**Вопросы к экзамену БД**

**Часть 1- Введение в БД**

1. Приложения БД. Требования к приложениям БД. Процесс разработки БД.

**Основная цель создания приложений БД** – хранение, накопление, обработка и представление информации различного рода. Области и масштабы их применения обширны.

Примеры приложений БД: адресная/телефонная книга, каталог продукции, э-каталог библиотеки.

**Требования** к приложениям БД могут зависеть от предметной области и от масштабов системы: производительность; характер накапливаемой информации; возможность сетевого доступа к данным; масштабируемость; безопасность; переносимость; надежность и отказоустойчивость; …

**Процесс разработки БД**:

Общие стратегии:

*– разработка сверху вниз* (получение абстрактной модели данных исходя из анализа стратегических целей организации, способов их достижения, инфо и способов ее представления, которые необходимы для достижения целей);

*– разработка снизу вверх* (для разработки выбирается конкретная система, которая выполняет только часть функций предприятия; исходя из сформированных требований, выбранная подсистема создается быстрее и с меньшими рисками).

**Одна из основ эффективной реализации** приложения БД – ясное представление модели предметной области.

2. Моделирование данных. Модель «сущность-связь» и модель семантических объектов: основные понятия.

**Моделирование данных** – процесс создания логического представления БД.

Модель данных содержит языковые и изобразительные стандарты для своего представления.

**Модели данных (ANSI)**: внешняя (набор представлений пользователей о той части предметной области, с которой он сталкивается); концептуальная (набор ключевых объектов предметной области, их взаимосвязей и ограничений, накладываемых на объекты); логическая (представление концептуальной модели в соответствии с логической моделью, используемой для хранения данных); физическая (описание физического представления, хранения и обработки данных).

**Модель «сущность-связь» и ее основные элементы**:

Сущность (entity) – некоторый объект, который способен существовать независимо и может быть идентифицирован:

* класс сущностей – набор сущностей определенного типа;
* экземпляр сущности – сущность некоторого класса с заданными значениями атрибутов;
* сущность может являться абстракцией: реального физического объекта; события, действия или процесса (заказ, услуга, транзакция).

Атрибут (attribute) – свойство, которым обладает сущность:

* все сущности определенного класса обладают одинаковым набором атрибутов, но различаются значениями данных атрибутов;
* в общем случае, атрибуты могут быть: составными (composite); многозначными (multi-value);
* идентификатор сущности (entity identifier) – набор атрибутов, которые определяют, именуют сущность;
* идентификаторы могут быть составными, т.е. состоять более, чем из одного атрибута.

Связь (relationship) определяет, как сущности соотносятся друг с другом:

* подобно классам и экземплярам сущностей, можно выделить классы связей и экземпляры связей;
* степень (degree) связи показывает, сколько сущностей определяют эту связь (участвуют в связи): бинарные связи (binary); тернарные связи (ternary).

**Семантическая объектная модель** – альтернативный способ представления модели данных.

Семантический объект (semantic object) – некоторый объект, который способен существовать независимо и может быть идентифицирован (именованный набор  атрибутов, исчерпывающе описывающий отдельную сущность). Аналогично сущностям ER-модели можно определить:

* класс объектов (object class) – набор объектов определенного типа;
* экземпляр объекта (object instance) – объект некоторого класса с заданными значениями атрибутов;
* атрибут (attribute) – свойство, которым обладает объект;
* объект может являться абстракцией: реального физического объекта; события, действия или процесса (заказ, услуга, транзакция).

Атрибут (attribute) – свойство, которым обладает объект:

* все объекты определенного класса обладают одинаковым набором атрибутов, но различаются значениями данных атрибутов (исчерпывающее описание, sufficient description);
* в общем случае, атрибуты могут быть: простыми (simple): атрибут со значением элементарного типа; групповыми, или составными (group): атрибут состоит из набора простых атрибутов; объектными (object): атрибут является семантическим объектом (очевидно, атрибуты данного типа являются <зеркальными> для двух соответствующих объектов).

3. История развития БД. Системы обработки файлов и базы данных. Системы управления БД (СУБД).

**История развития БД:**

до 1970-х : переход от записей вручную к системам обработки файлов

1970 – 1980 : появление первых БД (dBase, System2000)

1978 – 1985 : появление реляционных БД (Oracle)

1982 – 1992 : развитие БД для настольных компьютеров (dBase-II, Access)

1985 – 2000 : появление и развитие ОО БД

1995 – н.в. : интеграция технологий БД и интернет-технологий.

**Системы обработки файлов**: разделенные и изолированные файлы – каждому приложению может соответствовать свой набор файлов; зависимость прикладных программ от форматов файлов и несовместимость файлов; дублирование данных – возможное нарушение целостности данных; трудность представления данных в файлах в различных видах / совместного использования данных; сложность разработки и поддержки.

**База данных** – самодокументированный (т.е. БД содержит не только данные, но и их описание - метаданные) набор интегрированных записей (байты -> поля -> записи -> файлы, метаданные, индексы, метаданные приложения).

**СУБД** – посредник между приложениями и данными:

* изоляция от физической организации хранения данных;
* данные интегрированы;
* уменьшение дублирования данных;
* независимость программ от форматов файлов;
* возможность гибкого выбора формы представления данных.

4. Понятие реляционной модели и нормализации. Реляционные БД. Метаданные в реляционных БД.

**Реляционная модель** определяет способ хранения данных в виде таблиц со строками и столбцами, при этом минимизируется дублирование и исключаются определенные типы ошибок обработки. Она дает стандартный способ структурирования и обработки данных.

**Нормализация** – последовательность преобразований, обеспечивающих определенный набор требований.

**Объекты реляционной БД:**

*основной объект* – отношение, представляющее собой двумерную таблицу (столбцы – поля, атрибуты, определяющие набор данных, которыми обладает описываемый объект; строки – записи, содержащие набор значений атрибутов для конкретного объекта);

*домен* – множество допустимых значений атрибута (физическое описание – тип данных и дополнительные ограничения; семантическое описание – описание назначения данного атрибута);

*ключ* – набор одного или нескольких атрибутов, однозначно идентифицирующий конкретную запись.

**Хранение метаданных в реляционной БД** осуществляется в системных таблицах, которые могут располагаться в специализированной системной БД.

5. Системы управления БД (СУБД). Ядро СУБД. Язык SQL. Дополнительные компоненты современных СУБД.

**СУБД** – посредник между приложениями и данными:

* изоляция от физической организации хранения данных;
* данные интегрированы;
* уменьшение дублирования данных;
* независимость программ от форматов файлов;
* возможность гибкого выбора формы представления данных.

**Ядро СУБД** обеспечивает хранение, обработку и защиту данных (преобразование запросов в действия над данными; безопасность; управление транзакциями и блокировками; резервное копирование и восстановление).

**Компоненты современных СУБД**: ядро; средства интерактивной аналитической обработки данных (OLAP); средства интеллектуального анализа данных; средства интеграции данных; средства создания и публикации форм и отчетов; средства репликации (копирование и распространение данных с синхронизацией); средства выполнения полнотекстовых запросов к неструктурированным символьным данным в таблицах.

**Язык SQL** – язык манипулирования реляционными данными.

Язык представляет собой совокупность операторов (операторы определения данных (CREATE), манипуляции данными (SELECT), определения доступа к данным (GRANT) и управления транзакциями (COMMIT)), инструкций и вычисляемых функций.

6. Системы управления БД (СУБД). Создание БД и основные объекты ядра СУБД.

**СУБД** – посредник между приложениями и данными:

* изоляция от физической организации хранения данных;
* данные интегрированы;
* уменьшение дублирования данных;
* независимость программ от форматов файлов;
* возможность гибкого выбора формы представления данных.

**Создание БД** заключается в определении схемы БД, т.е. набора объектов, которые отражают разработанную модель предметной области: таблиц, связей, доменов и объектов, реализующих бизнес-логику.

**Бизнес-логика** – набор ограничений на возможные действия пользователя с данными в БД. Может быть реализована на уровне СУБД (независимо от источника действий, ограничения выполняются в любом случае); на уровне приложения (ограничения выполняются только в рамках конкретного приложения).

7. Модели БД. Иерархическая и сетевая модели. Реляционная модель.

**Иерархическая модель** – модель данных, в которой данные представляют собой древовидную структуру.

**Сетевая модель** – модель данных с сетевой структурой, возможно наличие циклов (как на уровне модели данных, так и на уровне самих данных).

**Реляционная модель** определяет способ хранения данных в виде таблиц со строками и столбцами, при этом минимизируется дублирование и исключаются определенные типы ошибок обработки. Она дает стандартный способ структурирования и обработки данных.

8. Модели БД. Реляционная модель. Обзор постреляционных моделей (объектно-ориентированные, документо-ориентированные, графовые и столбцовые базы данных).

**Реляционная модель** определяет способ хранения данных в виде таблиц со строками и столбцами, при этом минимизируется дублирование и исключаются определенные типы ошибок обработки. Она дает стандартный способ структурирования и обработки данных.

**Пост-реляционные модели:**

*ОО СУБД* предназначены для прозрачной и унифицированной обработки структур данных ООП, т.к. реляционная модель для этого не подходит (сложность переноса накопленных данных, не обладают достаточной универсальностью);

*хранилища ключей и значений* – хранилища сопоставляют значения ключам, дополнительные возможности: поддержка различных типов значений, поддержка веб-технологий, механизм запросов, коммуникационные средства, также могут рассматриваться более сложные способы организации данных на уровне хранения;

*документо-ориентированные БД* предназначены для хранения и обработки документо-ориентированной информации, основана на понятии документа, как сущности, включающей в себя некоторый набор данных, закодированных в стандартных форматах (XML, JSON, PDF);

*графовые БД* предназначены для хранения структуры графа: узлов и связей между ними (как с узлами, так и со связями могут быть ассоциированы свойства (атрибуты), предоставляющие собой пару ключ – значение и позволяющие хранить дополнительные данные);

*столбцовые БД* ориентированы на хранение данных по столбцам, в отличие от реляционных (по строкам) – несущественные накладные расходы на добавление нового столбца к уже существующим данным, гибкость схемы данных (набор столбцов у различных строк может быть разным), возможна высокая степень сжатия данных для столбцов с повторяющимися значениями.

9. Схемы доступа к данным. Терминальный доступ. Доступ в режиме разделения файлов. Архитектура «клиент-сервер».

10. Схемы доступа к данным. Архитектура «клиент-сервер». Распределенные БД. Параллельные БД.

11. Схемы доступа к данным. Архитектура «точка-точка». Облачные вычисления. БД для мобильных устройств.

12. Управление параллельной обработкой в БД. Приложения БД для оперативной обработки транзакций (OLTP) и поддержки принятия решений (OLAP).

**Часть 2 – Моделирование данных**

1. Процесс разработки БД и уровни моделирования информационных систем. Моделирование данных. Модель «сущность-связь» и ее основные элементы (сущность, связь, атрибуты).

2. Модель «сущность-связь» и ее основные элементы (сущность, связь, атрибуты). Кардинальные числа связей.

3. Модель «сущность-связь». Идентификационно-зависимые, слабые и сильные сущности. Сущности категория-подтип.

4. Модель «сущность-связь»: шаблоны моделирования типа «сопряжение» и «прототип-экземпляр», представление многозначных атрибутов и рекурсивных связей.

5. Процесс разработки БД и уровни моделирования информационных систем. Моделирование данных. Модель семантических объектов и ее основные элементы (объекты, атрибуты).

6. Модель семантических объектов и ее основные элементы (объекты, атрибуты). Кардинальные числа атрибутов. Идентификаторы.

7. Типы семантических объектов. Простой объект. Составной объект. Составной объект с независимыми и вложенными группами.

8. Типы семантических объектов. Сложный объект. Гибридный объект. Ассоциативный объект.

9. Типы семантических объектов. Схема «родитель-подтип». Взаимоисключающие и вложенные подтипы. Схема «прототип-экземпляр».

**Часть 3 – Реляционная модель и преобразование моделей данных**

1. Понятие реляционной модели. Отношения. Ключи, их свойства и типы. Функциональная зависимость. Многозначная зависимость.

2. Реляционная модель: аномалии модификации. Функциональная зависимость. Нормализация. Первая, вторая, третья нормальные формы. Нормальная форма Бойса-Кодда.

3. Реляционная модель: аномалии модификации. Многозначная зависимость. Нормализация. Четвертая нормальная форма. Пятая нормальная форма. Нормализация и денормализация.

4. Преобразование модели «сущность-связь»: основные этапы. Создание таблиц для каждой из сущностей. Определение свойств столбцов таблицы. Нормализация.

5. Преобразование модели «сущность-связь»: основные этапы. Создание связей: сильные сущности, идентификационно-зависимые сущности, шаблоны «сопряжение» и «прототип-экземпляр».

6. Преобразование модели «сущность-связь»: основные этапы. Создание связей: шаблон «категория-подтип», представление многозначных атрибутов. Рекурсивные связи.

7. Преобразование модели «сущность-связь»: основные этапы. Создание связей: ограничения минимальной кардинальности связей.

8. Преобразование модели семантических объектов. Типы объектов: простой, составной, сложный.

9. Преобразование модели семантических объектов. Типы объектов: гибридный, ассоциативный, «родитель-подтип», «прототип-экземпляр».

**Часть 4 – Язык SQL**

1. Ядро СУБД. Язык SQL: стандарты, категории операторов.

2. Язык SQL. Выборка данных: структура оператора SELECT. Указание списка выбора и источника. Использование псевдонимов.

3. Язык SQL. Выборка данных: условие отбора записей и сортировка. Значение NULL и трехзначная логика.

4. Язык SQL. Выборка данных: группировка, условия группировки и функции агрегирования.

5. Язык SQL. Выборка данных: комбинация результатов нескольких выборок и их сортировка.

6. Язык SQL. Выборка данных: соединения и их типы.

7. Язык SQL. Вставка данных: оператор INSERT.

8. Язык SQL. Обновление данных: оператор UPDATE. Условие отбора записей. Значение NULL и трехзначная логика.

9. Язык SQL. Удаление данных: оператор DELETE. Условие отбора записей. Значение NULL и трехзначная логика.

**Часть 5 – Создание баз данных в SQL Server 2012+**

1. Системные базы данных: состав и назначение. Именование объектов базы данных. Схемы.

2. Физическая структура базы данных: файлы и файловые группы.

3. Основные элементы синтаксиса Transact-SQL: именование объектов, идентификаторы, константы, встроенные функции, выражения.

4. Основные элементы синтаксиса Transact-SQL: управление выполнением, обработка ошибок.

5. Категории целостности данных: ссылочная целостность. Каскадные ограничения ссылочной целостности.

6. Категории целостности данных: сущностная, доменная и пользовательская целостность.

7. Типы данных: атрибуты, категории, применение. Уникальные идентификаторы и особенности их использования.

8. Представления: назначение, типы (стандартные, индексированные, секционированные).

9. Индексы: назначение, типы (кластеризованные и некластеризованные).

10. Хранимые процедуры: назначение, типы.

11. Курсоры: назначение, типы, обработка.

12. Определяемые пользователем функции: Скалярные функции. Свойства функций и их влияние на эффективность использования функций.

13. Определяемые пользователем функции: Табличные функции. Сравнение функций и хранимых процедур.

14. Триггеры: назначение, типы. Триггеры DDL.

15. Триггеры DML: триггеры типа INSTEAD OF.

16. Триггеры DML: триггеры типа AFTER.

17. Триггеры DML: сравнение триггеров типа INSTEAD OF и AFTER. Дополнительные средства управления данными, доступные в триггерах.

**Часть 6 - Управление параллельной обработкой в базах данных**

1. Управление параллельной обработкой в БД. Транзакции. Блокировка и взаимная блокировка.

2. Транзакции: свойства, уровни изоляции. Восстановление баз данных.

3. Управление параллельной обработкой в MS SQL Server.

4. Транзакции в SQL Server. Режимы транзакций. Распределенные транзакции.

5. Восстановление баз данных. Восстановление в MS SQL Server.

**Часть 7 – ADO.NET**

1. ADO.NET: режимы работы, объектная модель, базовые интерфейсы и базовые классы. Поставщики данных.

**Режимы работы: режим с поддержанием соединения(связный уровень):**

* используются объекты Connection, Command, DataReader, через которые устанавливается подключение к источнику данных и выполняется с ним взаимодействие.

**Режим без поддержания подключения(несвязный уровень):**

* операции производятся над локальной копией данных источника (объект - DataSet) и связанными объектами (DataTable etc.)
* объект DataAdapter обеспечивает взаимодействие приложения и источника данных, при этом операции открытия и закрытия соединения выполняются автоматически по мере необходимости (операции с объектами несвязного уровня можно выполнять вообще без подключения к источнику данных).

**Объектная модель:**

Connection -> Transaction, Command

DataAdapter -> Command

Command -> Parameter, DataReader

DataSet -> DataTable, DataView, DataRelation

DataTable -> DataRow, DataColumn, Constraint

DataView -> DataRowView

**Базовые интерфейсы и классы поставщиков данных:**

Connection (класс - DbConnection; интерфейс - IDbConnection) - обеспечивает подключение к источнику данных

Command (DbCommand; IDbCommand) - представляет запрос к источнику данных

DataReader (DbDataReader; IDataReader, IDataRecord) - считывает полученные в результате запроса данные.

DataAdapter (DbDataAdapter; IDataAdapter, IDbDataAdapter) - обеспечивает обмен данными между источниками данных объектами набора данных(DataSet)

Parameter (DbDataParameter ; IDataParameter , IDbDataParameter ) - представляет именованных параметр в параметризованном запросе

Transaction (DbTransaction; IDbTransaction) - инкапсулирует транзакцию

**Поставщики данных(data providers)** - это набор классов ADO.NET, которые позволяют получать доступ к определенной БД, выполнять команды SQL и извлекать данные.

Конкретные имена основных классов различаются у различных поставщиков (например, SqlConnection, OracleConnection, OdbcConnection и MySqlConnection), но все эти объекты порождены от одного и того же базового класса (в случае объектов подключения это DbConnection), который реализует идентичные интерфейсы (вроде IDbConnection).

2. Связный уровень ADO.NET. Объекты связного уровня и общий сценарий работы.

Используются объекты Connection, Command, DataReader, через которые устанавливается подключение к источнику данных и выполняется с ним взаимодействие.

Connection (класс - DbConnection; интерфейс - IDbConnection) - обеспечивает подключение к источнику данных

Command (DbCommand; IDbCommand) - представляет запрос к источнику данных

DataReader (DbDataReader; IDataReader, IDataRecord) - считывает полученные в результате запроса данные.

**Общий сценарий работы:**

1. создание, настройка и открытие объекта соединения
2. создание и настройка объекта запроса, в том числе:
   1. определение объекта соединения для объекта запроса
   2. определение строки запроса
3. вызов метода ExecuteReader (ExecuteNonQuery, ExecuteScalar)
4. прокрутка результатов выполнения запроса

3. Связный уровень ADO.NET. Общий сценарий работы. Пул подключений.

Используются объекты Connection, Command, DataReader, через которые устанавливается подключение к источнику данных и выполняется с ним взаимодействие.

**Общий сценарий работы:**

1. создание, настройка и открытие объекта соединения
2. создание и настройка объекта запроса, в том числе:
   1. определение объекта соединения для объекта запроса
   2. определение строки запроса
3. вызов метода ExecuteReader (ExecuteNonQuery, ExecuteScalar)
4. прокрутка результатов выполнения запроса

**Пул соединений:**

* позволяет улучшить производительность за счет повторного использования установленного физического соединения с источником
* поддерживается поставщиками подключений ADO.NET и по умолчанию используется (Pooling = true)
* соединение в пуле, которое не использовалось, уничтожается по истечении таймаута;
* возможна принудительная очистка пулов (вызов функций ClearPool() и ClearAllPools())
* мин/макс размер пула определяется параметром MinPoolSize/MaxPoolSize
* поставщик поддерживает минимальное число соединений в пуле независимо от того, являются ли они активными;
* если достигнут максимальный размер пула и предпринимается попытка открытия нового соединения: производится ожидание освобождения соединения пула в течении таймаута; по истечении таймаута возникает исключение InvalidOperationException;
* при закрытии соединения и уничтожении объекта соединения внутреннее соединение с СУБД не разрывается, а помещается в пул для дальнейшего использования;

4. Связный уровень ADO.NET. Поставщики данных. Создание программного кода, не зависящего от поставщика.

Используются объекты Connection, Command, DataReader, через которые устанавливается подключение к источнику данных и выполняется с ним взаимодействие.

**Поставщики данных(data providers)** - это набор классов ADO.NET, которые позволяют получать доступ к определенной БД, выполнять команды SQL и извлекать данные.

Конкретные имена основных классов различаются у различных поставщиков (например, SqlConnection, OracleConnection, OdbcConnection и MySqlConnection), но все эти объекты порождены от одного и того же базового класса (в случае объектов подключения это DbConnection), который реализует идентичные интерфейсы (вроде IDbConnection).

Для повышения гибкости приложений ADO.NET, которые **не зависят от поставщика**, можно использовать файл конфигурации app.config, элементы которого будут содержать произвольные пары ключ-значение. Чтобы программно получить соответствующий поставщик данных, создается фабрика объектов подключений, позволяющая изменить поставщик без необходимости перекомпиляции кодовой базы. Нужно лишь изменить файл \*.сonfig.

5. Связный уровень ADO.NET. Параметризованные запросы.

Используются объекты Connection, Command, DataReader, через которые устанавливается подключение к источнику данных и выполняется с ним взаимодействие.

string sql = string.Format("Insert Into Inventory" +

"(CarID, Make, Color) Values (@CarID, @Make, @Color)");

// This command will have internal parameters.

using(SqlCommand cmd = new SqlCommand(sql, this.sqlCn))

{

// Fill params collection.

SqlParameter param = new SqlParameter();

param.ParameterName = "@CarID";

param.Value = id;

param.SqlDbType = SqlDbType.Int;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@Make";

param.Value = make;

param.SqlDbType = SqlDbType.Char;

param.Size = 10;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@Color";

param.Value = color;

param.SqlDbType = SqlDbType.Char;

param.Size = 10;

cmd.Parameters.Add(param);

cmd.ExecuteNonQuery();

}

6. Несвязный уровень ADO.NET. Объекты несвязного уровня и общий сценарий работы.

**Режим без поддержания подключения(несвязный уровень):**

* операции производятся над локальной копией данных источника (объект - DataSet) и связанными объектами (DataTable etc.)
* объект DataAdapter обеспечивает взаимодействие приложения и источника данных, при этом операции открытия и закрытия соединения выполняются автоматически по мере необходимости (операции с объектами несвязного уровня можно выполнять вообще без подключения к источнику данных).

Объект **DataAdapter**:

* автономно управляет подключением к источнику данных;
* получает результаты запроса и записывает их в DataSet;
* переносит в БД изменения, внесенные в DataSet;
* устанавливает соответствие между результатами запроса и таблицами /столбцами в DataSet.

**DataSet**:

* содержит данные в виде набора таблиц с возможными связями между ними;
* является источником данных для элементов управления интерфейса пользователя;
* хранит локальную копию исходных данных и внесенных изменений;
* позволяет преобразовывать данные в XML документ и обратно, а также в двоичный формат;

**DataTable**

* содержит данные в виде набора записей, состоящих из одинаковых наборов полей;
* каждая запись хранится как в исходном варианте, так и в текущем – с учетом внесенных изменений.

**DataColumn**

* определяет столбец в таблице и его свойства: тип, значение по умолчанию, может ли содержать NULL, может ли значение быть изменено;
* задает выражение для вычисляемых столбцов.

**DataRow**

* соответствует одной записи в таблице;
* может быть в одном из состояний: Unchanged; Detached; Added; Modified;Deleted;
* содержит следующие версии значений полей: Original; Current; Proposed; Default.

Если вы это читаете, то вам очень не повезло

7. Несвязный уровень ADO.NET. Настройка запросов для синхронизации с источником данных.

**Режим без поддержания подключения(несвязный уровень):**

* операции производятся над локальной копией данных источника (объект - DataSet) и связанными объектами (DataTable etc.)
* объект DataAdapter обеспечивает взаимодействие приложения и источника данных, при этом операции открытия и закрытия соединения выполняются автоматически по мере необходимости (операции с объектами несвязного уровня можно выполнять вообще без подключения к источнику данных).

Если вы это читаете, то вам очень не повезло

**Часть 8 - Создание распределенных баз данных**

1. Методы распределения данных: горизонтальная фрагментация.

**Горизонтальная фрагментация** выполняется операцией селекции, распределяющей кортежи в соответствии с предикатом селекции (CHECK в столбцах ID); таблица разбивается на несколько частей так, чтобы каждая часть содержала записи со значениями только из некоторого диапазона;

**Пример**: на каждом сервере:

CREATE TABLE Table (

ID INTEGER PRIMARY KEY CHECK (ID < 1000),

ColumnA INTEGER,

ColumnB INTEGER)

на сервере, содержащем представление:

USE master

EXEC sp\_serveroption OtherServer', 'lazy schema validation', 'true'

CREATE VIEW View AS

SELECT \* FROM MyDB.dbo.Table

UNION ALL

SELECT \* FROM OtherServer.MyDB.dbo.Table

2. Методы распределения данных: вертикальная фрагментация.

**Вертикальная фрагментация** реализуется с применением операции проекции, необходимы дополнительные соединения для достижения значений нерезидентных атрибутов; таблица разбивается на части так, чтобы каждая часть содержала только некоторые из столбцов исходной таблицы (разбиение строк)

**Пример:** Создание вертикальной фрагментации:

• на первом сервере:

CREATE TABLE Table (

ID INTEGER PRIMARY KEY, ColumnA INTEGER)

• на втором сервере:

CREATE TABLE Table (

ID INTEGER PRIMARY KEY, ColumnB INTEGER)

• на сервере, содержащем представление:

CREATE VIEW View AS

SELECT one.ID, one.ColumnA, two.ColumnB

FROM MyDB.dbo.Table one, OtherServer.MyDB.dbo.Table two

WHERE one.ID = two.ID

3. Методы распределения данных: тиражирование. Связанные таблицы. Обновление данных.

**Тиражирование данных** (Data Replication - DR) - это асинхронный перенос изменений объектов исходной базы данных (source database) в БД, принадлежащим различным узлам распределенной системы. Функции DR выполняет специальный модуль СУБД - сервер тиражирования данных, называемый репликатором (replicator). Его задача - поддержка идентичности данных в принимающих базах данных (target database) данным в исходной БД. Сигналом для запуска репликатора служит срабатывание некоторого правила.

Принципиальная характеристика тиражирования данных заключается в отказе от физического распределения данных (любая база данных для СУБД и для работающих с ней пользователей всегда является локальной)

* на каждом сервере хранится копия исходной таблицы
* тиражирование целесообразно, если менее 5% инструкций, ссылающихся на таблицу, инструкции INSERT, UPDATE или DELETE.

В зависимости от требований к целостности можно использовать два подхода к обновлению данных:

1. Если требуется высокая целостность транзакций, можно создать триггеры, выполняющие распределенное обновление всех копий контексте распределенной транзакции;
2. Если высокая целостность не требуется, можно для переноса изменений от одной копии таблицы на все остальные воспользоваться механизмами репликации SQL Server;

Механизмы репликации:

1. Репликация моментальных снимков (snapshot replication) - распространяет данные целиком и точно в том виде, в котором они были представлены в момент синхронизации, и не контролирует обновления этих данных;
2. Транзакционная репликация (transactional replication) - начинается создания исходного моментального снимка объектов и данных базы данных публикации, последующие изменения данных и схемы на издателе обычно доставляются без задержек (практически в реальном времени), а изменения данных применяются на подписчике в том же порядке и в тех же рамках транзакций, в которых они выполнялись у издателя (в пределах публикации гарантируется согласованность транзакций).

4. Федеративные серверы баз данных. Симметричное и асимметричное секционирование.

Федеративные серверы баз данных - механизм доступа и управления разнородными данными на серверах, управляющихся независимо, но взаимодействующих во время обработки запросов к базе данных

* цель: обеспечение высокого уровня доступности путем создания федерации серверов баз данных (горизонтального секционирования данных базы), что позволит распределить нагрузку на группу серверов, управляющихся независимо, но взаимодействующих во время обработки запросов к базе данных;
* процесс построения федерации серверов баз данных включает разработку набора распределенных секционированных представлений, используемых для распределения данных по серверам;
* секционирование дает хорошие результаты, если строение таблиц в базе данных позволяет поделить их на одинаковые секции так, чтобы большинство строк, к которым обращается любая из инструкций SQL, были бы размещены на одном федеративном сервере.

**Симметричные секции**

* наиболее эффективно секционирование работает в случае, если все таблицы в базе данных могут быть секционированы симметрично (то есть равномерно по федеративным серверам), при этом:
  + взаимосвязанные данные размещаются на одном федеративном сервере так, чтобы большинство инструкций SQL минимально нуждалось в данных на других серверах;
  + большинство инструкций SQL должно обращаться к серверу с 80% требуемых данных, а на распределенные запросы должно приходиться менее 20% данных;
* пример: секционирование данных по регионам
  + большинство запросов нуждается в данных с сервера одного региона;
  + приложения направляют инструкции SQL на федеративный сервер с данными того региона, который был определен из контекста входных данных.

**Асимметричные секции**

* исп-ся в приложениях со сложными схемами доступа к данным, не позволяющими осуществить симметричное секционирование;
* некоторым федеративным серверам приходится назначать более масштабные роли, чем другим. В базе данных могут секционироваться некоторые таблицы (остальные таблицы приходится оставлять на первоначальном сервере), что улучшает производительность
* можно добиться производительности, как и при симметричном секционировании. При этом имеется ряд преимуществ:
  + можно значительно улучшить производительность баз данных, которые не могут быть секционированы симметрично;
  + большую существующую систему можно успешно секционировать посредством серии последовательных, асимметричных процедур секционирования;

5. Проектирование распределенных секционированных представлений.

**Федеративные серверы** баз данных - механизм доступа и управления разнородными данными на серверах, управляющихся независимо, но взаимодействующих во время обработки запросов к базе данных.

Создаются с целью распределить нагрузку на группу серверов.

Секционирование дает хорошие результаты, если строение таблиц в базе данных позволяет поделить их на одинаковые секции так, чтобы большинство строк, к которым обращается любая из инструкций SQL, были бы размещены на одном федеративном сервере

При **проектировании** набора распределенных секционированных представлений для реализации федерации серверов баз данных, необходимо выполнить следующие процедуры:

* определить шаблон инструкций SQL, выполняемых приложением;
* определить, каким образом таблицы связаны друг с другом;
* сопоставить частоту инструкций SQL с секциями, определенными при анализе внешних ключей;
* определить правила маршрутизации инструкций SQL.

6. Создание распределенных секционированных представлений в СУБД SQL Server.

Основные требования к представлениям для распределенного горизонтального секционирования:

* запросы должны быть соединены с помощью UNION ALL;
* базовые таблицы не должны входить в запрос более одного раза;
* столбцы должны быть одного типа, идти в одном и том же порядке и встречаться только один раз;
* столбец, определяющий сегментирование:
  + не должен быть вычисляемым;
  + должен входить в первичный ключ;
  + должен иметь единственное ограничение CHECK с операторами [BETWEEN, AND, OR, <, <=, >, >=, =] , определяющее набор непересекающихся диапазонов значений.

пример:

| -- On server1:  CREATE TABLE USER(  ID INT PRIMARY KEY  CHECK (ID BETWEEN 1 AND 1000),  …  ) | -- On server2:  CREATE TABLE USER(  ID INT PRIMARY KEY  CHECK (ID BETWEEN 1001 AND 2000),  …  ) | -- On server3:  CREATE TABLE USER(  ID INT PRIMARY KEY  CHECK (ID > 2000),  …  ) |
| --- | --- | --- |

-- On each server:

CREATE VIEW userView AS

SELECT \* FROM server1.db.dbo.USER

UNION ALL

SELECT \* FROM server2.db.dbo.USER

UNION ALL

SELECT \* FROM server3.db.dbo.USER

7. Связанные серверы. Нерегламентированные имена (OPENDATASOURCE, OPENROWSET). Сквозные запросы (OPENQUERY).

**Связанным** называется сервер, зарегистрированный на SQL Server 2012 со всеми сведениями, необходимыми для доступа к источнику данных OLE DB, включая данные о:

* поставщике OLE DB - динамической библиотеке, осуществляющей управление и взаимодействие с определенными источниками данных;
* источнике данных OLE DB - конкретной базе данных, доступ к которой выполняется через интерфейс OLE DB;

Доступ к связанному серверу может осуществляться следующими способами:

* через распределенные запросы, обращающиеся к таблицам на связанном сервере при помощи инструкций SELECT, INSERT, UPDATE и DELETE с именем, относящимся к такому серверу;
* через удаленные хранимые процедуры, выполняющие обращения к связанному серверу по имени, состоящему из четырех частей;
* через инструкцию EXECUTE, реализуемую произвольной параметризованной передаваемой командой при помощи расширения AT linked\_server\_name.

в SQL Server макрокоманды OPENROWSET и OPENDATASOURCE предоставляют сведения для подключения, выполняемого с целью доступа к данным из источников данных OLE DB:

**функция OPENROWSET**:

* может использоваться с любым поставщиком OLE DB, возвращающим набор строк;
* может применяться везде, где в инструкции языка Transact-SQL используется ссылка на таблицу или представление;
* требует задания следующих данных:
  + всех сведений, необходимых для подключения к источнику данных OLE DB;
  + имени объекта или запроса, которые должны формировать набор строк.

**функция OPENDATASOURCE**:

* применяется только в тех случаях, если поставщик предоставляет наборы строк и использует нотацию каталог.схема.объект;
* может использоваться там, где синтаксис Transact-SQL позволяет указать имя связанного сервера;
* может использоваться в нотации каталог.схема.объект в качестве ссылки на таблицу или представление;
* требует задания следующих данных:
  + зарегистрированного как PROGID имени поставщика OLE DB, используемого для доступа к источнику данных;
  + строки соединения, задающей различные свойства соединения, передаваемые поставщику OLE DB (последовательность пар "ключевое слово-значение").

**Сквозные запросы:** функции OPENQUERY позволяют выполнить передаваемый запрос к

указанному связанному серверу:

* на функцию ОРENQUERY как на имя таблицы можно ссылаться из предложения FROM, в инструкциях INSERT, UPDATE или DELETE;
* в качестве аргументов нельзя использовать переменные;
* функция OPENQUERY не может быть использована для выполнения расширенных хранимых процедур на связанном сервере.

8. Распределенные запросы. Доступ к удаленным данным: поставщики OLE DB. Ограничения распределенных запросов.

**Распределенные запросы** используются для доступа к данным из нескольких источников данных, которые могут размещаться на одном или различных компьютерах.

Microsoft SQL Server поддерживает распределенные запросы к следующим типам источников данных:

1. Распределенные данные, хранящиеся в нескольких экземплярах SQL Server
2. Удаленные данные, доступ к которым предоставляется с использованием поставщика OLE DB

OLE DB - это низкоуровневый интерфейс для доступа к данным. Он определяет набор COM-интерфейсов для предоставления приложениям единообразного доступа к данным, хранящимся в различных источниках информации (данные предоставляются в табличных объектах, именуемых «наборами строк»).

**(Ограничения?)**При выработке стратегий распределенного выполнения запросов следует:

− Адекватно оценивать интенсивность обмена данными и время доступа к удаленным узлам системы;

− Снизить время вычислений и операций ввода/вывода в запросах;

− По возможности осуществлять параллельную обработку данных на узлах.

Иначе распределенный запрос может выполняться очень медленно.

**(Или)**Распределенные ограничения целостности данных:

* БД разделена на фрагменты таким образом, что родительская таблица находится на одном узле, а дочерняя, связанная с ней по внешнему ключу, – на другом; при добавлении записи в дочернюю таблицу система обратится к узлу, на котором расположена родительская таблица, для проверки наличия соответствующего значения ключа;
* Разбиение одной таблицы на фрагменты и размещение этих фрагментов по разным узлам сети; здесь необходима проверка соблюдения ограничений первичного ключа и уникальных ключей.

9. Распределенные запросы. Оптимизация и выполнение распределенных запросов. Уровни диалекта SQL. Распределённые транзакции.

**Распределенные запросы** используются для доступа к данным из нескольких источников данных, которые могут размещаться на одном или различных компьютерах.

Microsoft SQL Server поддерживает распределенные запросы к следующим типам источников данных:

1. Распределенные данные, хранящиеся в нескольких экземплярах SQL Server
2. Удаленные данные, доступ к которым предоставляется с использованием поставщика OLE DB

OLE DB - это низкоуровневый интерфейс для доступа к данным. Он определяет набор COM-интерфейсов для предоставления приложениям единообразного доступа к данным, хранящимся в различных источниках информации (данные предоставляются в табличных объектах, именуемых «наборами строк»).

**Оптимизация распределенных запросов:**

* индексированный доступ при помощи поставщиков индексов OLE DB: SQL Server может пользоваться стратегиями выполнения, которые включают применение индексов от поставщика индексов для вычисления предикатов и выполнения операций сортировки для удаленных таблиц;
* удаленное выполнение запроса при помощи поставщиков OLE DB SQL-команд: SQL Server пытается Делегировать выполнение как можно большей части распределенного запроса поставщику SQL-команд.

**Удаленное выполнение распределенных запросов**

Запрос, который производит доступ только к удаленным таблицам, находящимся в источнике данных поставщика, извлекается из исходного распределенного запроса и выполняется поставщиком

На степень делегирования выполнения распределенного запроса влияют следующие факторы:

* уровень диалекта, поддерживаемый поставщиком SQL-команд: SQL Server делегирует только те операции, которые поддерживаются имеющимся уровнем диалекта;
* совместимость параметров сортировки: для распределенного запроса семантика сравнения для всех символьных данных определяется кодировкой и порядком сортировки локального экземпляра SQL Server. SQL Server может делегировать операции сравнения и ORDER BY для символьных столбцов поставщику только в том случае, если тот сможет определить, что
  + столбец в источнике данных имеет ту же кодировку и параметры сортировки;
  + семантика сравнения символов соответствует стандартам SQL-92 и SQL Server.

**Уровни диалекта SQL**

существуют следующие уровни диалекта, в порядке убывания от старшего к младшему (каждый из перечисленных уровней диалекта является надмножеством более низких уровней):

* SQL Server - внешние соединения (OUTER JOIN), CUBE, ROLLUP, оператор получения остатка от деления (%), битовые операторы (&, I, ^, ~), строковые и системные арифметические функции;
* SQL-92 начального уровня (SQL-92 Entry Level) - UNION, UNION ALL;
* базовый ODBC(ODBC Core) - агрегационные функции с ключевым словом DISTINCT и символьные константы;
* Jet (SQL Minimum) - статистические функции без ключевого слова DISTINCT, сортировка (ORDER BY), внутренние соединения (INNER JOIN). предикаты, операторы вложенных запросов (EXISTS, ALL, SÔME, IN), DISTINCT, арифметические операции и константы, не упомянутые на более высоких уровнях, и логические операторы;

дополнительные свойства SQL, поддерживаемые провайдерами:

* поддержка GROUP BY и HAVING;
* поддержка подзапросов (вложенных запросов);
* поддержка констант даты/времени;
* поддержка LIKE;
* поддержка внутренних соединений;
* поддержка параметрических запросов (маркеров параметров).

**Распределенные транзакции**

Участники распределенной транзакции:

* два или более сервера, которые называются диспетчерами ресурсов (экземпляры SQL Server Database Engine);
* диспетчер транзакций, который обеспечивает управление транзакцией путем координации между диспетчерами ресурсов

Двухфазовая фиксация транзакций (two-phase commit protocol - 2PC):

* фаза подготовки: каждый из диспетчеров ресурсов обеспечивает устойчивость транзакции, в частности, записывая журнал транзакции на диск;
* фаза фиксации: начинается в случае успешного завершения подготовки всеми диспетчерами ресурсов и завершается после получения диспетчером транзакций подтверждений об успешной фиксации от каждого из диспетчеров ресурсов;

Управление распределенными транзакциями может осуществляться через вызов инструкций Transact-SQL (как непосредственно, так и через ADO, OLE DB и ODBC), а также через вызов функций API OLE DB и ODBC.