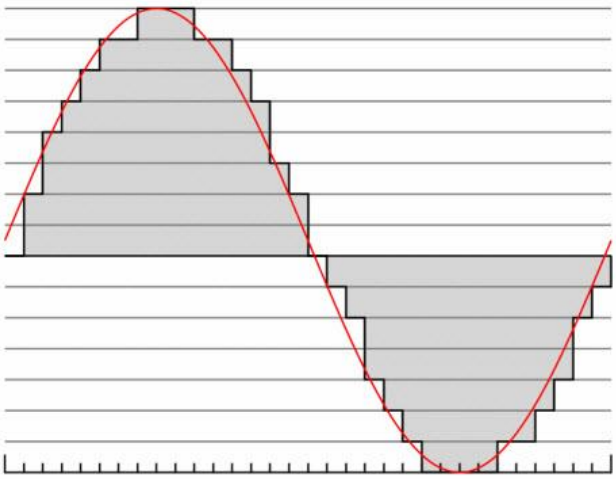
Современные (цифровые) технологии передачи сигнала связаны с его преобразованием (кодированием). Зачем нам это нужно? На то есть несколько причин:

Предотвращение ошибок передачи данных (за счет уверенного распознавания сигнала принимающей стороной)

Данные передаются быстрее (за счет более высокой плотности полезной информации в потоке)

Как видите, это - уже две весьма веские причины для того, чтобы уделить методам кодирования должное внимание :)

На фото ниже представлено два сигнала: аналоговый (красная линия) и цифровой (черные "ступеньки")



В данном случае аналоговая последовательность была оцифрована (дискретизирована) с определенной частотой. Чем выше будет частота дискритизации, тем меньший шаг будут иметь наши "ступеньки" и тем более похож будет оцифрованный сигнал на исходный (красный).

Похожие процессы происходят и при дискретизации (оцифровке) нашего голоса, снимаемого со входа микрофона звуковой картой компьютера.

В вычислительной технике используется двоичный код. Внутри системного блока компьютера это эквивалентно двум состояниям: наличию и отсутствию электрического напряжения (логический «ноль» или «единица»). Здесь - все просто: есть ток - "единица", нету - "ноль".

Современные технологии передачи данных позволяют производить кодирование сигнала и другими (более эффективными) способами. Но прежде, - еще одна небольшая классификация. По способу реализации процедура делится на:

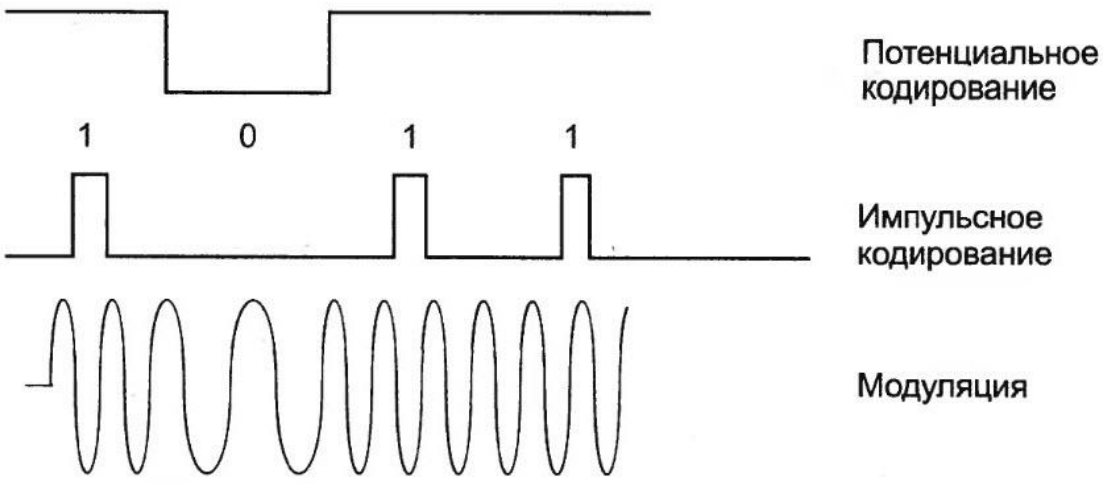
Физическое кодирование сигнала и - логическое (на более высоком уровне - поверх физического)

Давайте сначала обзорно рассмотрим первый пункт. Есть, к примеру, потенциальный способ кодирования, при котором единице соответствует один уровень напряжения (один потенциал), а нулю - другой. А при импульсном способе, для представления цифр используются импульсы разной полярности.

Для технологии кодирования определенная проблема при передаче данных состоит в том, что внешние (по отношению к самому компьютеру) линии передачи данных могут быть растянуты на большие расстояния и подвержены воздействию различных помех и наводок. Это приводит к искажению эталонных прямоугольных импульсов передачи сигнала и нужны новые (надежные) алгоритмы его кодирования и передачи.

В вычислительных сетях применяется как потенциальное, так и импульсное кодирование. Также применяется и такой способ передачи данных, как модуляция.

При модуляции дискретные данные передаются с помощью синусоидального сигнала той частоты, которую хорошо передает имеющаяся в распоряжении линия связи.



Первые два варианта преобразования применяются для линий высокого качества, а модуляция используется в каналах с сильными искажениями сигнала. Модуляция, к примеру, используется в глобальных сетях при передаче трафика через аналоговые телефонные каналы связи, которые были разработаны специально для передачи голоса (аналоговой составляющей) и поэтому плохо подходят для передачи цифровых импульсов.

На сам способ передачи оказывает влияние и такая вещь, как количество проводников (жил) в линиях связи. Для снижения их стоимости количество проводов, зачастую, снижается. При такой технологии передача данных осуществляется последовательно, а не параллельно (как это принято для линий связи внутри компьютера).

К способам кодирования на физическом уровне относятся такие алгоритмы, как NRZ (Non Return Zero), Манчестерский код (Manchester), MLT-3 (Multi Level Transmission) и ряд других.

Давайте пару слов скажем и о логическом кодировании. Как можно понять из названия, оно осуществляется по верху физического (накладываясь на него) и служит для обеспечения дополнительной надежности при передаче данных. Каким же образом?

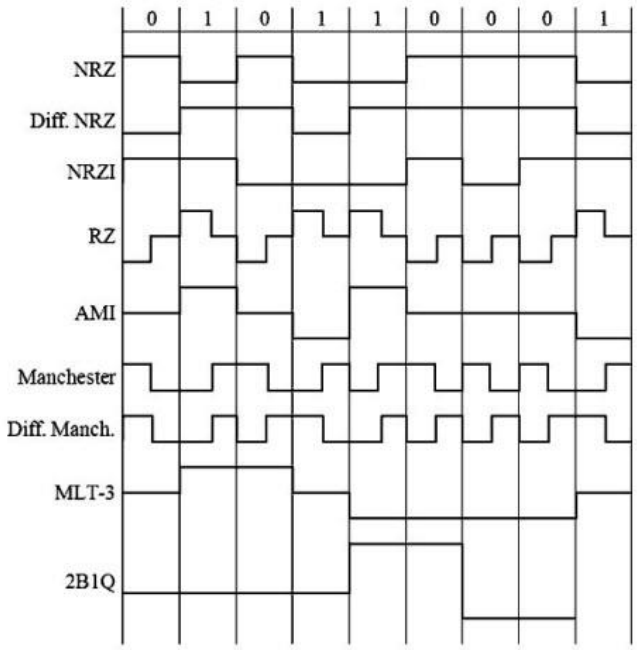
Например: если характер передаваемого сигнала долгое время не изменяется (при передаче длинных последовательностей логических нулей или единиц) приемник может ошибиться при считывании очередного бита информации. Он просто не сможет разложить общий поток данных на отдельные составляющие и, как следствие, - правильно собрать в своем буфере из них исходную структуру.

Логическое кодирование (которому подвергается исходная последовательность данных) внедряет в длинные последовательности бит свои биты с противоположным значением, или - вообще заменяет их другими последовательностями. Кроме того, оно позволяет улучшить спектральные характеристики сигнала, в целом - упростить его расшифровку, а кроме того - передавать в общем потоке дополнительные служебные сигналы управления.

В основном, для логического преобразования применяются три технологии:

- вставка бит (bit stuffing)
- избыточное кодирование
- скремблирование

Коротко отчитаюсь следующим скриншотом:



На нем Вы можете видеть, как выглядит один и тот же сигнал, при наложении на него различных алгоритмов:

Технологии передачи данных имеют еще ряд проблем, с которыми приходится бороться. И одна из них - проблема взаимной синхронизации передатчика одного компьютера и приемника другого. Согласитесь, что сложно будет разобраться в потоке данных, если два устройства начнут генерировать его одновременно "навстречу" друг другу. Начнется бардак! :)

Внутри корпуса компьютера эта проблема решается просто, так как все устройства: оперативная память, видеокарта, центральный процессор и т.д. синхронизируются от общего тактового генератора, расположенного на материнской плате.

Проблема же синхронизации удаленных компьютеров может решаться разными способами: путем обмена специальными тактовыми синхроимпульсами или же - передачей служебных данных, не имеющих отношения к основному потоку информации. Один из стандартных приемов, служащий для повышения надежности передачи это - подсчет контрольной суммы каждого байта (блока байтов) и передача этого значения принимающей стороне.

Примечание: контрольная сумма это - некоторое значение, рассчитанное путем "наложения" на данные определённого алгоритма и используемое для проверки их целостности при передаче. Контрольные суммы могут использоваться для быстрого сравнения двух наборов данных на их идентичность. Отличающиеся данные будут иметь разные контрольные суммы.

Еще одна технология подтверждения целостности данных это - обмен между взаимодействующими устройствами служебными сигналами-квитанциями, подтверждающими правильность приема. Зачастую эта функция по умолчанию включается в сам протокол сетевого взаимодействия.

Технологии передачи данных подразумевают передачу информации от одного компьютера к другому - в обеих направлениях. Даже в том случае, когда нам кажется, что мы только принимаем данные (например - скачиваем музыку), то на самом деле - обмен идет в двух направлениях. Просто есть основной поток данных (который интересует нас - музыка) и вспомогательный (служебный), идущий в обратном направлении, образуемый квитанциями об успешной (или не успешной) передаче.

В зависимости от того, могут ли они передавать данные в обоих направлениях или нет, физические каналы делятся на несколько видов:

Дуплексный канал - обеспечивает одновременную передачу информации в обоих направлениях Дуплекс может состоять из двух независимых физических сред (один проводник на прием, второй - на передачу). Возможен и вариант, при котором одна среда используется для обеспечения дуплексного режима работы. В этом случае на клиентах применяются дополнительные алгоритмы выделения каждого потока данных из общего массива информации.

Полудуплексный канал - также обеспечивает передачу в обоих направлениях, но не одновременно, а - по очереди. Т.е. в течение определенного времени данные передаются в одном направлении, а затем - в обратном.

Симплексный канал - позволяет передавать информацию только в одном направлении. Дуплексный может состоять из двух симплексных каналов.

Контрольные вопросы

1. Основные понятия
2. Современные технологи и методы передачи данных

Лекция 9.
Введение в SQL и его инструментарий

Цель: рассмотреть SQL и его инструментарий

План занятия:

1. SQL и его история
2. Описание основных операторов SQL

SQL - язык манипулирования данными в реляционной базе данных

SQL и его история

Единственным средством общения и *администраторов баз данных*, и проектировщиков, и разработчиков, и пользователей с реляционной базой данных является *структурированный язык* запрос SQL (Structured *Query Language*). SQL есть полнофункциональный язык манипулирования данными в реляционных базах данных. В настоящее время он является общепризнанным, стандартным интерфейсом для реляционных баз данных, таких как Oracle, *Informix, Sybase*, DB/2, MS SQL Server и ряда других (стандарты ANSI и ISO). SQL - непроцедурный язык, который предназначен для обработки множеств, состоящих из строк и колонок таблиц реляционной базы данных. Хотя существуют его расширения, допускающие процедурную обработку. Проектировщики баз данных используют SQL для создания всех физических объектов реляционной базы данных.

Теоретические основы SQL были заложены в известной статье [Кодд], положившей начало развитию теории реляционных БД. Первая практическая реализации была выполнена в исследовательских лабораториях фирмы IBM Chamberlin D.D. и Royce R.F. Промышленное применение SQL было впервые реализовано в *СУБД Ingres*. Одной из первых промышленных реляционных СУБД является Oracle. По сути дела, реляционная СУБД - это программное обеспечение, которое управляет работой реляционной базы данных.

Первый международный стандарт языка SQL был принят в 1989 г. (SQL-89). В конце 1992 г. был принят новый международный стандарт SQL-92. В настоящее время

большинство производителей реляционных СУБД используют его в качестве базового. Однако работы по стандартизации языка SQL далеки от завершения и уже разработан проект стандарта SQL-99, который вводит в обиход языка понятие объекта и разрешает на него ссылаться в операторах SQL. В исходном варианте SQL не было команд управления потоком данных, они появились в недавно принятом стандарте ISO/IEC 9075-5: 1996 дополнительной части SQL.

Каждой конкретной СУБД соответствует своя собственная реализация SQL, в целом поддерживающая определенный стандарт, но имеющая свои особенности. Эти реализации называются диалектами. Так, стандарт ISO/IEC 9075-5 предусматривает объекты, называемые постоянно хранимыми модулями или *PSM*-модулями (Persistent Stored Modules). В СУБД Oracle расширение PL/SQL является аналогом указанного выше расширения стандарта 1.

Описание основных операторов SQL

SQL состоит из набора команд манипулирования данными в реляционной базе данных, которые позволяют создавать объекты реляционной базы данных, модифицировать данные в таблицах (вставлять, удалять, исправлять), изменять схемы отношений базы данных, выполнять вычисления над данными, делать выборки из базы данных, поддерживать безопасность и целостность данных.

Весь набор команд SQL можно разбить на следующие группы:

- команды определения данных (*DDL* - Data Defininion Language);
- команды манипулирования данными (*DML - Data Manipulation Language*);
- команды выборки данных (DQL - Data *Query Language*);
- команды управления транзакциями;
- команды управления данными.

При выполнении каждая команда SQL проходит четыре фазы обработки:

- фаза *синтаксического разбора*, которая включает проверку синтаксиса команды, проверку имен таблиц и колонок в базе данных, а также подготовку исходных данных для оптимизатора;
- фаза оптимизации, которая включает подстановку действительных имен таблиц и колонок базы данных в представление, идентификацию возможных вариантов выполнения команды, определение стоимости выполнения каждого варианта, выбор наилучшего варианта на основе внутренней статистики;
- фаза генерации исполняемого кода, которая включает построение выполняемого кода команды;
- фаза выполнения команды, которая включает выполнение кода команды.

В настоящее время оптимизатор является составной частью любой промышленной реализации SQL. Работа оптимизатора основана на сборе статистики о выполняемых командах и выполнении эквивалентных алгебраических преобразований с отношениями базы данных. Такая статистика сохраняется в системном каталоге базы данных. Системный каталог является словарем данных для каждой базы данных и содержит информацию о таблицах, представлениях, индексах, колонках, пользователях и их *привилегиях доступа*. Каждая база данных имеет свой системный каталог, который представляет совокупность предопределенных таблиц базы данных.

Таблица 8.1 содержит список команд SQL в соответствии с принятым стандартом, за исключением некоторых практически не используемых в диалектах команд.

Таблица 8.1. Типичный список команд SQL

Команда	**Описание**
Команды определения данных объектов	
ALTER TABLE	Изменяет описание таблицы (схему отношения)
CREATE EVENT	Создает событие таймера в базе данных

CREATE INDEX	Создает индекс для таблицы
CREATE SEQUENCE	Создает последовательность
CREATE TABLE	Определяет таблицу
CREATE TABLESPACE	Создает табличное пространство
CREATE TRIGGER	Создает триггер в базе данных
CREATE VIEW	Определяет представление на таблицах
DROP INDEX	Физически удаляет индекс из баз данных
DROP SEQUENCE	Удаляет последовательность
DROP TABLE	Физически удаляет таблицу из базы данных
DROP TABLESPACE	Удаляет табличное пространство
DROP VIEW	Удаляет представление
Команды манипулирвания данными	
DELETE	Удаляет одну или более строк из таблицы базы данных
INSERT	Вставляет одну или более строк в таблицу базы данных
UPDATE	Обновляет значения колонок в таблице базы данных
Команды выборки данных	
SELECT	Выполняет запрос на выборку данных из таблиц и представлений
UNION	Объединяет в одной выборке результаты выполнения двух или более команд SELECT
Команды управления транзакциями	
COMMIT	Завершает транзакцию и физически актуализирует состояние базы данных
ROLLBACK	Завершает транзакцию и возвращает текущее состояние базы данных на момент последней завершенной транзакции и контрольной точки
SAVEPOINT	Назначает контрольную точку внутри транзакции
Команды управления данными	
ALTER DATABASE	Изменяет группы хранения или журналы транзакций
ALTER DBAREA	Изменяет размер областей хранения базы данных
ALTER PASSWORD	Изменяет пароль для доступа к базе данных
ALTER STOGROUP	Изменяет состав областей хранения в группе хранения
CHECK DATABASE	Проверяет целостность базы данных
CHECK INDEX	Проверяет целостность индекса
CHECK TABLE	Проверяет целостность таблицы и индекса
CREATE DATABASE	Физически создает базу данных
CREATE DBAREA	Создает область хранения базы данных
CREATE STOGROUP	Создает группу хранения

CREATE SYSNONYM	Создает синоним для таблицы или представления
DEINSTALL DATABASE	Делает базу данныхх недоступной пользователям вычислительной сети
DROP DATABASE	Физически удаляет базы данных
DROP DBAREA	Физически удаляет область хранения данных
DROP STOGROUP	Удаляет группу хранения
GRANT	Определяет привилегии пользователей и разграничение доступа к базе данных
INSTALL DATABASE	Делает базу данных доступной пользователям вычислительной сети
LOCK DATABASE	Блокирует текущую активную базу данных
REVOKE	Отменяет привилегии пользователей и разграничения доступа к базе данных
SET DEFAULT STOGROUP	Определяет группу хранения по умолчанию
UNLOCK DATABASE	Деблокирует текущую активную базу данных
UPDATE STATISTIK	Обновляет статистику для базы данных
Другие команды	
COMMENT ON	Размещает в системном каталоге комментарии к описанию объектов БД
CREATE SYNONYM	Определяет в системном каталоге альтернативные имена для таблиц и представлений БД
DROP SYNONYM	Удаляет из системного каталого альтернативные именя для таблиц и представлений БД
LABEL	Изменяет метки системных описаний
ROWCOUNT	Вычисляет число строк в таблице БД

Набор команд SQL, перечисленный в таблице, не является полным. Этот список приведен, чтобы вы составили впечатление о возможностях SQL в целом. Для получения полного списка команд следует обратиться к соответствующему руководству для конкретной СУБД. Следует помнить, что SQL является единственным средством общения всех категорий пользователей с реляционными базами данных.

Встроенные функции SQL и их использование в запросах

Арифметические функции

SQL поддерживает полный набор арифметических операций и математических функций для построения арифметических выражений над колонками базы данных (+, -, *, /, ABS, LN, SQRT и т.д.). Список основных встроенных математических функций дан ниже в таблице 8.2.

Таблица 8.2. Математические функции SQL

Математическая функция	**Описание**
ABS(X)	Возвращает абсолютное значение числа X
ACOS(X)	Возвращает арккосинус числа X
ASIN(X)	Возвращает арксинус числа X
ATAN(X)	Возвращает арктангенс числа X
COS(X)	Возвращает косинус числа X

EXP(X)	Возвращает экспоненту числа X
SIGN(X)	Возвращает -1, если X0
LN(X)	Возвращает натуральный логарифм числа X
MOD(X,Y)	Возвращает остаток от деления X на Y
CEIL(X)	Возвращает наименьшее целое, большее или равное X
ROUND(X,n)	Округляет число X до числа с n знаками после десятичной точки
SIN(X)	Возвращает синус числа X
SQRT(X)	Возвращает квадратный корень числа X
TAN(X)	Возвращает тангенс числа X
FLOOR(X)	Возвращает наибольшее целоеб меньшее или равное X
LOG(a,X)	Возвращает логарифм числа X по основанию A
SINH(X)	Возвращает гиперболический синус числа X
COSH(X)	Возвращает гиперболический косинус числа X
TANH(X)	Возвращает гиперболический тангенс числа X
TRANC(X,n)	Усекает число X до числа с n знаками после десятичной точки
POWER(A,X)	Возвращает значение A, возведенное в степень X

Набор встроенных функций может изменяться в зависимости от версии СУБД одного производителя и также в СУБД различных производителей. Так, например, в СУБД SQLBase, Centure Inc. есть функция @ATAN2(X,Y), которая возвращает арктангенс Y/X, но отсутствует функция SIGN(X).

Арифметические выражения необходимы для получения данных, которые непосредственно не сохраняются в колонках таблиц базы данных, но значения которых необходимы пользователю. Допустим, что вам необходим список служащих, показывающий выплату, которую получил каждый служащий с учетом премий и штрафов.

```
SELECT ENAME, SAL, COMM, FINE, SAL + COMM - FINE
FROM EMPLOYEE
ORDER BY DEPNO;
```

Арифметическое выражение SAL + COMM - FINE выводится как новая колонка в результирующей таблице, которая вычисляется в результате выполнения запроса. Такие колонки называют еще производными (вычисляемыми) атрибутами или полями.

Функции обработки строк

SQL предоставляет вам широкий набор функций для манипулирования со строковыми данными (конкатенация строк, CHR, LENGTH, INSTR и другие). Список основных функций для обработки строковых данных приведен в таблице 8.3.

Таблица 8.3. Функции SQL для обработки строк

Функция	**Описание**
CHR(N)	Возвращает символ ASCII кода для десятичного кода N
ASCII(S)	Возвращает десятичный ASCII код первого символа строки
INSTR(S2.S1.pos[,N]	Возвращает позицию строки S1 в строке S2 большую или равную pos.N - число вхождений
LENGHT(S)	Возвращает длину строки
LOWER(S)	Заменяет все символы строки на прописные символы
INITCAP(S)	Устанавливает первый символ каждого слова в строке на

	заглавный, а остальные символы каждого слова - на прописные
SUBSTR(S,pos,[,len])	Выделяет в строке S подстроку длиной len, начиная с позиции pos
UPPER(S)	Преобразует прописные буквы в строке на заглавные буквы
LPAD(S,N[,A])	Возвращает строку S, дополненную слева симолами A до числа символов N. Символ - наполнитель по умолчанию - пробел
Rpad(S,N[,A])	Возвращает строку S, дополненную справа симолами A до числа символов N. Символ - наполнитель по умолчанию - пробел
LTRIM(S,[S1])	Возвращает усеченную слева строку S. Символы удаляются до тех пор, пока удаляемый символ входит в строку - шаблон S1 (по умолчанию - пробел)
RTRIM(S,[S1])	Возвращает усеченную справа строку S. Символы удаляются до тех пор, пока удаляемый символ входит в строку - шаблон S1 (по умолчанию - пробел
TRANSLATE(S,S1,S2)	Возвращает строку S, в которой все вхождения строки S1 замещены строкой S2. Если S1 S2, то символы, которым нет соответствия, исключаются из результирующей строки
REPLACE(S,S1,[,S2])	Возвращает строку S, для которой все вхождения строки S1 замещены на подстроку S2. Если S2 не указано, то все вхождения подстроки S1 удаляются из результирующей строки
NVL(X,Y)	Если X есть NULL, то возвращает в Y либо строку, либо число, либо дату в зависимости от исходного типа Y

Названия одних и тех же функций могут отличаться в различных СУБД. Так, например, функция СУБД Oracle SUBSTR(S, pos, [, len]) в СУБД SQLBase называется @SUBSTRING(S, pos, len). В СУБД SQLBase имеются функции, которых нет в СУБД Oracle (см. таблицу ниже, где приведен список таких функций).

Таблица 8.4. Строковые функции СУБД SQLBase, отличающиеся от строковых функций СУБД Oracle

Функция	**Описание**
@EXACT(S1,S2)	Возвращает результат сравнения двух строк
@LEFT(S,len)	Возвращает левую подстроку длиной len
LENGTH(S)	Возвращает длину строки
@MID(S, pos, len)	Возвращает подстроку указанной длины, начиная с позиции pos
@REPEAT(S,n)	Повторяет строку S n раз
@REPLACE(S1,pos,len,S2)	Замещает с позиции pos len символов в строке S2 символами строки S1
RIGHT(S,len)	Возвращает правую подстроку S длиной len
@SCAN(S,pat)	Возвращает позицию подстроки pat в строке S
@STRING(X,scale)	Возвращает символьное представление числа с указанным масштабом scale
@TRIM(S)	Удаляет пробелы в строке справа и слева
@VALUE(S)	Преобразует символьное представление числа в

	числовое значение

Можно использовать функцию INITCAP, чтобы при получении списка имен служащих фамилии всегда начинались с заглавной буквы, а все остальные были прописными.

```
SELECT INITCAP(ENAME)
FROM EMPLOYEE
ORDER BY DEPNO
```

Контрольные вопросы:

1. SQL и его история
2. Описание основных операторов SQL

Лекция 10.
Установка и настройка SQL-сервера

Цель: изучить способ установки и настройки SQL- сервера

План занятия:

1. Служба SQL Server Agent: назначение, автоматический запуск от имени доменной учетной записи, роль базы данных MSDB
2. Автоматизация административных операций средствами SQL Server Agent

Служба SQL Server Agent: назначение, автоматический запуск от имени доменной учетной записи, роль базы данных MSDB

В предыдущих модулях было рассмотрено выполнение основных административных операций на SQL Server. Однако, как знают опытные администраторы, многие административные операции на сервере являются повторяющимися. Например, очень часто изо дня в день приходится производить:

q - резервное копирование;
q - проверку целостности баз данных;
q - загрузку и выгрузку данных;
q - перестроение индексов и дефрагментацию,

а также многие другие действия, набор которых зависит от конкретной задачи.

Кроме того, в некоторых ситуациях необходимо сделать так, чтобы администратор был немедленно извещен о каких-то важных событиях на сервере (например, внесены изменения в важные таблицы, выявлена ошибка при проверке целостности данных, возникли проблемы при массовой загрузке и т. п.).

И выполнение определенных действий по расписанию, и отслеживание событий рекомендуется автоматизировать. Чем более опытен и квалифицирован администратор, тем большее количество повседневных операций он старается автоматизировать, чтобы освободить свое время для других дел.

В этом модуле речь и пойдет про автоматизацию административных операций. Также будут рассмотрены средства, которые обеспечивают дополнительные функциональные возможности для автоматизации.

Отметим также, что часто для автоматизации административных операций удобно использовать скрипты VBScript или JavaScript (или программы на любом COM-

совместимом языке, например, Visual Basic, VBA, C++, Java, Delphi и другие, или .NET-совместимые программы), которые работают с объектными моделями SMO, SQL-DMO и WMI.

Автоматизация административных операций средствами SQL Server Agent

Что такое SQL Server Agent

SQL Server Agent — это служба SQL Server, основное назначение которой — автоматизация выполнения административных операций. Сама автоматизация осуществляется при помощи:

q - заданий (jobs) — именованных наборов действий, которые можно выполнять по расписанию;

q - предупреждений (alerts) — действий, которые выполняются в ответ на событие, произошедшее на SQL Server. Таким действием может быть, например, выполнение задания или отправка сообщения оператору. События — это либо ошибки с определенным номером на SQL Server (их можно определять и генерировать самостоятельно), либо выход счетчика производительности за какие-то границы, либо специальные события WMI (т. е. ответ, пришедший на специальный событийный запрос на языке WQL);

q - операторов (operators) — записей в адресной книге, на которые будут отправляться сообщения.

Задания, предупреждения и операторы будут подробно рассмотрены в следующих разделах. Пока же отметим только общие моменты, которые связаны с SQL Server Agent.

Первое, что необходимо отметить — для использования автоматизации административных операций необходимо, чтобы служба SQL Server Agent работала. Будет ли она запускаться автоматически при запуске сервера или ее нужно будет запускать вручную, зависит от параметров, которые вы выбрали при установке сервера. Проверить, работает ли эта служба (и при необходимости запустить или изменить режим запуска), можно при помощи SQL Server Configuration Manager.

Второй момент — служебная информация SQL Server Agent (в том числе информация о заданиях, предупреждениях и операторах) хранится в специальной служебной базе данных msdb. Если вы активно работаете с заданиями и предупреждениями, не забывайте регулярно производить резервное копирование этой базы данных.

Третье, что нужно отменить, — возможности SQL Server Agent (и его работоспособность) зависят от того, от имени какой учетной записи работает эта служба. Рекомендуется, чтобы:

q - SQL Server Agent работал от имени той же доменной учетной записи, от имени которой работает сам SQL Server;

q - эта доменная учетная запись обладала бы на компьютере правами локального администратора.

Учетная запись, от имени которой работает SQL Server Agent, определяется при установке SQL Server . Затем ее можно изменить, например, при помощи SQL Server Configuration Manager.

Возможностей настройки безопасности при работе SQL Server Agent в SQL Server очень много.

И, в четвертых, для SQL Server Agent предусмотрена своя система журналов, при помощи которой можно получить информацию о всех происходящих с этой службой событиях. Просмотреть журналы можно из контейнера SQL Server Agent | Error Logs (Журналы ошибок) в SQL Server Management Studio.

Перед созданием заданий, оповещений и операторов рекомендуется проверить параметры SQL Server Agent: соответствуют ли они вашим потребностям.

Контрольные вопросы:

1. Служба SQL Server Agent: назначение, автоматический запуск от имени доменной учетной записи, роль базы данных MSDB
2. Автоматизация административных операций средствами SQL Server Agent
3. Что такое SQL Server Agent?

Лекция 11.
Импорт и экспорт данных

Цель: рассмотреть импорт и экспорт данных

План занятия:

1. Средства для массового импорта и экспорта данных
2. Повышение производительности передачи данных
3. Минимизация ведение журнала транзакций
4. Отключение и перестроение индексов
5. Отключение и включение ограничений

Средства для массового импорта и экспорта данных

Большое количество данных, находящихся в системе Microsoft SQL Server вводится непосредственно пользователями в приложениях, однако часто возникает необходимость импорта/экспорта данных. SQL Server предоставляет для этого набор инструментов: оператор BULK INSERT и функции OPENROWSET реализуются в ядре базы данных, утилиты BCP и служба SSIS являются внешними по отношению к ядру БД.
При выполнении массовых операций нередко возникает ряд проблем.
Например, когда большие объемы данных должны быть вставлены в таблицы SQL Server, необходимо обеспечить наилучшую производительность. Для этого необходимо уметь правильно настраивать параметры для ограничений, триггеров и индексов.

Не все данные могут быть введены пользователями базы данных построчно. Часто данные должны быть импортированы из внешних источников данных, таких как файлы или другие серверы баз данных. Кроме того, пользователи часто просят, чтобы данные из таблиц БД экспортировались в текстовые файлы. Может вызвать проблемы неправильно настроенные параметры сортировки. Корректировка параметров сортировки базы данных часто требует экспорта и повторного импорта данных из базы данных.

Обзор вопросов передачи данных

Передача данных. Хотя не все требования передачи данных идентичны, существует стандартный алгоритм, которого придерживается большинство задач передачи данных:
- извлечение данных из источника данных;
- преобразование данных в формат целевой системы;
- загрузка данных в целевую систему.

Вместе эти три шага обычно называют процессом ETL (Extract, Transform, Load), который может быть реализован с использованием ETL-инструментов.

В некоторых ситуациях более подходящим может быть процесс ELT (Extract, Load, Transform) - выполнять преобразования данных уже после их загрузки в базу данных.

Извлечение данных. Извлечение данных, как правило, включает в себя выполнение запросов на источнике, или открытие и чтение исходных файлов, хотя есть и другие варианты.

В процессе извлечения данных преследуют две следующие общие цели.

- Избежать чрезмерного воздействия на систему-источник. Например, не читают целые таблицы данных, когда нужно прочитать только выбранные строки или столбцы. Кроме того, не перечитывают одни и те же данные и в любом случае избегают выполнения операторов, которые блокируют пользователей системы-источника.
- Обеспечить согласованность извлечения данных. Например, не включают в выходные данные одну строку из системы-источника больше, чем один раз.

Преобразование данных. Этап преобразования процесса ELT обычно включает в себя несколько следующих шагов.

- Данные должны быть очищены. Например, может потребоваться удалить ошибочные данные или предоставить значения по умолчанию для пропущенных (отсутствующих) столбцов.
- Возможно, придется выполнять поиск. Например, входные данные могут включать имя клиента, но базе данных в таком случае может понадобиться идентификатор клиента.
- Данные должны быть агрегированы. Например, входные данные могут включать в себя все транзакции, которые произошли за день, но базе данных могут понадобиться только ежедневные итоговые значения.
- Данные, возможно, потребуется разгруппировать. Это часто называют распределение данных. Например, входные данные могут включать в себя квартальные бюджеты, но база данных может потребовать ежедневный бюджет.

Помимо этих общих операций, данные, возможно, должны быть в некотором роде реорганизованы, например, сведение данных таким образом, чтобы столбцы становились строками, объединяющими нескольких столбцов источника в одну колонку, или разбиение одного исходного столбца на несколько столбцов.

Загрузка данных. После того, как данные преобразованы в соответствующий формат, их можно загрузить в целевую систему. Вместо выполнения операции вставки данных построчно, можно использовать специальные опции для массовой загрузки данных. Кроме того, можно изменить временную конфигурацию для повышения производительности операции загрузки.

Средства для массового импорта и экспорта данных

SQL Server поддерживает массовый экспорт данных из таблиц SQL Server и массовый импорт данных в таблицы или несекционированные представления.

Для выполнения этих задач SQL Server предоставляет **набор инструментов**.

Важно понять, какой метод лучше всего использовать и для каких типов сценария. Обзор доступных средств приведен в табл. 1.

Таблица 1

Основные средства для выполнения операций импорта/экспорта данных

Метод	Описание	Импорт/экспорт данных
Экспорт данных программа bcp	Программа командной строки (bcp.exe), массово экспортирующая и импортирующая данные и создающая файлы форматирования.	Да/Да
Инструкция BULK INSERT	Инструкция T-SQL, импортирующая данные непосредственно из файла данных в таблицу базы данных или несекционированное представление.	Да/Нет
Инструкция INSERT … SELECT *FROM OPEN-	Инструкция T-SQL INSERT, использующая поставщик больших наборов строк OPENROWSET для массового импорта данных в таблицу.	Да/Нет

ROWSET(BULK…)	OPENROWSET – это табличная функция, для подключения и извлечения данных из источников OLE DB. Полная информация о том, как подключиться к источнику данных, указывается в параметрах функции. SQL Server имеет OLE DB, именуемый BULK, который можно использовать	
с помощью функции OPENROWSET. Можно использовать OPENROWSET для подключения к другим СУБД. мастер импорта и экспорта	Создает простые пакеты, которые импортируют и экспортируют данные в многочисленных рапространенных форматах, включая базы данных, электронные таблицы и текстовые файлы.	Да/Да

Подробное описание использования этих средств и методов можно найти в электронной документации Microsoft.

Повышение производительности передачи данных

При выполнении операций массового экспорта/импорта для того, чтобы обеспечить наилучшую производительность, рекомендуется:
- минимизировать блокировки;
- минимизировать ведение журнала транзакций;
- отключить ограничения, индексы и триггеры.

Для чего надо минимизировать блокировки. Блокировка используется в компоненте SQL Server Database Engine для синхронизации одновременного доступа нескольких пользователей к одному и тому же фрагменту данных. При изменении фрагмента данных транзакция удерживает блокировку, защищая изменения до конца транзакции. По умолчанию SQL Server управляет гранулярностью блокировок, начиная с уровня блокировки строк и пытаясь укрупнить блокировку, когда значительное число отдельных строк заблокированы в таблице.

Управление большим количеством блокировок занимает ресурсы, которые могли бы использоваться для минимизации времени выполнения запросов. При выполнении операций массового импорта, как правило, нецелесообразно ставить блокировки на уровне строк, а заблокировать всю таблицу сразу.

Программа bcp и инструкция BULK INSERT и INSERT ... SELECT * FROM OPENROWSET(BULK...) позволяют указать, что на время выполнения операции массового импорта таблица будет блокирована (целиком). Если указана блокировка таблицы для операции массового импорта, то к таблице применяется блокировка массового обновления (BU) на время выполнения операции массового импорта. Блокировка BU позволяет одновременно выполнять массовый импорт данных в одну и ту же таблицу нескольким потокам и вместе с тем предотвращает доступ к таблице других процессов, не осуществляющих массовый импорт данных. Блокировка таблицы может повысить производительность массового импорта, снизив конкуренцию за эту таблицу.

В табл. 2 перечислены квалификаторы, определяющие блокировку таблиц в командах массовой загрузки.

Таблица 2

Квалификаторы, задающие блокировку таблиц	Команда	Квалификатор
Тип квалификатора программа bcp	-h " TABLOCK "	Подсказка
Инструкция BULK INSERT	TABLOCK	Аргумент
Инструкция INSERT … SELECT *	WITH(TABLOCK)	Табличная подсказка

FROM OPENROWSET(BULK…)		

Минимизация ведение журнала транзакций

В простой модели восстановления массовые операции производятся с минимальным протоколированием. В БД, использующей модель полного восстановления, все операции вставки строк полностью записываются в журнал транзакций. Во время импорта большого количества данных это может привести к быстрому заполнению журнала транзакций. Чтобы выполнять операции массового импорта с минимальным протоколированием в базе данных, которая обычно использует модель полного восстановления, ее рекомендуется сначала переключить на модель восстановления с неполным протоколированием, а после выполнения массового импорта данных снова переключить на модель полного восстановления.

Не все команды могут использовать минимальное протоколирование. Способствовать минимизации ведения журнала помогут следующие рекомендации.

- Таблица не реплицируется.
- Указана блокировка таблицы (с помощью TABLOCK).
- Если у таблицы нет кластеризованного индекса, но имеется один или несколько некластеризованных индексов, страницы данных всегда протоколируются минимально. Однако, как идет запись страниц индекса в журнал, зависит от того, пуста ли таблица.
- Если таблица пуста, страницы индекса протоколируются минимально.
- Если таблица не пуста, страницы индекса протоколируются полностью.
- Если таблица имеет кластеризованный индекс и пуста, ведется минимальная запись страниц данных и индекса в журнал.
- Если таблица имеет кластеризованный индекс и не пуста, страницы данных и индекса протоколируются полностью, независимо от модели восстановления.

Это не исчерпывающий перечень ограничений, которые должны быть выполнены для того, чтобы минимизировать запись в журнал транзакций.

Отключение и перестроение индексов

Чтобы в процессе импорта или обновления не проверять каждое значение каждого индекса для каждой строки, можно повысить общую производительность путем отключения процесса обслуживания индексов до тех пор, пока все данные не будут загружены. Для этого можно индекс не удалять из базы данных, а только отключить его, т.е. метаданные об индексе остаются, просто останавливается его обновление. Запросы не будут использовать отключённые индексы.

Отключение индекса обеспечивает:

- предотвращает доступ пользователей к индексу;
- предотвращает доступ к данным, если индекс кластеризованный;
- сохраняет определение индекса в метаданных;
- ускоряет импорт данных в таблицах.

Индекс можно отключить с помощью графического интерфейса в SSMS или с помощью инструкции ALTER INDEX. В следующем примере кода отклю- чается индекс с именем idx_emailaddress в таблице БД dbo.Customer.

```
ALTER INDEX idx_emailaddress ON dbo.Customer
DISABLE;
```

Можно отключить все индексы сразу в таблице БД dbo.Customer, как по- казано в следующем примере.

```
ALTER INDEX ALL ON dbo.Customer
DISABLE;
```

Кластеризованный индекс определяет структуру таблицы. Если отключен кластеризованный индекс, таблица становится недоступной до тех пор, пока индекс не будет перестроен.

После завершения импорта данных можно перестроить (фактически создать заново) индексы для таблицы с помощью графических средств в среде

SSMS, с помощью инструкции ALTER INDEX или команды DBCC DBREINDEX.

В следующем примере кода показано, как перестроить индекс с именем

idx_emailaddress в таблице БД dbo.Customer.

ALTER INDEX idx_emailaddress ON dbo.Customer

REBUILD;

Вы также можете использовать ключевое слово ALL с инструкции ALTER

INDEX для перестроения всех индексов в указанной таблице (аналогично примеру отключения индекса).

Если загружен большой объем данных более эффективным может быть удаление существующих индексов и повторное создание индексов. Для повторного создания индекса, который заменит существующий, можно использовать инструкцию CREATE INDEX с параметром DROP_EXISTING, как показано в следующем примере.

```
CREATE INDEX idx_emailaddress ON dbo.Customer(EmailAddress)
WITH (DROP_EXISTING = ON);
```

Отключение и включение ограничений

Ограничения используются, чтобы обеспечить соблюдение правил целостности данных.

Ограничения PRIMARY KEY и UNIQUE. SQL Server создает индексы для обеспечения этих ограничений. Чтобы отключить первичный ключ или ограничение уникальности, необходимо отключить индекс, связанный с ограничением.

Это, как правило, используется только для некластеризованного первичного ключа. При повторном включении ограничения, соответствующие индексы автоматически восстанавливаются. Если во время перестройки индекса будут найдены повторяющиеся значения, повторное включение ограничения завершится ошибкой. По этой причине, если вы отключаете эти ограничения при импорте данных, то должны быть уверены, что импортируемые данные не будут нарушать соблюдение этих ограничений.

Ограничения FOREIGN KEY и CHECK. Ограничения внешнего ключа используются, чтобы убедиться, что сущности в одной таблице, на которые ссылаются сущности из другой, на самом деле существуют. Например, поставщик должен существовать до того, как может быть введен заказ на поставку. Ограничения внешнего ключа при проверке ссылок используют первичный ключ или ограничения уникальности. Поэтому, если отключить первичный ключ или ограничение уникальности, на которое оно указывает, то ограничение внешнего ключа автоматически отключается. Тем не менее, при повторном включении первичного ключа или ограничения уникальности, ограничения внешнего ключа, которые на них ссылаются, не будут включены автоматически.

Ограничения check можно использовать для ограничения значений, которые могут содержаться в столбце или взаимосвязи между значениями в нескольких столбцах таблицы. Можно отключить и включить ограничения FOREIGN

KEY и CHECK с помощью параметры CHECK и NOCHECK. Пример кода для отключения и включения ограничения с именем SalaryCap:

ALTER TABLE Person.Salary NOCHECK CONSTRAINT SalaryCap;

ALTER TABLE Person.Salary CHECK SalaryCap;

Также можно отключить или включить все ограничения, заменив в инструкции ALTER TABLE имя ограничения на ключевое слово ALL.

Копирование и перемещение баз данных

В некоторых случаях необходимо скопировать или переместить всю базу данных с одного экземпляра SQL Server на другой. Это можно выполнить одним

из следующих способов:

- с помощью мастера копирования баз данных (Copy Database Wizard);
- при помощи отсоединения (detach) и присоединения (attach);
- путем создания (backup) и восстановления (restore) резервных копий;
- приложений уровня данных (Data-tier applications).

С помощью мастера копирования баз данных можно легко перемещать или копировать базы данных и их объекты с одного сервера на другой, без перерывов в работе сервера. Можно также обновить базы данных с прошлой версии

SQL Server до версии SQL Server 2017. С помощью этого мастера можно сделать следующее.

- Выбрать исходный и целевой серверы.
- Выбрать базы данных для перемещения, копирования или обновления.
- Указать расположение файлов для баз данных.
- Создать имена входа для целевого сервера.
- Копировать дополнительные вспомогательные объекты, задания, пользовательские хранимые процедуры и сообщения об ошибках.
- Задать расписание перемещения или копирования баз данных.

Мастер предоставляет два метода копирования или перемещения базы данных. Он может быть настроен на использование метода отсоединения и присоединения: это самый быстрый вариант, но имеет недостаток, так как исходная база данных должна находиться в автономном режиме.

Второй – метод SMO, использует библиотеки объектов управления SQL

Server (Server Management Objects) для создания объектов и передачи данных.

Этот метод выполняет чтение определения каждого объекта базы данных-источника и создает каждый из этих объектов в целевой базе данных. После этого происходит перенос данных из исходных таблиц в целевые таблицы с воссозданием индексов и метаданных. Этот вариант медленнее, но исходная база данных во время копирования может находиться в режиме online.

Если в мастере выбран параметр «Переместить», то после перемещения

базы данных мастер автоматически удаляет базу данных-источник. Если выбран параметр «Копировать», исходная база данных остается без изменений.

Существует ряд ограничений:

- мастер копирования баз данных недоступен в выпуске Express;
- мастер копирования базы данных нельзя использовать для перемещения или копирования системных баз данных;
- после обновления БД возврат к предыдущей версии невозможен;
- на сервере назначения должен быть запущен агент SQL Server.

Если исходная база данных используется, когда мастер пытается переместить или скопировать базу данных, операция не выполняется.

Запуск мастера копирования баз данных требует привилегий системного администратора (sysadmin) на обоих экземплярах (как на исходном, так и на целевом сервере) и наличия сетевого подключения.

Использование операций Detach и Attach. Отсоединение базы данных в SQL

Server можно выполнить с помощью среды SSMS или T-SQL. Отсоединенные файлы останутся на диске и могут быть повторно присоединены с помощью среды SSMS или с помощью инструкции CREATE DATABASE с параметрами FOR

ATTACH или FOR ATTACH_REBUILD_LOG. Файлы можно также переместить на другой сервер и подсоединить там.

Не рекомендуется подключать или восстанавливать базы данных, полученные из неизвестных или ненадежных источников. В этих базах данных может содержаться вредоносный код, вызывающий выполнение непредусмотренных инструкций T-SQL или появление ошибок из-за изменения схемы или физической структуры базы данных. Перед тем как использовать базу данных, полученную из неизвестного или ненадежного

источника, выполните на тестовом сервере инструкцию DBCC CHECKDB для этой базы данных, а также изучите исходный код в базе данных, например, хранимые процедуры и другой пользовательский код.

Для отсоединения БД используется системная хранимая процедура sp_detach_db, которой надо передать как минимум один обязательный параметр – имя отсоединяемой базы данных. Следующий код выполняет отсоединение БД Works20.

```
EXEC sp_detach_db 'Works20', 'true';
```

В этом примере указан второй – необязательный – параметр skipchecks в значении TRUE, чтобы не выполнять инструкцию UPDATE STATISTICS. Чтобы явно запустить инструкцию UPDATE STATISTICS, надо указать значение FALSE.

При выполнении операции отсоединения, необходимо учитывать следующие обстоятельства.

- Невозможно отсоединить базу данных, если она в настоящий момент используется. Для отсоединения базы данных требуется монопольный доступ, для чего надо переключить ее в режим SINGLE_USER.

Например, следующая инструкция ALTER DATABASE получает монопольный доступ к базе данных Works20 после отключения от этой базы данных всех текущих пользователей:

```
USE master;
ALTER DATABASE Works20
SET SINGLE_USER;
GO
```

- База данных помечена как подозрительная. Подозрительную базу данных перед ее отсоединением необходимо перевести в аварийный режим.
- База данных является системной базой данных.
- Перед отсоединением базы данных необходимо удалить все моментальные снимки, если таковые имеются.
- Для выполнения операций требуется членство в предопределенной роли сервера sysadmin.
- При отсоединении базы данных все метаданные удаляются. Если эта база данных была базой данных по умолчанию для учетной записи входа, базой данных по умолчанию становится master.

Копирования базы данных с помощью Backup и Restore. В SQL Server можно создать новую базу данных, восстановив резервную копию пользовательской базы данных, созданной в SQL Server 2005 или более поздней версии. Однако резервные копии баз данных master, model и msdb, созданных в более ранней версии SQL Server, восстановить на SQL Server 2017 невозможно. Кроме того, резервные копии, созданные в SQL Server 2017, невозможно восстановить в более ранних версиях SQL Server.

При использовании резервного копирования и восстановления для копирования базы данных на другой экземпляр SQL Server компьютер-источник и целевой компьютер могут быть любой платформой, на которой запускается SQL Server.

Основные этапы данной технологии следующие.

1. Создайте резервную копию базы данных источника, которая может находиться на экземпляре SQL Server 2005 или более поздней версии. Компьютер, на котором запущен этот экземпляр SQL Server, называется компьютером- источником.

2. На компьютере, куда нужно скопировать базу данных (целевой компь- ютер), подключите экземпляр SQL Server, на котором будет восстановлена база данных. При необходимости создайте те же устройства резервного копирования на целевом экземпляре сервера, что использовались для резервного копирования баз данных-источников.

3. Восстановите резервную копию базы данных источника на целевом компьютере. При восстановлении базы данных автоматически создаются все ее файлы.

Существуют ряд других вопросов, влияющих на процесс копирования БД.

При восстановлении базы данных необходимые файлы базы данных создаются автоматически. По умолчанию у файлов, созданных SQL Server в процессе восстановления, те же имена и пути, что и у файлов резервной копии исходной базы данных на компьютере-источнике. В некоторых ситуациях при восстановлении базы данных необходимо указать сопоставление дисков, имена файлов

или путь для восстановления. Например, на целевом диске может быть недостаточно свободного места, или используется имя базы данных, которое уже существует на целевом сервере восстановления, а имя каждого из ее файлов совпадает с именем файла базы данных в резервном наборе данных.

Во избежание ошибок и непредвиденных последствий перед операцией восстановления можно использовать таблицы журнала backupfile, чтобы найти в резервной копии файлы базы данных и журнала, которые планируется восстановить.

При восстановлении базы данных на другом компьютере имя входа пользователя SQL Server или Microsoft Windows, начавшего процесс восстановления, автоматически становится именем владельца базы данных. При восстановлении базы данных системный администратор или владелец новой базы данных могут сменить ее владельца. Для предотвращения несанкционированного восстановления базы данных устанавливайте пароли на носители или сами резервные копии.

Чтобы обеспечить целостность работы пользователей и приложений при восстановлении базы данных на другой экземпляр сервера, на новом экземпляре необходимо повторно создать некоторые или все метаданные, например имена входа и задания.

Создание новой учетной записи может быть произведено с помощью системной хранимой процедуры sp_addlogin. sp_addlogin

[@login=] 'учетная_запись'

[, [@password=] 'пароль']

[, [@defdb=] 'база_данных_по_умолчанию']

После завершения аутентификации и получения идентификатора учетной записи (login ID) пользователь считается зарегистрированным, и ему предоставляется доступ к серверу. Для каждой базы данных, к объектам которой он намерен получить доступ, учетная запись пользователя(login) ассоциируется с поль- зователем (user) конкретной базы данных, что осуществляется посредством про- цедуры sp_adduse. sp_adduser

[@loginame=] 'учетная_запись'

[, [@name_in_db=] 'имя_пользователя']

[, [@grpname=] 'имя_роли']

Data Export for SQL Server – это мощный инструмент, предназначенный для быстрого экспорта данных из баз данных SQL Server в любой из 20 доступных форматов, включая MS Access, MS Excel, MS Word (RTF), HTML, XML, PDF,

TXT, CSV, DBF, ODF, ACCDB и другие. Data Export for SQL Server располагает удобным мастером настройки для визуальной установки параметров экспорта для каждой таблицы (конечные имена файлов, экспортируемые поля, форматы данных и многое другое) и консольной утилитой для быстрого экспорта данных из таблиц и запросов (см. файл в папке КП ТЕМА 19 - Data Export for SQL
Server).

Контрольные вопросы:

1. Перечислите средства для массового импорта и экспорта данных
2. В чем заключается повышение производительности передачи данных

3. Опишите минимизацию ведение журнала транзакций
4. Опишите процесс отключения и перестроения индексов
5. Опишите отключения и включения ограничений

Лекция 12.
Автоматизация управления SQL

Цель: рассмотреть автоматизацию управления SQL

План занятия:

1. Компоненты агента SQL Server
2. Агент SQL Server как система планирования заданий

Компоненты агента SQL Server

Агент SQL Server представляет собой систему планирования, обработчик заданий, средство общения и круглосуточного администрирования для Microsoft SQL Server.

Компоненты агента SQL Server

Агент SQL Server выглядит как единая служба, но состоит из нескольких компонентов:

- планирование заданий;
- уведомления и предупреждения;
- операторы;
- журнал ошибок;
- серверы-посредники.

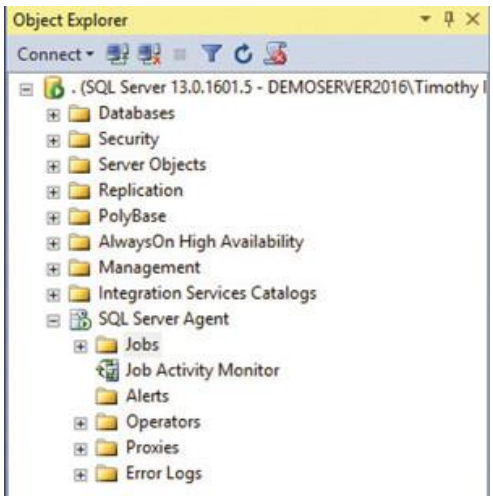
Тема агента SQL Server весьма обширна, и потребуется несколько статей, чтобы рассмотреть каждую грань его функциональности. В первой статье мы сосредоточимся на компоненте планирования заданий.

Агент SQL Server как система планирования заданий

Когда речь идет о необходимости круглосуточного присутствия, я имею в виду функциональность агента SQL Server как планировщика заданий. Агент SQL Server позволяет создавать задания различной сложности, состоящие из одного или нескольких «шагов», на каждом из которых можно выполнять команды на различных языках, вплоть до Transact-SQL, и наборы ориентированных на SQL задач (запросы или команды SQL Server Analysis Services, пакеты SQL Server Integration Services), а также сценарии PowerShell, VB Script и задачи операционной системы. Все шаги могут быть объединены в цепочку для последовательного выполнения или запускаться в зависимости от успеха, неудачи или завершения каждого шага в цепочке. Кроме того, вы можете отправлять предупреждения, сохранять выходные данные шага и создавать файлы журнала на каждом шаге. Выполнение задач можно запланировать согласно расписанию. Как шаги, так и собственно задание могут быть настроены для рассылки предупреждений и уведомлений. Можно даже настраивать задания для выполнения на удаленных целевых объектах.

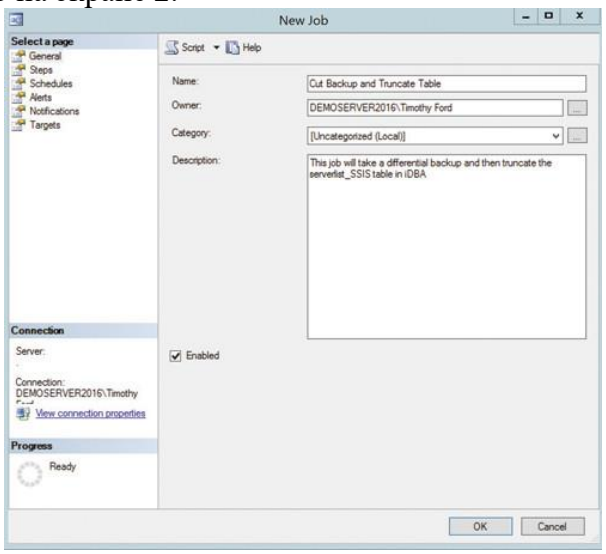
Рассмотрим для примера процесс создания простого многошагового задания, которое усекает таблицу в базе данных после создания разностной резервной копии, чтобы убедиться в возможности восстановления в случае внесения ошибочных изменений.

Функции для создания задания можно найти с помощью обозревателя объектов в среде Microsoft SQL Server Management Studio (SSMS). Разверните узел сервера, и вы увидите агент SQL Server как подузел (см. экран 1).



Экран 1. Компоненты агента SQL Server

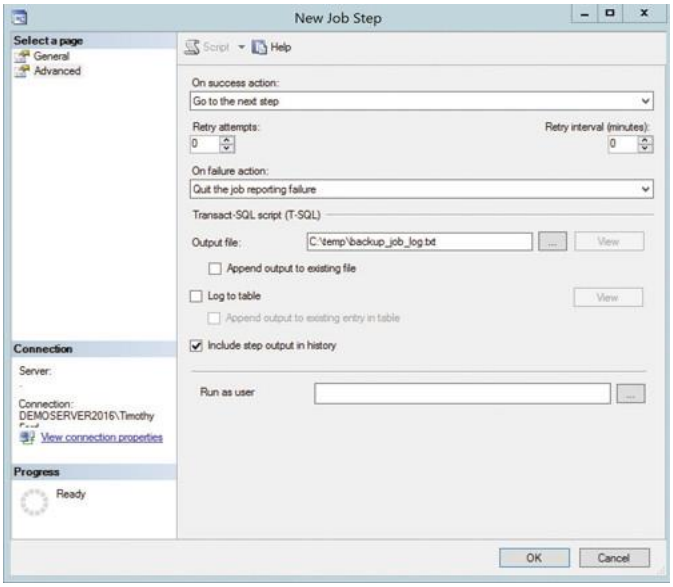
Щелкните правой кнопкой мыши на разделе Jobs («Задания») и выберите в контекстном меню пункт New Job («Создать задание»). Вы увидите форму, представленную на экране 2.



Экран 2. Создание нового задания

На данном этапе я заполнил необходимые поля, чтобы сделать то, что требуется на первом шаге: создать разностную резервную копию. По умолчанию владелец задания — текущий зарегистрированный пользователь. Задание принадлежит моей учетной записи в AD, но это не значит, что у него тот же контекст выполнения. По умолчанию задание выполняется в контексте учетной записи службы агента SQL Server. Это означает, что нужно предоставить учетной записи службы все права, необходимые для выполнения требуемых задач. Кроме того, если в выполнение задачи вовлечены какие-либо локальные или удаленные диски, служба агента SQL Server (или выполняющая учетная запись, так как это определяемый параметр) должна иметь права для доступа к соответствующим томам.

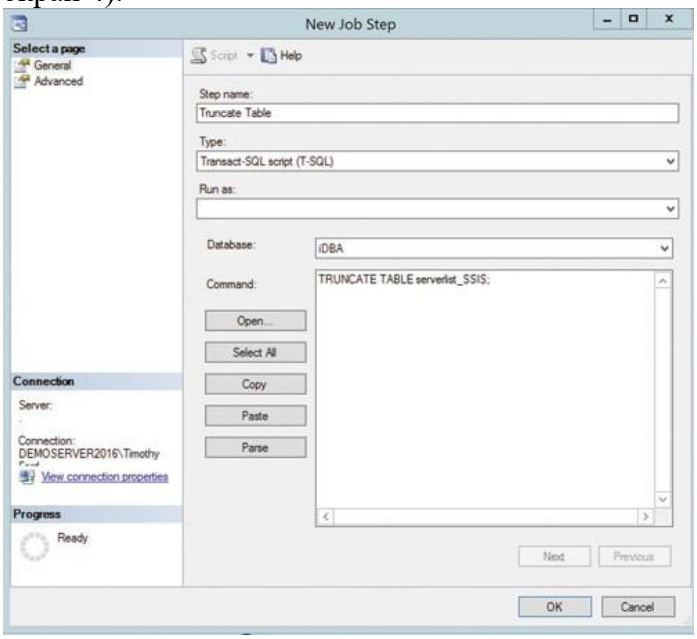
На каждом шаге мы видим два окна: общих настроек и дополнительных параметров. Общие настройки показаны на экране 2, и в этом окне можно объявить, какую команду следует выполнить. В окне дополнительных параметров представлены критерии, согласно которым происходят переходы от одного шага к другому, а также регистрируемая информация о шаге (см. экран 3).



Экран 3. Окно дополнительных параметров

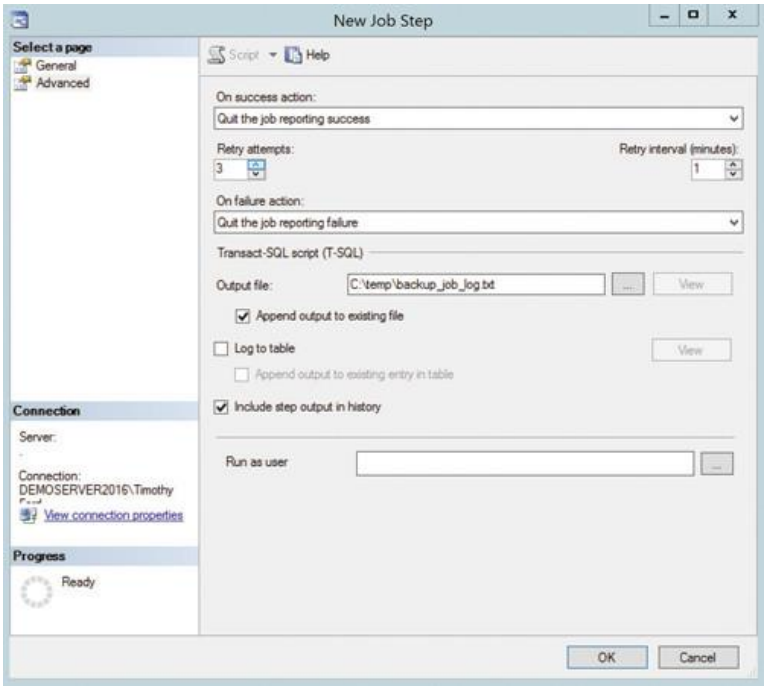
Необходимо сделать так, чтобы шаг усечения (следующий в цепочке) выполнялся только в случае успеха резервного копирования. Если резервное копирование прошло неудачно, то задание должно завершиться ошибкой. Я не пытаюсь повторить этот шаг, хотя такая возможность существует. Кроме того, я хочу записать выходные данные шага (вы увидите их на вкладке сообщений окна запроса в SSMS, если выполните программный код шага) в файл, хранящийся в C:\temp. Этот файл, если он существует, будет перезаписан, и я хочу сохранить выходные данные шага в журнале. Обычно я записываю выходные данные двумя способами, так как в самом файле выходные данные не обрезаются, а для журнала шага действуют ограничения по размеру строки. Вы также видите, что можно настроить задачу на выполнение другим пользователем, а не только учетной записью службы по умолчанию для агента SQL Server.

Теперь мы повторяем процесс для шага усечения с подходящими значениями (см. экран 4).



Экран 4. Создание шага усечения базы данных

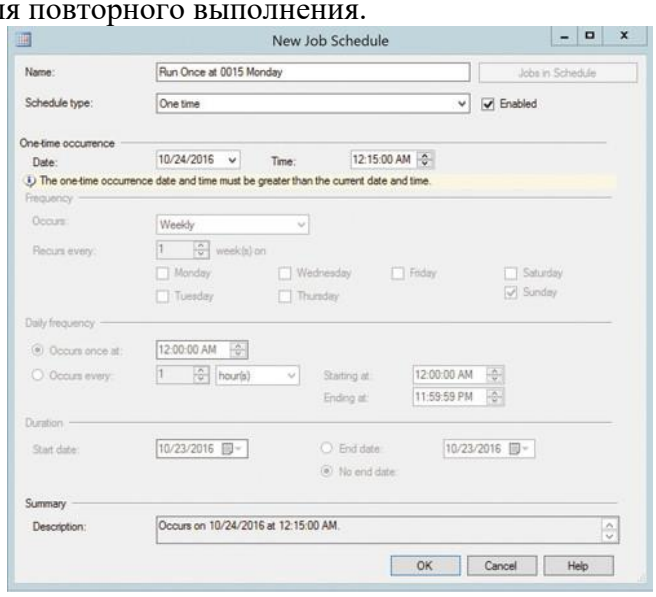
На шаге усечения назначается контекст базы данных, указывающий на базу данных, в которой планируется выполнить усечение таблицы. Я мог бы оставить ее как основную базу данных по умолчанию, при условии что команда содержит имя базы данных и схему (см. экран 5).



Экран 5. Настройка шага усечения базы данных

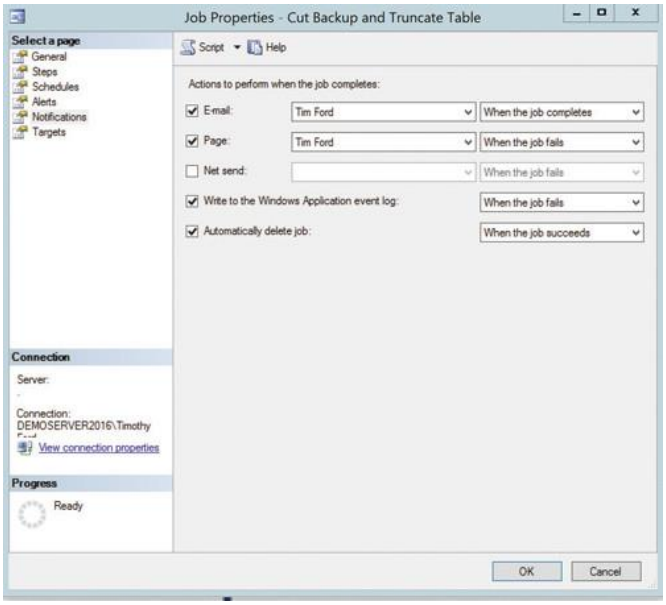
Для шага усечения я готов повторить попытку до трех раз с ожиданием в течение одной минуты между попытками. Если шаг завершается успешно, то нужно, чтобы и задание завершилось как успешное. Аналогично, если шаг завершается неудачей, то неудачно и задание. Это учитывается при настройке предупреждений и уведомлений о заданиях. И вновь я хочу записать выходные данные в файл, созданный или указанный на первом шаге, но на этот раз присоединить выходные данные к существующему заданию, чтобы увидеть упорядоченные выходные данные всех шагов в одном файле.

При планировании задания пользователям предоставляются широкие возможности. Вы можете выполнить задание один раз, как показано на экране 6, или составить расписание для повторного выполнения.



Экран 6. Планирование выполнения задания

Наконец, мы дошли до окна настройки параметров уведомления (см. экран 7). В нашем случае я хочу, чтобы мне (Тиму Форду) было послано уведомление о сбое задания через страницу, поскольку это важно. Если задание завершается (успешно или неудачно, неважно), я хочу получить по электронной почте сообщение о состоянии. Я также хочу сохранить в журнале Windows запись о состоянии задания, и, если оно выполнено успешно, удалить задание, так как оно предназначено лишь для разового выполнения.



Экран 7. Окно настройки параметров уведомления

Как уже подчеркивалось в начале статьи, агент SQL Server — инструмент с широкими возможностями.

Контрольные вопросы:

1. Перечислите компоненты агента SQL Server
2. Опишите работу агента SQL Server как систему планирования заданий

Лекция 13.
Выполнение мониторинга SQL Server с использование оповещений и предупреждений

Цель: рассмотреть выполнение мониторинга SQL Server с использование оповещений и предупреждений

План занятия:

1. Механизм оповещений
2. Настройка профиля компоненты database mail
3. Добавление оператора оповещений
4. Настройка почты агента sql server
5. Включение триггеров и задач оповещения
6. Проверка работоспособности оповещений

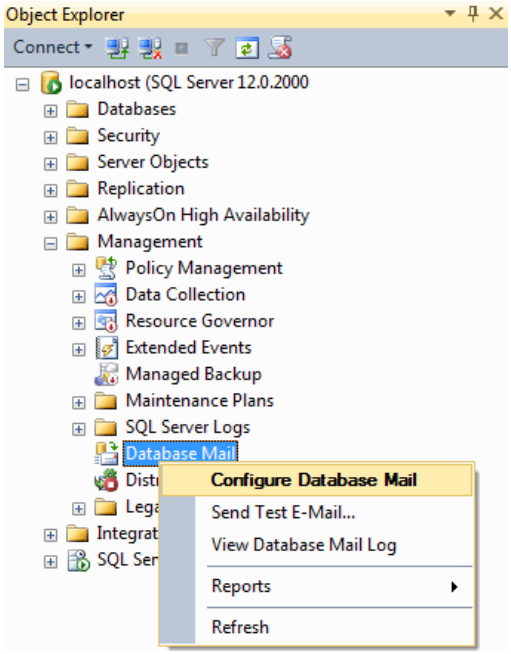
Механизм оповещений

Настройку оповещений можно разделить на 4 этапа:

1. Настройка профиля компонента Database Mail
2. Создание оператора оповещений и настройка почты агента SQL Server
3. Включение триггеров и задач оповещений
4. Проверка работоспособности

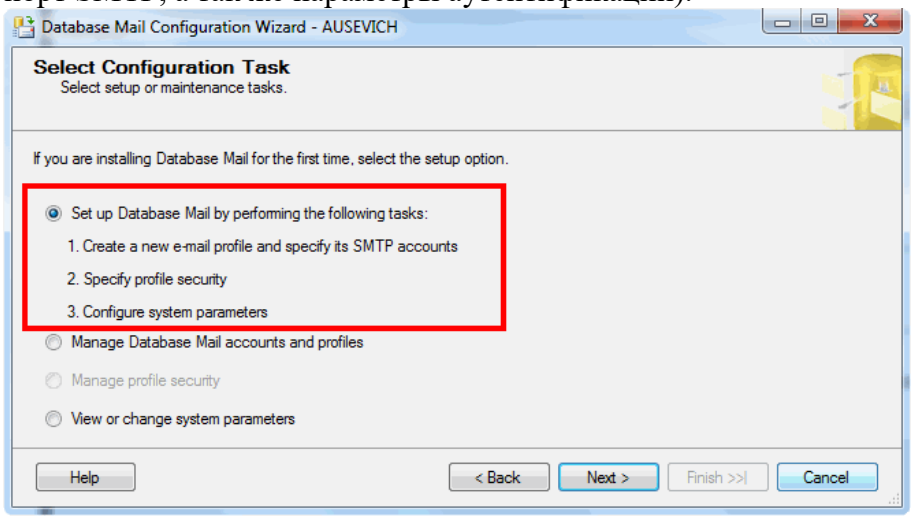
Настройка профиля компоненты Database Mail

Для того чтобы настроить учетную запись компонента Database Mail необходимо в дереве **«Обозревателя Объектов»** (Object Explorer) выбрать настраиваемый экземпляр SQL Server, перейти в «**Управление**» (Management), далее «**Компонент Database Mail**» (Database Mail). Щелкнув правой клавишей мыши на данном пункте, будет открыто контекстное меню, в котором необходимо выбрать «**Настроить…**» (Configure…), после чего будет открыто окно мастера настройки.

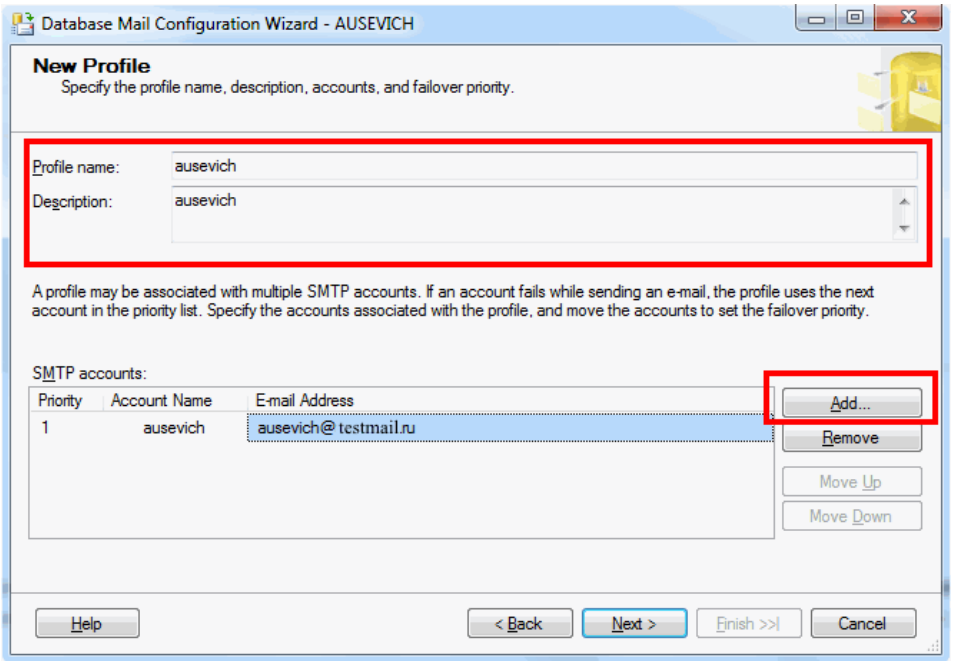


Компонент DatabaseMail в обозревателе объектов

Первую страницу можно пропустить, поэтому перейдем сразу ко второй. Здесь нам необходимо выбрать первый пункт **«Установить компонент Database Mail…»** (Setup Database Mail…) и нажать **«Далее»** (Next). На третьей странице задаем имя и описание профиля, затем нажимаем кнопку **«Добавить»** для добавления учетной записи SMTP. В открывшемся окне необходимо заполнить данные учетной записи (e-mail адрес, сервер и порт SMTP, а так же параметры аутентификации).

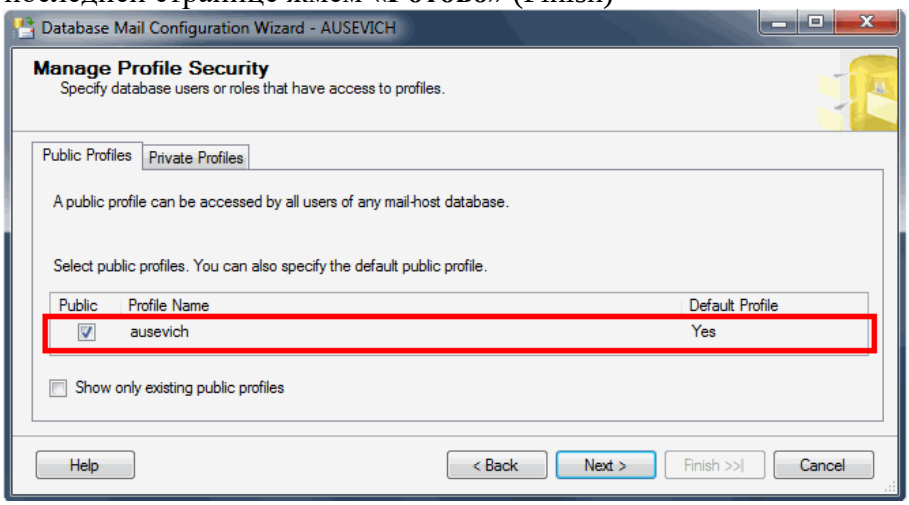


Мастер настройки компонента Database Mail



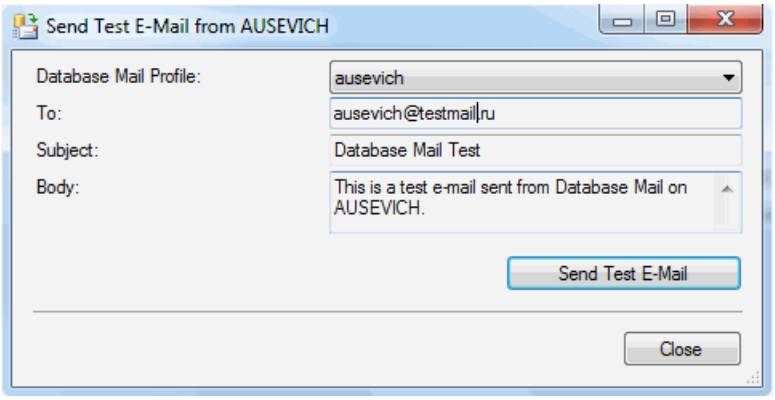
Создание нового профиля компонента Database Mail

На следующей странице необходимо назначить безопасность профиля: указать открытый это профиль или частный (и для каких пользователей), а также можно указать является ли профилем по умолчанию. Для наших целей достаточно сделать профиль открытым и профилем по умолчанию. На пятой странице оставляем все по умолчанию, на последней странице жмем **«Готово»** (Finish)



Настройка безопасности профиля Database Mail

После того как профиль настроен, его надо проверить, для этого в контекстном меню пункта **«Компонент Database Mail»** (вызываемым щелчком правой мыши по данному пункту) надо выбрать **«Отправить тестовое сообщение»** (Send Test E-Mail). В открывшемся окне следует заполнить поле **«Кому»** (To) и нажать **«Отправить …»** (Send …)

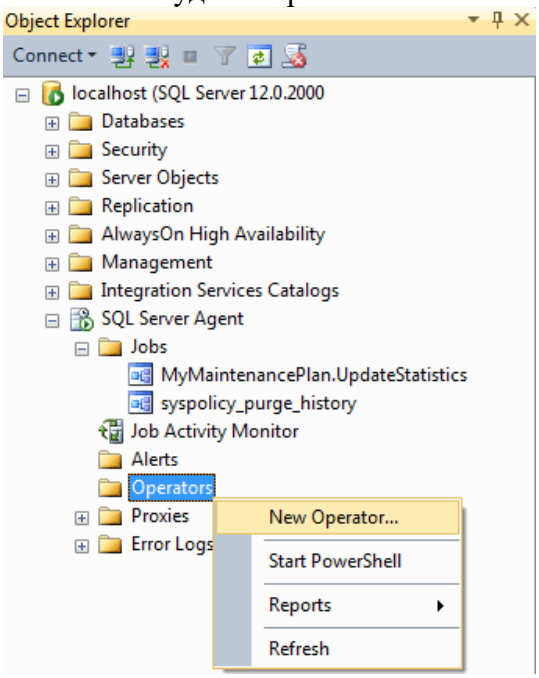


Отправка тестового письма при помощи компоненты Database Mail

Если все сделано правильно, тогда в ближайшее время на почту будет доставлено тестовое письмо.

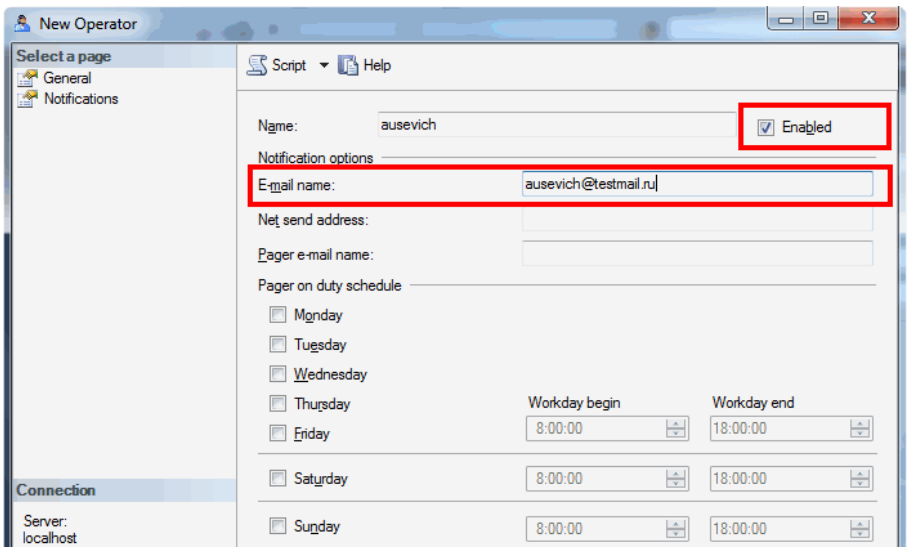
Добавление оператора оповещений

Операторы — это псевдонимы людей или групп, которые могут получать электронные уведомления о завершении задач, заданий или предупреждения. Для добавления нового оператора оповещений необходимо в дереве **«Обозревателя Объектов»** (Object Explorer) выбрать настраиваемый экземпляр SQL Server, перейти в **«Агент SQL Server»** (SQL Server Agent), далее **«Операторы»** (Operators). Щелкнув правой клавишей мыши на данном пункте, выбрать **«Создать оператора»** (New Operator), после чего будет открыто окно свойств оператора оповещений.



Операторы в дереве обозревателя объектов

Настройки оператора находятся на закладке **«Общие»** (General). Здесь необходимо заполнить **«Имя»** (Name), указать состояние **«Включено»** (Enabled), ввести адрес электронной почты. В целом, существуют альтернативные способы оповещения помимо электронной почты: с помощью команды net send или сообщением на пейджер.

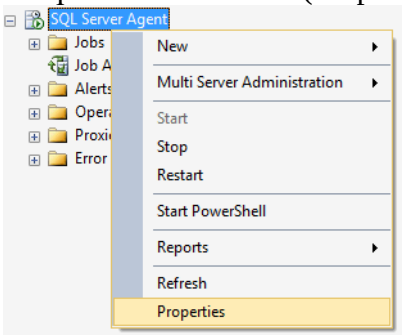


Установка свойств оператора

На этом настройка оператора завершена, перейдем к следующему шагу.

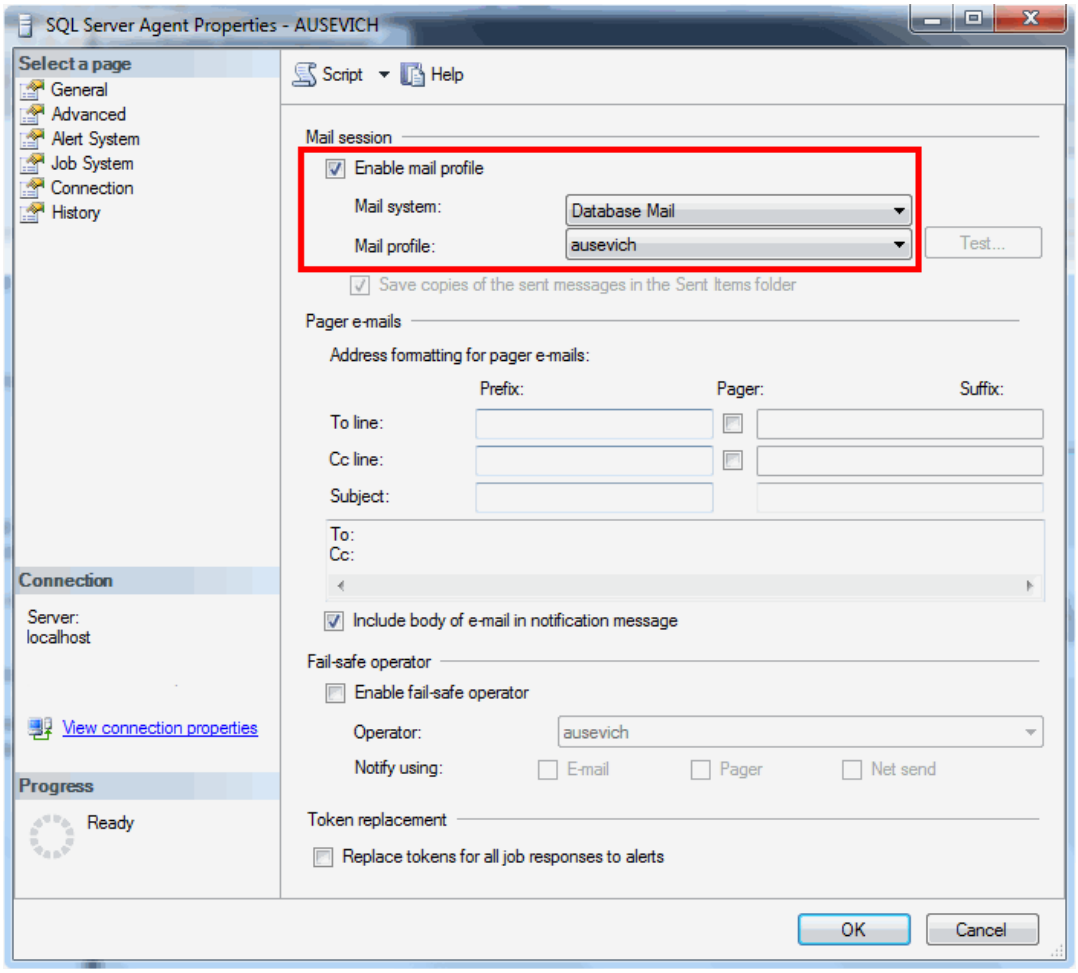
Настройка почты агента SQL Server

Данный пункт необходим для рассылки уведомлений по электронной почты агентом SQL Server. Такая рассылка происходит, например, для уведомления о статусе выполнения задания. Для настройки почты агента необходимо в дереве **«Обозревателя Объектов»** (Object Explorer) выбрать настраиваемый экземпляр SQL Server, перейти в **«Агент SQL Server»** (SQL Server Agent), из контекстного меню выбрать **«Свойства»** (Properties).



Агент SQL Server в дереве обозревателя объектов

В открывшемся окне перейдем на вкладку **«Система предупреждений»** (Alert System), установим флажок **«Включить почтовый профиль»** (Enable mail profile), в качестве почтовой системы оставим **«Компонент Database Mail»** (Database Mail) и выберем ранее созданный профиль в соответствующем поле.



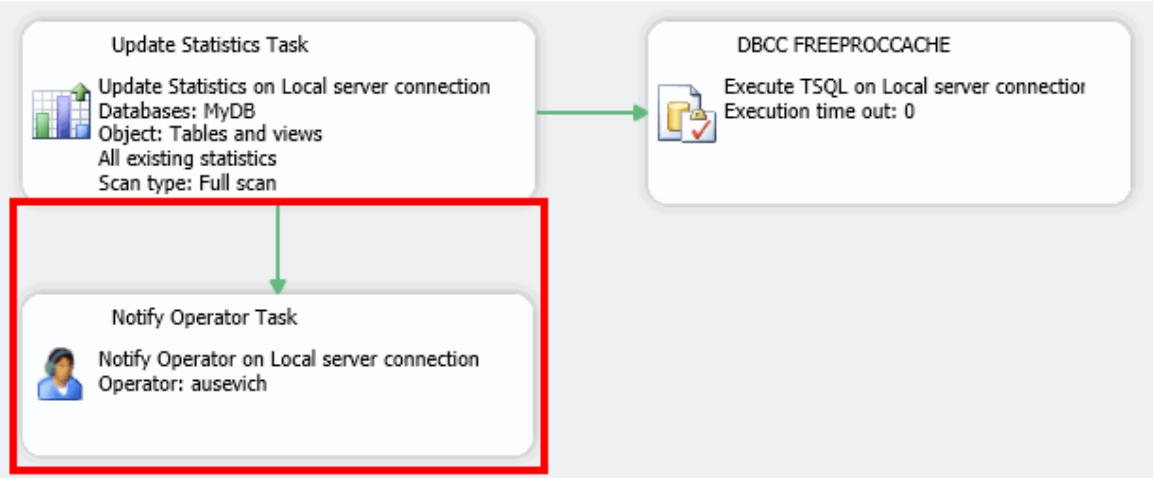
Система предупреждений агента SQL Server

Включение триггеров и задач оповещения

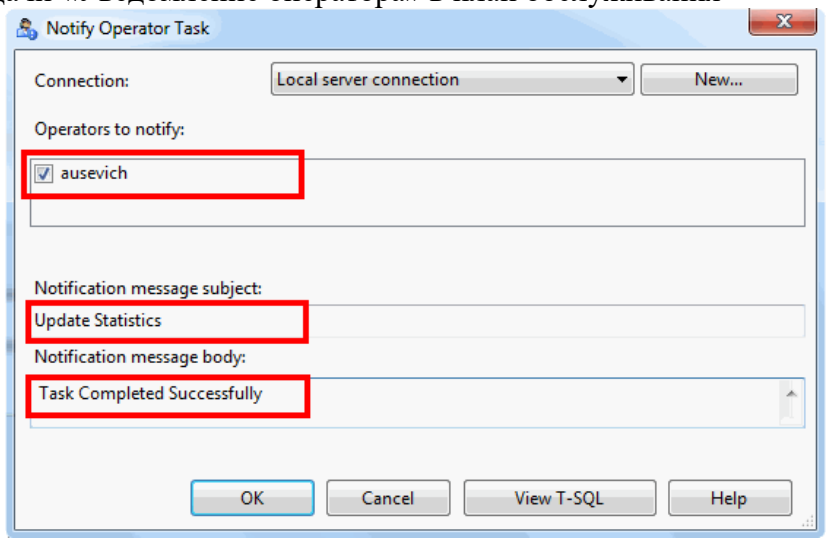
За основу для подключения оповещений возьмем план обслуживания из статьи Механизм «Планы обслуживания» и механизм заданий MS SQL Server.

Для удобства проверки настроим оповещения на успешное выполнение заданий, в реальной жизни больше имеет смысл подключать оповещения на случай ошибки. В целях демонстрации настроим оповещения на наше **«Задание»** (Job), а так же добавим оповещение в наш план обслуживания.

Итак, откроем наш план обслуживания, в нем выделим субплан **«UpdateStatistics»**. На рабочую область субплана перетащим задачу **«Уведомление оператора»** (Notify Operator Task). Протянем стрелку от задачи **«Обновление статистики»**. Двойным кликом по задаче **«Уведомление оператора»** откроем ее свойства, отметим созданного оператора, а так же введем **«Тему»** (Subject) и **«Текст»** (Message) письма. Для того чтобы изменить условие оповещения (по умолчанию стрелка вида **«Успешное завершение»**) надо щелкнуть правой клавишей мыши на стрелке и выбрать ее вид: успешное выполнение/ошибка/выполнение.

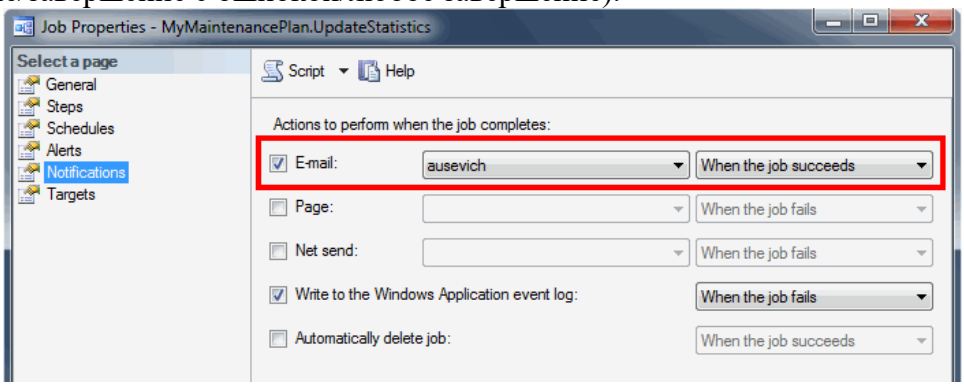


Добавление задачи «Уведомление оператора» в план обслуживания



Настройка свойств задачи «Уведомление оператора»

Теперь перейдем к настройке оповещений в **«Заданиях»** (Jobs). Откроем свойства нашего задания и перейдем на вкладку **«Уведомления»** (Notifications). Установим галки рядом с видами уведомлений, которые мы хотим использовать (у меня это только электронная почта), далее выберем оператора и условие оповещения (успешное завершение/завершение с ошибкой/любое завершение).



Настройка свойств Задания

Проверка работоспособности оповещений

Выполним ручной запуск нашего задания, для этого щелкнем правой клавишей мыши на нем и выберем **«Запустить задание на шаге»** (Start Job at Step). В результате должно прийти на почту 2 письма: одно (с установленными нами темой и текстом) соответствует задаче **«Уведомления оператора»** в плане обслуживания; второе — информирует о выполнении задания в целом.

Уведомления компонента Database Mail

Если не пришло ни одного письма тогда, возможно:

1. Ошибка в параметрах учетной записи SMTP. Попробуйте отправить тестовое сообщение, если оно не отправляется — перепроверьте параметры
2. Ошибка в правах безопасности профиля. Перепроверьте установлены ли права как описано ранее. Если все установлено верно, а результата нет — попробуйте указать все права: открытый и все частные, а так же сделать профилем по умолчанию для каждого права

Если не пришло только письмо с результатом выполнения задания: проверьте что почта SQL Server'а настроена в соответствии с вышеизложенным.

Контрольные вопросы:

1. Перечислите механизм оповещений
2. Перечислите этапы настройки профиля компонента database mail
3. Включение триггеров и задач оповещения, проверка работоспособности оповещений

Лекция 14.
Настройка текущего обслуживания баз данных

Цель: изучить настройки текущего обслуживания баз данных

План занятия:

1. Надежное обслуживание баз MS SQL Server
2. Основные возможности QMB
3. Архитектура
4. Задачи

Надежное обслуживание баз MS SQL Server

Вероятно, вы знаете, что обслуживание баз данных это целый комплекс процедур: создание бэкапов, проверка целостности, обслуживание индексов, статистики и т.д. На просторах сети (да и на Хабре в том числе) на эту тему написано множество статей и рекомендаций. Однако занимаясь внедрением «1С: Предприятие», нам частенько приходится сталкиваться с тем, что обслуживание баз данных настраивается либо неправильно, либо по очень упрощённой схеме. Например, чтобы не заморачиваться с управлением журналами транзакций, для «боевых» баз устанавливается Простая модель восстановления (Simple Recovery model). И это несмотря на то, что потеря информации за пару часов уже критична для компании. Иногда задача по сжатию файлов БД включается в регулярное обслуживание («шобы не росло»), или после обновления индексов идёт уничтожение статистики и прочие подобные ляпы. Так происходит потому, что чаще всего в компаниях нет опытного администратора БД и обслуживанием приходится заниматься кому-то из сотрудников ИТ-службы – «невольному» администратору баз данных (DBA). При этом такой DBA не всегда осознаёт все риски и возложенную на него ответственность.

Для обслуживания баз Microsoft предлагает планы обслуживания (Maintenance Plan) в SQL Server Management Studio (SSMS). Однако как показывает практика, создать и настроить качественный и надёжный план обслуживания может только опытный DBA.

Отмечу, что надёжное обслуживание максимально автоматизировано и не требует регулярного ручного мониторинга администратором, а также гарантирует, что данные удастся восстановить в случае сбоя.

Сторонние программы, которые имеются на рынке и способны облегчить жизнь, в основном автоматизируют создание бэкапов. Выбор таких программ очень широк. Они позволяют делать бэкапы сжатые и шифрованные, на FTP/GoogleDrive/Amazon и так далее. Бэкапы тут можно сравнить с креветками, о которых говорил Бабба в картине «Форест Гамп»: «… их можно жарить, варить, печь, тушить, можно приготовить шашлык из креветок, креветки по-креольски, креветки гамбо, поджаренные с рисом.» программы Quick Maintenance & Backup (QMB), которая поможет вам просто и быстро настроить обслуживание баз данных на Microsoft SQL Server. Бесспорно, что для больших и высоконагруженных баз данных не обойтись без опытного DBA и индивидуального тюнинга производительности, но если вам приходится иметь дело с множеством небольших баз (как правило до 50-80 Гб), то данная утилита будет полезна как новичкам, так и продвинутым пользователям.

Основные возможности QMB

Концепция: доступно новичкам, удобно профессионалам

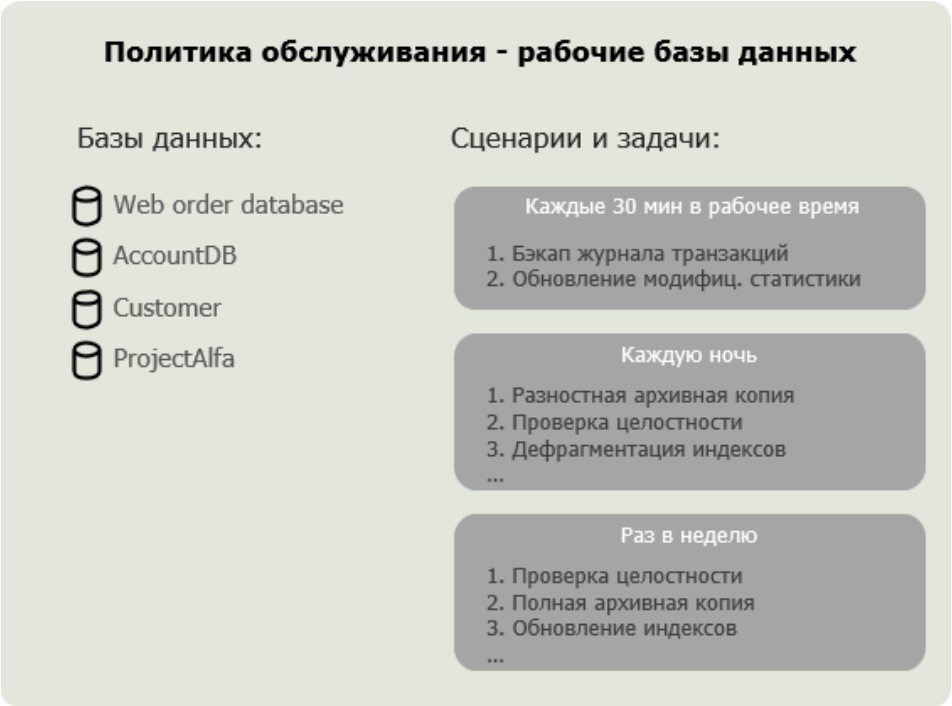
С одной стороны, мы старались сделать программу доступной новичкам и реализовать наиболее распространённые сценарии обслуживания. С другой стороны, мы хотели сделать так, чтобы программа была удобна продвинутым пользователям и помогала им настроить самые разнообразные сценарии, в том числе объединить операции обслуживания баз средствами Transact SQL с другими регламентными процедурами ваших приложений. Например, в QMB можно сделать сценарий, который вначале загрузит данные в «1С: Предприятие», а только потом сделает бэкап и выполнит остальное обслуживание. В итоге получился планировщик, предоставляющий свой фреймворк для исполнения T-SQL скриптов и пакетных файлов (с возможностью хранения результатов их исполнения).

Архитектура

Программа имеет три компонента: GUI клиента, службу QMB Service и файловую базу для хранения своих данных. При установке QMB устанавливаются все три компонента программы. Планы обслуживания не создаются, поэтому не требуется наличия службы агента SQL Server.

Политика обслуживания, сценарии и задачи

Как было сказано выше, QMB не создаёт планов обслуживания на SQL Server. Вместо этого создаётся Политика обслуживания, которая сохраняется в локальном хранилище (файловой базе данных). По сути, политика – это группировка баз данных со схожими свойствами, которые обслуживаются по одним правилам. Политика содержит список баз данных, настройки хранения и копирования бэкапов. В политику входит один или несколько сценариев обслуживания. Сценарий содержит набор задач, исполняемых последовательно для каждой (включенной в политику) базы данных. Если проводить аналогию с планами обслуживания, то сценарий можно сравнить с вложенными Планами обслуживания (Maintenance Plan).



Задача в QMB может иметь один из пяти типов:

- Скрипт T-SQL
- Создание архивной копии (скрипт T-SQL)
- Восстановление архивной копии (динамический скрипт T-SQL)
- Произвольный скрипт (не T-SQL)
- Копирование архивных копий (используется в одноименной системной задаче)

В программе имеется два набора встроенных задач. Первый набор задач базируется на T-SQL скриптах, полученных из открытых источников и созданных разработчиками QMB. Второй набор базируется на скриптах Ола Халенгрэн (администратор баз данных из Швеции), который разработал три популярные хранимые процедуры для обслуживания баз данных. Процедуры Ола устанавливаются автоматически в системную базу данных master, при создании политики из шаблона.

Обслуживание больших и маленьких баз данных. Шаблоны политик

Политику обслуживания можно создать из шаблона или вручную с нуля.

Программа включает 7 шаблонов, которые преимущественно различаются:

Моделью восстановления баз данных. 5 политик с полной моделью восстановления (Full recovery model) и 2 с простой (Simple recovery model).

Порядком обслуживания индексов. Для небольших баз дефрагментация индексов выполняется каждую ночь, а обновление изменившейся статистики в течении дня; для больших баз дефрагментация индексов выполняется раз в неделю.

Набором используемых задач в сценариях. Для обслуживания используются задачи/скрипты Ола Халенгрэн или QMB.

Для рабочих баз данных, с ежедневным оперативным вводом информации (OLTP-базы), рекомендуется выбирать политику с Полной моделью восстановления – например, для баз «1С: Предприятие», в которые ежедневно вводятся данные. Такая модель позволяет восстанавливать базу данных на актуальный или на произвольный момент времени.

Простую модель восстановления рекомендуется использовать для архивных и тестовых баз данных, а также различных хранилищ с редкой эпизодической загрузкой данных.

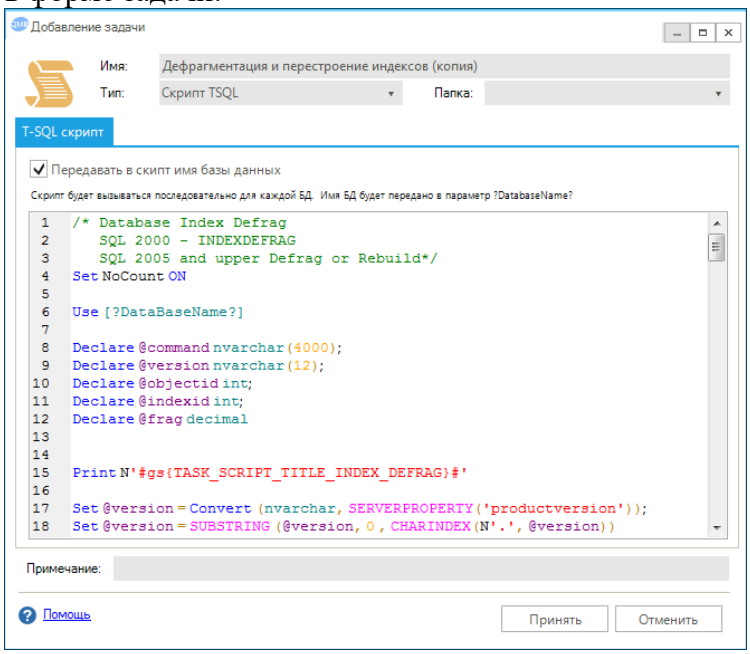
После создания политики из шаблона можно изменить любые её параметры – сценарии и расписание, задачи и порядок уведомлений. В дальнейшем созданную политику можно копировать для других серверов, зарегистрированных в программе.

Задачи

Как было сказано выше, в QMB имеется 5 типов задач – ниже описаны некоторые их особенности.

Исполнение скриптов

Большинство системных задач это T-SQL скрипты. Сам скрипт можно просмотреть в форме задачи:

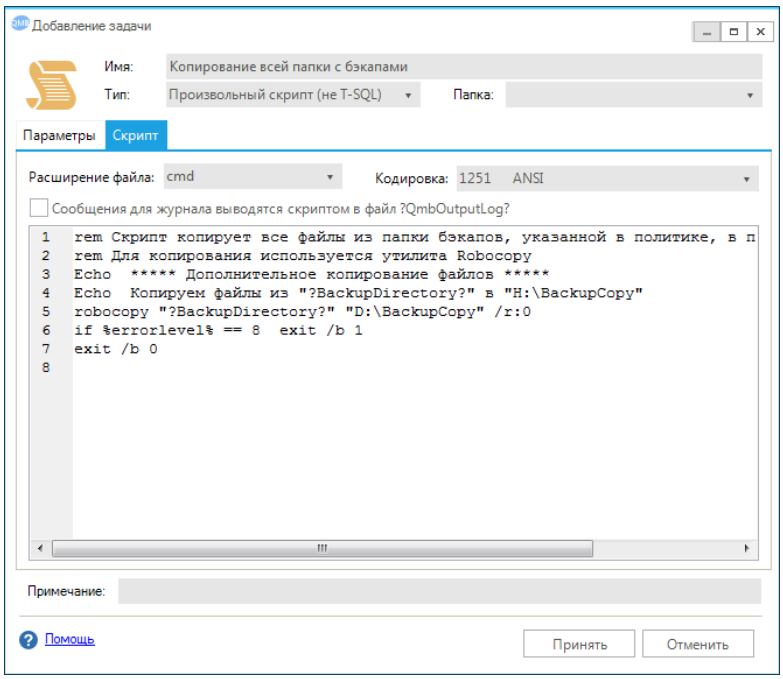


Тексты скриптов (T-SQL, CMD, VBS, PowerShell и других) могут содержать маркеры, которые будут заменены на соответствующие значения, перед его исполнением. Например, маркер? DataBaseName? будет заменён на имя базы данных, а маркер? BackupDirectory? – на путь к каталогу архивных копий, указанному в политике.

Оптимизация окна обслуживания

Бывает, что в ограниченное временное окно необходимо уместить не только обслуживание баз средствами SQL Server, но и исполнение других регламентных операций вашего приложения. Например, тестирование и исправление баз 1С, выгрузку средствами платформы «1С: Предприятие», проведение обмена и т.п. Обычно для этого используется планировщик заданий Windows или планировщик «1С: Предприятие». Однако при этом приходится разносить процедуры во времени с хорошим запасом – так, чтобы они гарантированно не пересекались. В итоге задачи могут не уложиться в имеющееся временное окно.

С QMB можно максимально эффективно использовать окно обслуживания, объединив в сценарии исполнение T-SQL скриптов и пакетных файлов на языках VBS, JavaScript, CMD, PowerShell и других. Ниже показан простой пример задачи альтернативного копирования бэкапов с помощью утилиты Robocopy:



Нужно отметить, что пакетный файл может выполняться как на машине, где установлена программа, так и на стороне SQL Server. Это позволяет оперировать файлами бэкапов на стороне SQL Server. Например, можно написать скрипт, который будет архивировать последний бэкап и выкладывать его в любое облачное хранилище или реализовать собственный алгоритм копирования. В следующих статьях я планирую подробнее рассказать об этой возможности и привести скрипты для работы с базами «1С: Предприятие 8».

Вывод сообщений и журнал обслуживания

Все сообщения, выводимые при исполнении скрипта, перенаправляются в журнал обслуживания программы. Это касается сообщений, выводимых командами print, raiserror для T-SQL скриптов, а также сообщений, выводимых в консоль командами echo, для прочих CMD-скриптов и пакетных файлов. И это здорово! Потому что читабельные и понятные логи – это колоссальная экономия времени, а в качестве бонуса – текст ошибок отправляется в email-уведомлении.

Автоматизированная проверка бэкапов через восстановление

Наличие бэкапов ещё не означает, что удастся восстановить данные при сбое – восстановление может завершаться ошибкой по самым различным причинам. Например, может случится так, что цепочка архивных копий будет прервана, а вы даже не будете знать об этом, пока не попробуете восстановить данные. Именно поэтому лучшие практики говорят, что хороший DBA должен регулярно проверять созданные архивные копии, выполняя восстановление из них. Другого 100% способа просто не существует. Microsoft также рекомендует хотя бы единожды протестировать все архивные копии. Задачи для автоматизированного восстановления в SSMS нету, а ежедневно проверять бэкапы вручную желающих найдётся не много.

В QMB имеется специальная задача, которая последовательно восстановит всю цепочку бэкапов для каждой базы данных политики: Full backup – Differential backup – Transaction log backup. Восстановление выполняется во временную тестовую базу, которая удаляется после проверки её целостности.

Например, у нас в компании на виртуальном SQL Server имеется около 60 небольших баз данных, общим объёмом под 100 Гб. QMB каждую ночь выполняет проверку возможности восстановления всех баз. Проверка занимает около полутора часов и это даёт нам гарантию того, что все резервные копии проверены. Если цепочка бэкапов будет прервана, то придёт уведомление с примерно такой ошибкой:

1. Задача 'Восстановление из арх. копий во временную БД с последующей проверкой целостности' (база данных: Buh_Oazis)

Message: 4305, Level: 16, State: 1, Line: 21

Журнал в этом резервном наборе данных начинается с номера LSN 5235000000291100001, который еще не может применяться к базе данных. Может быть восстановлена более ранняя резервная копия журналов, включающая номер LSN 5228000000281600001.

Message: 3013, Level: 16, State: 1, Line: 21

RESTORE LOG прервано с ошибкой.

Message: 50000, Level: 16, State: 1, Line: 119

В процессе восстановления возникла ошибка

Подобные ошибки случаются редко, как правило, по невнимательности или незнанию сотрудников. В таком случае мы просто делаем дополнительный полный бэкап.

Советы тем, кто захочет настроить подобную проверку:

Операция восстановления ресурсоемкая, поэтому её следует включать в сценарии исполняемые только в нерабочее время.

Для создания временной тестовой базы и восстановления бэкапов в неё требуется запас дискового пространства, равный как минимум самой большой базе данных в политике + 10% от её объёма.

Восстановление больших баз может занимать значительное время. Не включайте задачу по проверке, если не уверены, что операция успеет завершиться в отведённое окно обслуживания.

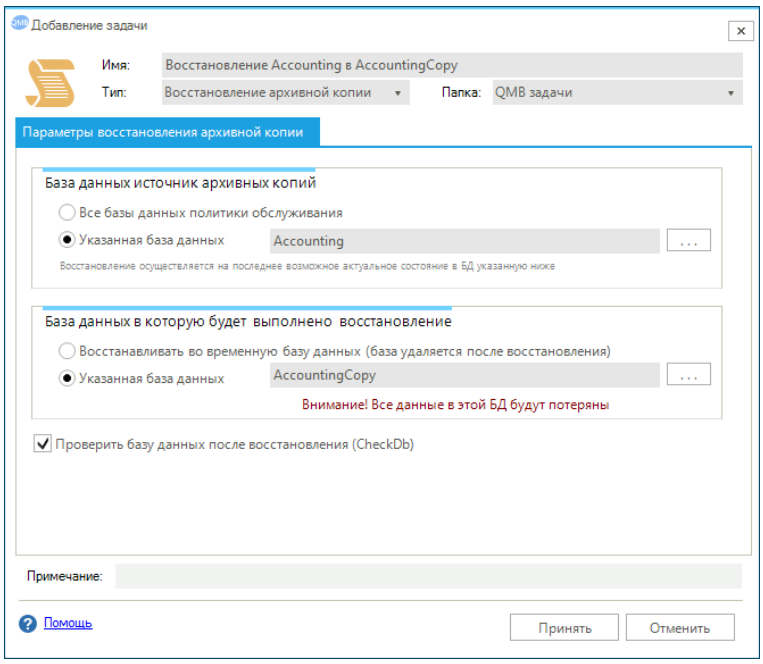
Правильно размещайте задачу в сценарии. Учитывайте, что восстановление выполняется на актуальный момент времени, т.е. на момент исполнения задачи. Например, если задачу проверки бэкапов разместить сразу после создания полной резервной копии, то будет протестирован только последний бэкап, т.к. его будет достаточно для восстановления базы на актуальный момент времени.

Если на проверку бэкапов всех баз политики не хватает окна обслуживания, можно проверять бэкапы только определённых баз данных. Либо распределить задачи по дням недели. Например, сегодня ночью проверить бэкапы баз A и B, а завтра – баз C и D.

Не рекомендуется делать бэкапы в сетевую папку, т.к. при восстановлении приходится «тащить» файлы бэкапов по сети, что значительно увеличивает время восстановления. Правильнее будет настроить создание бэкапов на локальный диск с ежедневным копированием в сетевую папку.

Автоматизированная поддержка копий баз в актуальном состоянии

С помощью программы можно поддерживать копии баз данных в актуальном состоянии. Например, для разработчиков 1С можно каждую ночь актуализировать тестовую базу. Для этого нужно создать задачу, аналогичную встроенной «Восстановление из арх. копий во временную БД». В задаче необходимо указать базу-источник бэкапов и базу, в которую будет выполняться восстановление. И уже потом разместить задачу в ночном сценарии. На рисунке ниже показана задача, которая выполняет восстановление бэкапов базы Accounting в базу данных AccountingCopy. Причём если на SQL Server нет базы данных AccountingCopy, то она будет создана автоматически.



Во время процедуры восстановления база AccountingCopy будет переведена в однопользовательский режим с отключением всех пользовательских соединений.

Копирование файлов бэкапов

В видеоролике было показано, как в программе настраивается дополнительное копирование бэкапов в сетевую или локальную папку. Копирование бэкапов позволяет в определённой степени застраховаться от повреждения файлов, диска или сервера целиком. В случаях же с виртуальным SQL Server, копирование бэкапов на реальный физический диск позволит быстро восстановить одну или несколько баз данных, не дожидаясь восстановления всей виртуальной машины.

Ниже я хотел бы акцентировать внимание на нескольких особенностях копирования бэкапов с помощью QMB:

Частота копирования определяется расписанием сценария, содержащего задачу «Копирование архивных копий». Задачу можно разместить в одном или нескольких сценариях.

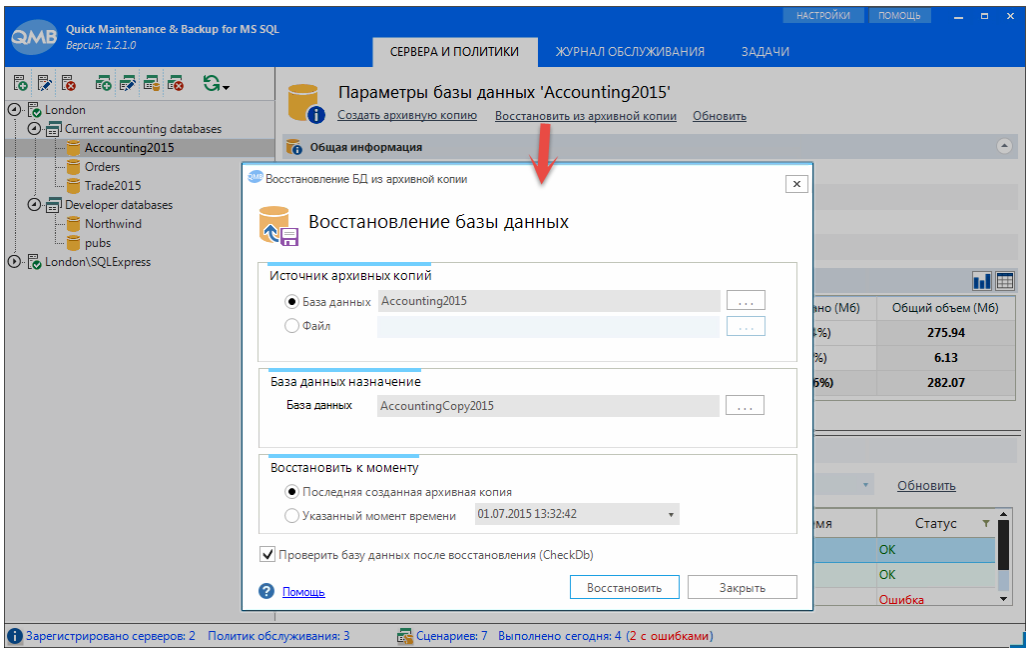
Копируются только новые и изменённые файлы бэкапов – это снижает нагрузку на сеть и позволяет выполнять частое копирование. Например, можно копировать каждый раз после создания нового бэкапа журнала транзакций.

Для сетевой папки можно задать срок хранения файлов. Таким образом, на локальном диске SQL Server можно хранить бэкапы, например, за 1 неделю, а в сетевой папке за 1 месяц.

Возможно настроить копирование бэкапов только для избранных баз политики.

Восстановление баз данных

Восстановить базу данных можно в стандартной консоли SSMS. Однако в QMB есть аналог с более простыми настройками:



Команда «Восстановление из архивной копии» позволяет:

Восстанавливать базу данных из бэкапов на указанный момент времени с автоматическим подбором цепочки бэкапов (если бэкапы создавались на этом же SQL Server).

Восстанавливать базу данных из файла с полной резервной копией.

Восстанавливать бэкапы одной базы данных в другую базу данных, в том числе новую.

Выполнять проверку целостности базы данных после её восстановления.

Оповещения на Email

Если вы когда-нибудь использовали оповещения по электронной почте в SSMS, то наверняка знаете, что сообщения компонента DataBase Mail содержат минимум информации.

Контрольные вопросы:

1. Надежное обслуживание баз MS SQL Server
2. Основные возможности QMB
3. Архитектура и задачи

Лекция 15.

Поиск и решение типичных ошибок, связанных с администрированием

Цель: рассмотреть функции поиска и решения типичных ошибок, связанные с администрированием

План занятия:

1. Ошибки администрирования БД
2. Базовая модель поиска ошибок
3. Стратегии определения ошибок
4. Технологии работы NMS

Ошибки администрирования БД

Администраторы БД совершают много ошибок. Когда ошибки касаются производительности, их надо срочно устранять. В этом блоге будем перечислять распространенные ошибки

Ошибка №1: Размещение файла данных и транзакции БД на одном диске:

Чтение и запись данных в файл БД имеет случайный характер в отличие от транзакций, которые записываются последовательно, т.е. транзакции записываются на диск одна за другой. Распространенная ошибка здесь заключается в том, что транзакции и файлы с данными записываются на один диск. Это приводит к тому, что головки дисков обращаются к конфликтным требованиям случайного и последовательного ввода-вывода. При размещении транзакций на предназначенном для них диске головки дисков остаются в режиме последовательной записи. Это очень важно в работе с приложениями, имеющими очень высокую пропускную способность, чтобы избежать проблем с транзакцией.

Ошибка №2: Использование массив RAID-5:

На мой взгляд, следует избегать массив RAID-5. Первая проблема RAID-5 - калькуляция чётности (Parity). Эта операция снижает производительность записи на диск. Вторая проблема RAID-5 заключается в том, что каждая логическая операция записи требует много физической записи. Я рекомендую RAID-10 .

Ошибка №3:Использование нескольких файлов журнала:

Не выгодно использовать несколько файлов журнала в одной БД. Улучшение производительности происходит только за счет изоляции дисков и использования быстрых дисков. Также не рекомендуется включать опцию " Auto growth", хотя для фиксирования размера файла транзакции надо определить оптимальный размер файла, например для БД системы DIRECTUM, я бы порекомендовал 5 Гб, в том случае если используется тип восстановление "Simple mode".

Ошибка №4: Использование одного файла данных для каждого процессора в целях повышения производительности SQL - I/O операций.

Например, если на сервере установлены 2 процессора, то рекомендуется создавать 2 файла данных для каждой БД. Такие рекомендации были очень популярны между администраторами БД (DBA).Опыт работ показал, что они помогают только в случаях с БД Tempdb, но не пользовательской БД (например БД системы DIRECTUM). Для повышения производительности SQL - I/O операцией при работе с пользовательской БД, рекомендуется создавать Filegroups, которые хранятся на разных дисках или массивах.

Ошибка №5: не ограничивать размер памяти, используемой SQL сервером.

Очень часто встречал администраторов, которые устанавливают SQL server, и не обращаютвнимание на настройки самого сервера, Minimum и Maximum Memory, по умолчанию Min= 0 и Max=2147483647 Мб (почти не ограничен),а в этом случае SQL будет использовать всю доступную память, и в результате, ОС и другие установленные на сервере ПО начинают тормозить.Рекомендуется ограничивать размер памяти, используемой SQL сервером. Также рекомендуется назначать Minimum Memory ровную Maximum Memory и не забывать об 1Гб как минимум для ОС.

Базовая модель поиска ошибок

Базовая модель поиска ошибок предусматривает последовательное выполнение администратором системы следующих действий.

1. Убедиться в том, что ошибки действительно есть. Другими словами, после сообщения пользователя о некорректной работе ИС надо убедиться в том, что этот пользователь выполняет все процедуры корректно и правильно оценивает работу ИС. Например, некая операция действительно занимает много времени, а пользователь считает, что ИС медленно работает.

2. Провести инвентаризацию. Это означает, что необходимо выяснить, все ли части ИС на месте: все кабели существуют, все части ИС взаимодействуют и правильно соединены. При этом NMS может помочь провести автоматический опрос параметров работы оборудования и программного обеспечения, дать план системы. У администратора системы должна быть исполнительная документация по ИС с картой сети и списками всех параметров загрузки серверов, рабочих станций, коммутационного оборудования (worksheet). Нужно убедиться в том, что «все на месте» и соответствует документации.

3. Сделать копии ИС (backup). Причем желательно это делать «быстрыми средствами» (например, не утилитой копирования СУБД, а утилитами ОС «том в том» или «диск в диск»).

4. Сделать перезагружу всех компонент ИС (restart). Есть два режима перезагрузки: холодный режим (с отключением питания) и горячий режим (без отключения питания). При холодном рестарте заново загружается все ПО оборудования, все драйверы, все процессы ОС и СУБД, заново инициализируется память серверов. Поэтому при ошибочных ситуациях надо использовать холодный рестарт. Однако если есть ошибки оборудования, то оно после этого может вообще не загрузиться. Перед перезагрузкой нужно не забыть завершить работу всех процессов различных ОС и СУБД (обычно команды типа Down или Shutdown).

5. После перезагрузки необходимо упростить работу ИС, например, завершить работу всех резидентных программ, не обязательных для работы в простейшем варианте ИС.

6. Если система загрузилась, нужно проверить права и привилегии работающих пользователей (например, одно приложение запускается и работает нормально с данными правами пользователя, а другое нет).

7. Надо убедиться, что версии программного обеспечения являются текущими. Следует работать не на последней версии продуктов, а на стабильной, хорошо отлаженной. Нужно убедиться в том, что никто из пользователей не поставил себе никаких обновлений программного обеспечения. Хотя при правильных действиях АС и NMS такой возможности у пользователя не должно быть.

8. Только после всех перечисленных действий надо собирать информацию об ошибке. Для этого следует проанализировать журналы ИС (логи). Выявить симптомы проблемы, а также тех, кто был ею затронут, проанализировать использование процессов во время возникновения ошибки, изменения, произошедшие в системе, после которых появились сообщения об ошибке в журналах.

9. Необходимо разработать план по изоляции ошибки. Для этого строятся гипотезы о причинах ошибки в ИС. Это могут быть ошибки каналов связи (80% всех ошибок), аппаратные ошибки, ошибки системного программного обеспечения, прикладного программного обеспечения. Всегда следует учитывать, что тираж аппаратных средств больше, чем тираж программных продуктов. Например, процессоров Intel выпускается больше, чем установок какой-либо одной ОС, поэтому аппаратных ошибок будет меньше, чем программных. Аналогично тираж системного программного обеспечения больше, чем тираж прикладного ПО, поэтому в первом меньше ошибок, чем в последнем. Просто чем больше тираж продукта, тем лучше он отлажен.

10. После разработки плана по изоляции ошибки следует ранжировать гипотезы по вероятности их подтверждения. Начинать проверку целесообразно не с самой вероятной гипотезы, а с той, которую можно быстрее всего проверить. Тем самым можно быстро отсечь часть гипотез и сузить процесс проверки.

11. Затем гипотезы проверяются по очереди (строго по одной в единицу времени), в определенной последовательности. В восходящем направлении — от рабочей станции к коммутационной аппаратуре или серверу либо в нисходящем направлении — от сервера или коммутационной аппаратуры к рабочей станции. Для проверки используются только специальные проверенные версии программных

продуктов, специальные тестовые кабели и проверенные надежные тестовые диагностические средства.

12. Наконец, последним действием является документирование проблемы и способа ее решения в специальном журнале. Обязательно должны быть созданы инструкции службам администратора системы по действиям, предотвращающим повторное появление проблемы.

Стратегии определения ошибок

Существуют два подхода к поиску неисправностей — теоретический и практический.

При **теоретическом подходе** специалист-теоретик анализирует ситуацию до тех пор, пока не будет найдена точная причина ошибки. При таком решении, например, сетевой проблемы требуется современный высокопроизводительный протокольный анализатор для набора и анализа огромного количества сетевого трафика в течение значительного времени. Затем сетевому специалисту необходим длительный теоретический анализ данных. Этот процесс надежен, однако не многие компании могут себе позволить, чтобы их ИС или сеть не функционировала в течение нескольких часов или даже дней.

При **практическом подходе** опыт специалиста-практика подсказывает, что при возникновении неисправности целесообразно начинать менять сетевые платы, кабели, аппаратные средства и программное обеспечение до тех пор, пока система не начнет работать. Это вовсе не означает, что все компоненты системы функционируют должным образом, главное, что они вообще функционируют. К сожалению, во многих руководствах по эксплуатации в разделе поиска неисправностей фактически рекомендуется прибегнуть к стилю специалиста- практика, вместо предоставления подробной инструкции по устранению технических неисправностей. Этот подход быстрее предыдущего. Однако он очень ненадежен и первопричина неработоспособности системы может быть так и не устранена.

Ни тот, ни другой метод чаще всего не дают желаемых результатов при поиске и устранении неисправностей. Поэтому действия администратора системы должны базироваться на стратегии управления ошибками.

Стратегия управления ошибками может быть проактивной либо реактивной. С ростом объема ИС возрастает потребность в ее надежности и, соответственно, возрастает потребность в предварительном мониторинге производительности системы, предупреждениях пользователям о возможных проблемах, постоянной бдительности администратора системы. Такая стратегия предупреждения ошибок называется проактивной. Стратегия, при которой АС не предупреждает появление ошибок, а разбирается с ошибками по мере их возникновения, называется реактивной. АС должен приложить усилия и воспользоваться средствами MS или NMS для перехода от реактивной стратегии к проактивной.

Технологии работы NMS

Обычно системы управления отказами (ошибками) — NMS разбивают сложную задачу идентификации и диагностики ошибки на четыре подзадачи:

1. Определение ошибки;
2. Генерация тревожного сигнала;
3. Изоляция ошибки;
4. Коррекция ошибки.

При этом возможны две **технологии работы NMS** — пассивная и активная.

Пассивная технология. С помощью протокола SNMP устройства оповещают управляющую систему о выполнении заранее предусмотренного и заданного параметрами системы условия, например, отличие какого-либо параметра от номинального значения.

Эта технология должна применяться администратором системы при идентификации проблем, не связанных с аппаратными сбоями, на пример, при изменении производительности, проблемах интерфейсов и т. д.

Активная технология. Система NMS тестирует ИС (например, с помощью утилиты PING) и опрашивает каждое из устройств на регулярной основе. Если какое-либо устройство не реагирует в заданный администратором системы интервал времени или его параметры отличаются от желаемых, посылается сообщение администратору системы о сбое устройства. Иногда этот процесс называют up/down monitoring.

АС должен выбрать систему управления, позволяющую использовать обе стратегии. Кроме того, правильно спроектированная система управления дает возможность администратору системы выполнять далее перечисленные логические действия по управлению ошибками.

1. Выбрать время, когда управление ошибками осуществляется полностью, не осуществляется вовсе или осуществляется частично. Время работы ИС определяется в специальном документе — соглашении об уровне сервиса SLA (Service Level Agreement). И это время может отличаться от часов работы данного предприятия. Например, предприятие работает с 9.00 до 18.00, а ИС работает 24 часа, 7 дней в неделю и 365 дней в году. Часть времени ИС может быть занято под специальные действия, не требующие контроля над возможными ошибками. Это можно указать в параметрах настройки MS., например, мониторинг ошибок проводится в течение 20 из 24 часов. Если это требование выполняется, считается, что ошибок нет.
2. При настройке MS создать специальные триггеры, определяющие, какую ситуацию в данной системе следует рассматривать как ошибочную. В некоторых случаях надо подавлять сообщения об ошибках. Например, сообщение о том, что производительность упала на 0,5%, что не существенно для большинства систем.
3. Настроить параметры автоматической перезагрузки системы и переустановки параметров (reset). Можно настроить параметры MS так, чтобы в определенных случаях система сама перезагружалась и устанавливала определенные параметры в номинальные значения.
4. Установить подавление предупреждений об ошибках в некоторых случаях. Например, если известен дефект работы устройства, но он не влияет на работу ИС.

Контрольные вопросы:

1. Перечислите ошибки администрирования БД
2. Опишите базовую модель поиска ошибок
3. Опишите стратегию определения ошибок
4. В чем заключается технология работы NMS

Лекция 16.
Способы контроля доступа к данным и управления привилегиями

Цель: изучить способы контроля доступа к данным и управления привилегиями

План занятия:

1. Двухуровневая защита данных
2. Резервное копирование

Двухуровневая защита данных

После проектирования логической структуры базы данных, связей между таблицами, ограничений целостности и других объектов необходимо определить категории пользователей, которые будут иметь доступ к базе данных.

Как правило, в СУБД организована двухуровневая настройка ограничения доступа к данным. На первом уровне необходимо создать так называемую учетную запись для подключения к самому серверу, что еще не дает возможности доступа к базам данных. На втором уровне для каждой базы данных сервера на основании учетной записи необходимо создать запись пользователя для подключения к базе данных. Учетную запись пользователя можно добавить, изменить или удалить.

Таким образом, сервер обеспечивает двухуровневую защиту данных:

- 1) аутентификацию на уровне сервера;
- 2) идентификацию на уровне базы данных.

Права доступа устанавливаются командой GRANT. Пользователь, выдающий права командой GRANT, может присвоить другому пользователю только те права, которыми он владеет сам, или более слабый набор прав. Для работы с таблицей или с представлением пользователь должен обладать правами на выполнение команд SELECT, INSERT, UPDATE, DELETE или REFERENCES. Для того чтобы задать привилегии на все эти команды сразу, можно использовать привилегию ALL. Для вызова хранимой процедуры в приложении пользователь или объект должны обладать правом на выполнение команды EXECUTE.

Перед тем как присвоить пользователю новые права, необходимо забрать у него старые. Это позволяет сделать команда REVOKE. Привилегии, которые были предоставлены указанному пользователю другими пользователями, не могут быть затронуты оператором REVOKE.

Как правило, в работе с учетными записями используют более крупные объекты — роли. Роль представляет собой объект базы данных, обладающий некоторыми правами и содержащий в себе учетные записи пользователей. Работать с ролью гораздо удобнее, чем с отдельными пользователями. Намного проще переопределить права одной роли, чем каждой учетной записи в отдельности. Роль позволяет объединить в одну группу пользователей, выполняющих одинаковые функции. Для создания роли используется оператор:

CREATE ROLE

Одна учетная запись может состоять сразу в нескольких ролях. Но во время одной сессии клиент может работать только под одной ролью. Это может быть удобно только в тех случаях, когда один пользователь должен иметь разные права доступа в зависимости от ситуации. Пользователь, работающий под правами какой-либо роли, наследует собственные права.

Каждая СУБД должна поддерживать механизм, гарантирующий, что доступ к базе данных смогут получить только те пользователи, которые имеют соответствующее разрешение. Язык SQL включает операторы GRANT и REVOKE, предназначенные для организации защиты таблиц в базе данных. Механизм защиты построен на использовании идентификаторов пользователей, предоставляемых им прав владения и привилегий.

Резервное копирование

Резервное копирование (backup copy) — процесс создания копии данных на внешнем носителе. Оно предназначено для последующего восстановления данных в случае их повреждения или разрушения.

Причины повреждения баз данных могут быть различными, среди них:

• неисправное состояние сервера — жестких дисков, дисковых контроллеров, оперативной памяти компьютера и кэш-памяти RAID-контроллеров и т.п.;

• некорректное соединение с базой данных одного или более клиентов (пользователей);

• файловое копирование или другой файловый доступ к базе данных при запущенном сервере.

Резервное копирование — практически самый надежный способ защиты данных от потери в результате программных и аппаратных сбоев и повреждений. Оно не является обычным файловым копированием. Это считывание информации из базы данных, выполняемое специальной утилитой в режиме клиентского доступа.

Особенности резервного копирования следующие:

копирование базы данных может осуществляться одновременно с работой обычных клиентских программ;

• копия базы данных содержит данные, которые находились в БД на момент начала подключения утилиты, осуществляющей резервное копирование. Все изменения, проводимые выполняющимися параллельно процессу резервного копирования клиентскими программами, в резервную копию не попадают;

• во время резервного копирования происходит считывание каждой записи из всех таблиц в базе данных. Версии записей или их фрагменты, которые не являются актуальными, уничтожаются. Оставшиеся записи оптимизируются;

• в процессе резервного копирования происходит перестройка индекса, что улучшает производительность операций, которые используют эти индексы.

Важно заметить, что недостаточно одного лишь резервного копирования, нужно иногда проверять восстанавливаемость базы данных из резервной копии, потому что бывают случаи, что база данных при нормальном резервном копировании в силу определенных причин не восстанавливается.

Контрольные вопросы:

1. Перечислите два уровня защита данных
2. Дате определение резервному копированию

Лекция 17.
Алгоритм проведения процедуры резервного копирования

Цель: изучить алгоритм проведения процедуры резервного копирования

План занятия:

1. Резервное копирование данных
2. Особенности резервного копирования
3. Полное резервирование
4. Полное резервирование с сохранением журнала
5. Полное плюс разностное резервирование

Резервное копирование данных

Резервное копирование (backup copy) — процесс создания копии данных на внешнем носителе. Оно предназначено для последующего восстановления данных в случае их повреждения или разрушения.

Причины повреждения баз данных могут быть различными, среди них:

• неисправное состояние сервера — жестких дисков, дисковых контроллеров, оперативной памяти компьютера и кэш-памяти RAID-контроллеров и т.п.;

• некорректное соединение с базой данных одного или более клиентов (пользователей);
• файловое копирование или другой файловый доступ к базе данных при запущенном сервере.

Резервное копирование — практически самый надежный способ защиты данных от потери в результате программных и аппаратных сбоев и повреждений. Оно не является обычным файловым копированием. Это считывание информации из базы данных, выполняемое специальной утилитой в режиме клиентского доступа.

Особенности резервного копирования следующие:

• копирование базы данных может осуществляться одновременно с работой обычных клиентских программ;

• копия базы данных содержит данные, которые находились в БД на момент начала подключения утилиты, осуществляющей резервное копирование. Все изменения, проводимые выполняющимися параллельно процессу резервного копирования клиентскими программами, в резервную копию не попадают;

• во время резервного копирования происходит считывание каждой записи из всех таблиц в базе данных. Версии записей или их фрагменты, которые не являются актуальными, уничтожаются. Оставшиеся записи оптимизируются;

• в процессе резервного копирования происходит перестройка индекса, что улучшает производительность операций, которые используют эти индексы.

Важно заметить, что недостаточно одного лишь резервного копирования, нужно иногда проверять восстанавливаемость базы данных из резервной копии, потому что бывают случаи, что база данных при нормальном резервном копировании в силу определенных причин не восстанавливается.

Резервное копирование баз данных является самым простым и дешевым средством обеспечения сохранности корпоративных данных. Не стоит доверять ложному чувству защищенности, возникающему после ввода в эксплуатацию новейшей системы высокой доступности. Если все данные виртуализованы и консолидированы, риски даже возрастают

Полное резервирование

Стратегия полного резервирования является самой простой для понимания и реализации. В конце каждого рабочего дня (или в любой другой промежуток времени, который вы можете назначить) просто запускается процедура полного резервирования базы данных (рисунок 1). При этом не нужно выполнять отдельное резервирование журналов и не требуется использовать дополнительные параметры. Управление файлами в таком режиме резервирования также не требует особого внимания, так как речь идет о единственном файле полной резервной копии. Восстановление из полной резервной копии тоже очень простое: необходимо просто восстановление из единственного файла. Использование полных резервных копий – хороший выбор для организаций с недостаточно опытным ИТ-персоналом.

Воскресенье	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
Полное резервирование	Полное резервирование	Полное резервирование	Полное резервирование	Полное резервирование	Полное резервирование	Полное резервирование

Рисунок 1. Расписание заданий на полное резервирование

Полное резервирование с сохранением журнала

Если недопустима любая потеря данных при восстановлении, можно воспользоваться стратегией полного резервирования с добавлением журнала. Этот метод позволит предотвратить потерю данных; он подходит для часто обновляемых баз данных. Хотя эта стратегия увеличивает сложность операций и сопровождения, общие затраты времени на резервирование базы данных сокращаются.

На рисунке 2 приведен пример расписания для полного резервирования с сохранением журнала – еженедельное полное резервирование по воскресеньям и сохранение журнала транзакций в каждый следующий день до следующего воскресенья, когда снова будет выполнено полное резервирование

Воскресенье	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
Полное резервирование	Резервирование журнала	Резервирование журнала	Резервирование журнала	Резервирование журнала	Резервирование журнала	Резервирование журнала

Рисунок 2. Расписание заданий на полное резервирование с ведением журнала

Для восстановления с полной резервной копии или полной копии с сохранением журнала выполните следующие шаги.

1. Если база данных в состоянии онлайн, ограничьте доступ к ней, переключив режим доступа (в окне свойств) на RESTRICTED_USER. Таким образом доступ к базе данных будет разрешен только членам группы базы данных db_owner и членам групп сервера dbcreator и sysadmin.
2. Выполните резервирование заключительного фрагмента журнала.
3. Исправьте ошибку, вызвавшую крушение базы данных.
4. Выполните восстановление полной резервной копии с параметром NORECOVERY.
5. Если возможно, примените все сохраненные в резервных копиях журналы транзакций с параметром NORECOVERY.
6. Выполните восстановление резервной копии заключительного фрагмента журнала с параметром RECOVERY.

Полное плюс разностное резервирование

В тех случаях, когда требуется дополнительный уровень гарантий, в схему наряду с резервированием журналов можно добавить разностное резервирование. Эта стратегия подходит для транзакционных баз данных, в которые часто добавляются записи, при этом потеря данных при восстановлении недопустима, а администраторы считают приоритетной задачей быстрое восстановление.

Разностное резервирование имеет накопительный характер – оно включает в себя все данные и структуры, которые были изменены с момента последнего полного резервирования вне зависимости от того, когда осуществлялось последнее полное резервирование и сколько раз с того момента выполнялось разностное резервирование.

Воскресенье	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
Полное резервирование	Разностное резервирование	Разностное резервирование	Разностное резервирование	Разностное резервирование	Разностное резервирование	Разностное резервирование

Рисунок 3. Расписание заданий на разностное резервирование

Чтобы восстановить базу данных из разностной резервной копии, выполните следующие шаги.

1. Если база данных в состоянии онлайн, ограничьте к ней доступ, переключив режим доступа (в окне свойств) на RESTRICTED_USER. Тем самым доступ к базе данных будет разрешен только членам группы базы данных db_owner и членам групп сервера dbcreator и sysadmin.
2. Выполните резервирование заключительного фрагмента журнала.
3. Исправьте ошибку, вызвавшую сбой базы данных.
4. Выполните восстановление полной резервной копии с параметром NORECOVERY.
5. Выполните восстановление последней имеющейся разностной резервной копии с параметром NORECOVERY.
6. Выполните восстановление резервной копии заключительного фрагмента журнала с параметром RECOVERY.

Воскресенье	Понедельник утро	Понедельник день	Понедельник ночь	Вторник утро	Вторник день	Вторник ночь
Полное резервирование	Резервирование журнала	Резервирование журнала	Разностное резервирование	Резервирование журнала	Резервирование журнала	Разностное резервирование

Рисунок 4. Расписание заданий на полное, разностное и журнальное резервирование

Контрольные вопросы:

1. Перечислите особенности резервного копирования
2. Опишите полное резервирование
3. Опишите полное резервирование с сохранением журнала
4. Опишите полное плюс разностное резервирование

ЛИТЕРАТУРА

1. Архитектура и технологии IBM eServer zSeries / В.А. Варфоломеев и др. - М.: Интернет-университет информационных технологий, **2015**. - 640 с.
2. Владимир, Михайлович Илюшечкин Основы использования и проектирования баз данных / Владимир Михайлович Илюшечкин. - М.: Юрайт, **2015**. - **516** с.
3. Голицына, О. Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, **2015**. - 400 с.
4. Зубов, А. В. Основы искусственного интеллекта для лингвистов / А.В. Зубов, И.И. Зубова. - Москва: **РГГУ**, **2013**. - 320 с.
5. Илюшечкин, В. М. Основы использования и проектирования баз данных / В.М. Илюшечкин. - М.: Юрайт, Юрайт, **2013**. - 224 с.
6. Илюшечкин, В. М. Основы использования и проектирования баз данных. Учебник / В.М. Илюшечкин. - М.: Юрайт, 2014. - 214 с.
7. Илюшечкин, В. М. Основы использования и проектирования баз данных. Учебник / В.М. Илюшечкин. - М.: Юрайт, 2015. - 214 с.
8. Исаев, Г. Н. Информационные системы в экономике. Учебник / Г.Н. Исаев. - М.: Омега-Л, 2015. - 464 с.
9. Карпова, И. П. Базы данных / И.П. Карпова. - М.: Питер, 2013. - 240 с.
10. Кириллов, В.В. Введение в реляционные базы данных (+ CD-ROM) / В.В. Кириллов. - М.: БХВ-Петербург, **2016**. - **318** с.
11. . Комплекснозначные и гиперкомплексные системы в задачах обработки многомерных сигналов / Я.А. Фурман и др. - М.: ФИЗМАТЛИТ, **2015**. - 456 с.
12. Костин, А. Е. Организация и обработка структур данных в вычислительных системах. Учебное пособие / А.Е. Костин, В.Ф. Шаньгин. - М.: Высшая школа, **2014**. - 248 с.
13. Кудрявцев, В.Б. Интеллектуальные системы. Учебник и практикум для бакалавриата и магистратуры / В.Б. Кудрявцев, Э.Э. Гасанов, А.С. Подколзин. - Москва: **ИЛ**, 2016. - 219 с.
14. Кузнецов, С. Д. Базы данных. Модели и языки / С.Д. Кузнецов. - М.: Бином-Пресс, **2013**. - 720 с.
15. Кузнецов, С. Д. Основы баз данных / С.Д. Кузнецов. - М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, **2017**. - 488 с.
16. Латыпова, Р. Р. Базы данных. Курс лекций / Р.Р. Латыпова. - Москва: **Высшая школа**, 2016. - **177** с.
17. Мартишин, С. А. Базы данных. Практическое примечание СУБД SQL и NoSOL. Учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, Инфра-М, 2016. - 368 с.
18. Миркин, Б. Г. Введение в анализ данных. Учебник и практикум / Б.Г. Миркин. - М.: Юрайт, 2015. - 176 с.
19. Нейрокомпьютеры в системах обработки изображений. - М.: Радиотехника, **2013**. - 192 с.
20. Остроух, А. В. Ввод и обработка цифровой информации / А.В. Остроух. - М.: Академия, **2016**. - 288 с.
21. Персианов, Вячеслав Венедиктович; Технология Проектирования Информационной Базы Для Педагогических Вузов Страны. / Персианов Вячеслав Венедиктович;. - Москва: **Огни**, **2016**. - **594** с.
22. Персианов, Вячеслав Венедиктович; Электронное Образовательное Пространство Педагогического Университета:Формирование, Применение, Проблемы / Персианов Вячеслав Венедиктович;. - Москва: **Гостехиздат**, **2013**. - **195** с.
23. Проектирование баз данных. СУБД Microsoft Access. Учебное пособие. - М.: Горячая линия - Телеком, 2013. - 240 с.
24. Свиридова, М. Ю. Система управления базами данных Access / М.Ю. Свиридова. - М.: Академия, **2016**. - 192 с.

25. Советов, Б. Я. Моделирование систем / Б.Я. Советов, С.А. Яковлев. - М.: Высшая школа, **2015**. - 343 с.
26. Стружкин, Н. П. Базы данных. Проектирование. Учебник / Н.П. Стружкин, В.В. Годин. - М.: Юрайт, 2016. - 478 с.
27. Фуфаев, Э. В. Базы данных / Э.В. Фуфаев, Д.Э. Фуфаев. - М.: Академия, **2016**. - 320 с.
28. Э. В. Базы данных. Учебное пособие / Э.В. Фуфаев, Д.Э. Фуфаев. - М.: Академия, 2014. - 320 с.
29. Хомоненко, А. Работа с базами данных в C++ BUILDER / А. Хомоненко. - М.: Книга по Требованию, **2017**. - 488 с.
30. Цуканова, Н. И. Онтологическая модель представления и организации знаний. Учебное пособие / Н.И. Цуканова. - М.: Горячая линия - Телеком, 2015. - 272 с.

Курс лекций

«Технология разработки и защиты баз данных »

для студентов направления подготовки бакалавров 01.03.02– «Прикладная математика и информатика»

Составители: *Мирземагомедова Мадина Миязуллаховна*

Исабекова Тамила Илахидиновна

Редактор

Подписано в печать
Формат Бумага
Печать
Тираж Заказ №

ИПЦ ДГТУ
367015 Махачкала, пр. Шамиля, 70