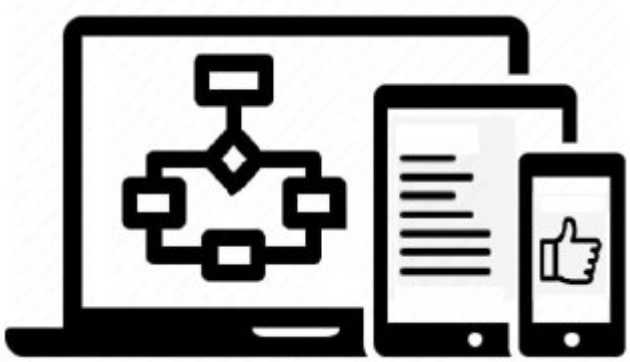
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

БАЗЫ ДАННЫХ

Методические рекомендации к лабораторным работам для студентов направления подготовки 01.03.04 «Прикладная математика» очной формы обучения



Могилев 2022

УДК 004.65
ББК 32.973.26-0.18.2
Б17

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «16» сентября 2022 г., протокол № 2

Составители: канд. техн. наук, доц. К. В. Захарченков;
канд. техн. наук, доц. Т. В. Мрочек

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации содержат описание восемнадцати лабораторных работ», выполняемых при изучении дисциплины «Базы данных». Рассматриваются основы работы с CASE-средством Sparx Systems Enterprise Architect, Microsoft Office Access, Microsoft SQL Server и языком Transact-SQL.

Учебно-методическое издание

БАЗЫ ДАННЫХ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Разработка технического задания на проектирование информационной системы

Цель: разработать проект технического задания на проектирование базы данных информационной системы для выбранной предметной области.

Теоретические положения

База данных (БД) является основным компонентом любой информационной системы (ИС), поэтому проект технического задания (ТЗ) разрабатывается на основе [1] и имеет структуру, приведенную далее в примере.

Пример технического задания на проектирование БД ИС «*Библиотека*».

1 Общие сведения.

1.1 Объект автоматизации – университетская библиотека.

1.2 Документы, на основании которых создается система.

Систематический, алфавитный и предметный каталоги; библиотечно-библиографическая классификация (ББК); должностные инструкции; правила пользования библиотечным фондом; акт на списание книг.

2 Назначение и цели создания системы.

2.1 Назначение системы.

Проектируемую БД предполагается использовать на рабочих местах библиотекарей для увеличения скорости обслуживания читателей. Система позволит облегчить процесс поиска книг, т. к. он будет вестись автоматизированно. Применение БД позволит упростить процессы подбора литературы, проверки наличия книг в фонде, слежения за сохранностью книжного фонда.

2.2 Цели создания системы.

Систему предполагается создать для улучшения качества обслуживания читателей и ускорения работы библиотекаря. Так как система позволит увеличить скорость обслуживания, то возрастет число обслуживаемых читателей.

Критерии оценки достижения целей системы:

– увеличение числа обслуживаемых читателей за счет увеличения скорости обслуживания;

– уменьшение вероятности потери информации о книгах;

– уменьшение вероятности неверного закрепления книг за читателем.

3 Характеристика объектов автоматизации.

3.1 Краткие сведения.

Примечание – Здесь необходимо:

– выполнить описание работы объекта автоматизации (например, отдела предприятия);

– перечислить основные функции объекта автоматизации и указать информацию, подлежащую хранению;

– перечислить категории пользователей будущей БД, определить права доступа разных категорий пользователей к различной информации в БД.

Университетская библиотека включает следующие отделы: отделы обслуживания (абонемент учебной литературы, абонемент научной литературы, читальный зал), отдел комплектования, справочно-библиографический отдел.

Каждый отдел библиотеки выполняет свои функции.

Далее в *примере* будет приведено описание только для справочно-библиографического отдела, который выполняет следующие функции:

– обработка каталожных карточек;

– проведение библиотечных мероприятий (проведение занятий со студентами на первом курсе, выставок);

– издание тематических списков литературы.

Информация, подлежащая хранению: инвентарный номер книги (шифр), автор, название, издательство, год издания, цена книги, отдел, где хранится книга.

Пользователи будущей БД: директор библиотеки, заместитель директора библиотеки, заведующие отделами, библиотекари.

Далее приведен *пример* описания функций библиотекаря.

В функции библиотекаря входит:

– запись читателей в библиотеку;

– выдача и прием книг.

Информация, подлежащая хранению: номер читательского билета (номер формуляра), имя и фамилия студента, год поступления, год окончания (отчисления), факультет и специальность, форма обучения (дневная или заочная), инвентарный номер книги (шифр), автор, название, издательство, год издания, цена книги, отдел, где хранится книга.

Библиотекарь имеет доступ к информации о читателях, о книжном фонде.

3.2 Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

В отделах обслуживания БД будет использоваться для поиска книг, для поиска читателя по номеру читательского билета, просмотра его задолженностей, внесения сведений о выдаваемых книгах.

4 Требования к системе.

4.1 Требования к системе в целом.

Система должна удовлетворять следующим требованиям:

– надежности;

– безопасности;

– защиты информации от несанкционированного доступа;

– доступности БД с любого компьютера в библиотечной сети;

– защищенности информации, хранящейся в системе, от аварийных ситуаций, влияния внешних воздействий;

– к квалификации персонала. В библиотеке работают служащие с высшим и средним специальным образованием. Персонал должен быть обучен правилам работы с ИС. Наличие специального технического образования не требуется.

4.2 Требования к функциям (задачам), выполняемым системой.

Примечание – При перечислении функций рекомендуется указать форму получения информации по каждой функции (в виде запроса (результатом выполнения которого является виртуальная таблица) либо отчета (или иных видов бумажной документации)).

Примеры функций, выполняемые подсистемами объекта автоматизации.

4.2.1 Запрос информации о книгах, выданных студенту, сотруднику.

4.2.2 Формирование тематических списков литературы.

4.2.3 Предоставление информации для отчета о работе справочно-библиографического отдела.

4.3 Требования к видам обеспечения.

Программное обеспечение системы не должно зависеть от аппаратных средств компьютера. Необходимое программное обеспечение: MS Word 2019, MS Access 2019, MS SQL Server 2019.

Задание

Выбрать предметную область (согласовать с преподавателем тему) и разработать техническое задание на проектирование базы данных информационной системы для выбранной предметной области.

Содержание отчета: тема и цель работы; техническое задание объемом не менее 5 страниц; в пункте 4.2 должно быть указано не менее 15 функций.

Контрольные вопросы

1 Указать состав и содержание ТЗ на автоматизированную ИС.
2 Каковы правила оформления ТЗ?
3 Каков порядок разработки, согласования и утверждения ТЗ на ИС?

2 CASE-средства концептуального проектирования функциональных моделей информационных систем

Цель: разработать функциональную модель информационной системы для выбранной предметной области с использованием методологии BPMN и CASE-средства концептуального проектирования функциональных моделей информационных систем Sparx Systems Enterprise Architect.

Теоретические положения

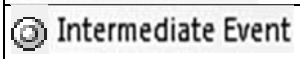
Функциональная модель ИС отображает функциональную структуру системы, т. е. выполняемые системой действия по преобразованию входов системы в выходы и связи между этими действиями.

BPMN (Business Process Modeling Notation, нотация моделирования бизнес-процессов) – это нотация (система условных обозначений и правил по их использованию), предназначенная для описания предметной области реального бизнеса, т. е. для моделирования бизнес-процессов.

При создании новой функциональной модели в Sparx Systems Enterprise Architect необходимо выбрать File → New project, задать имя файла и тип проекта Enterprise Architect Project (*.eap). Далее в окне Model Wizard в разделе BPMN следует выбрать тип модели BPMN 2.0 Business Process, добавляемой в основной

пакет Model. В таблице 2.1 приведен перечень основных элементов диаграммы BPMN Business Process, используемых в функциональном моделировании.

Таблица 2.1 – Элементы диаграммы BPMN Business Process в Enterprise Architect

Изображение элемента	Наименование элемента
1	2
Элементы управления (события, действия, шлюзы)	
	Стартовое событие, произошедшее в описании процесса
	Промежуточное событие, возникающее между стартовым и конечным событиями
	Конечное событие, указывающее, в какой точке завершается процесс
	Бизнес-процесс
	Деятельность, задача, подпроцесс – действия (обозначаемые глаголами), которые должны быть выполнены на определенном этапе бизнес-процесса. Действия бывают следующих видов: процесс – сложное действие, подлежащее дальнейшей декомпозиции при моделировании задача – элементарное действие, которое уже не может быть дальше декомпозировано. Подпроцесс – это деятельность верхнего уровня (помеченная графическим элементом ⊞), которая может быть смоделирована с использованием деятельностей, шлюзов, событий и потоков последовательности. Для того, чтобы создать блок действия подпроцесса, нужно для блока данного действия выбрать Properties → Type: subProcess. Чтобы привязать к блоку действия подпроцесса диаграмму декомпозиции, нужно в контекстном меню блока выбрать New Child Diagram → Add diagram → BPMN 2.0 → Business Process На одном уровне декомпозиции желательно размещать не более 9 действий. Типы задач в нотации BPMN 2.0 (Properties → Type процесса):
	abstract task (задача общего типа)
	manual task (задача, выполняемая вручную, исключающая применение приложений, выполняемая за рамками автоматизированной информационной системы (например, проведение совещания))
	business rule task (бизнес-правило – задача, обеспечивающая доступ к механизму бизнес-правил и получение на выходе представленной этим механизмом информации об изменениях в бизнес-процессе)
	user task (задача, типичная для технологического процесса (упорядоченной последовательности взаимосвязанных действий), где пользователь выступает в роли исполнителя и выполняет задачи с помощью других людей или программного обеспечения)
	receive task (получение сообщения)
	send task (отправка сообщения)
	service task (сервисная задача, предназначенная для оказания услуги, которая может, например, являться веб-сервисом или автоматизированным приложением)

Продолжение таблицы 2.1

1	2
Gateway	Шлюз (развилка) – логический оператор, с помощью которого организуется ветвление и синхронизация потоков управления в модели процесса. Отображает точки принятия решений в процессе. Типы логических операторов: Exclusive – оператор исключающего ИЛИ. При ветвлении потоков исполняется одна ветвь, для которой истинно условие. При слиянии потоков после завершения одной входящей ветви активирует исходящий поток Event-Based – событийный оператор исключающего ИЛИ. Поток управления направляется по той ветви, где событие произошло раньше. Для слияния потоков не используется Parallel Event-Based – параллельный событийный оператор. При запуске одной ветви процесса по событию ожидаются также другие события, и при их наступлении запускаются соответствующие ветви Inclusive – включающий оператор И/ИЛИ, используется для разделения потока операций на несколько альтернативных и параллельных маршрутов  Complex – комплексный оператор. Используется для моделирования сложных условий ветвления и слияния Parallel – параллельный оператор И, который используется для создания параллельных маршрутов (без необходимости проверки каких-либо условий) и их объединения. При разделении на параллельные потоки все ветви активируются одновременно
Зоны ответственности	
Pool	Пул (область, набор) – это объект, использующийся для отображения области процесса (совокупности всех действий и ответственных за их выполнение лиц). Пул не отображает конкретные внутренние процессы участников, он показывает глобальные взаимодействия и зависимости между участниками процесса. Пул может быть не изображен на диаграмме, но он всегда есть. На одной диаграмме может быть несколько пулов. Пул может также содержать несколько дорожек
Lane	Дорожка – отображает зону ответственности (внутреннюю роль) участника процесса (директора, менеджера и т. п.). В дорожке описываются все действия лица, ответственного за выполнение задач. Дорожки могут располагаться как вертикально, так и горизонтально
Данные	
Data Object   	Объект данных – это элемент, который показывает, какие данные и документы нужны для того, чтобы какое-то действие запустилось, или являются результатом выполненного действия. Типы объектов данных: входные данные (внешний вход, который может использоваться для процесса, задачи) выходные данные (результат выполнения процесса, задачи) коллекция объектов данных, представляющая группу объектов, несущих информацию, например, список заказанных товаров (Properties DataObject: isCollection=true)
Data Store	Хранилище данных – объект, который процесс может использовать для записи и извлечения данных, например, база данных или таблица

Окончание таблицы 2.1

1	2
Message  	Сообщение – элемент, отображающий коммуникацию между двумя участниками процесса (e-mail, sms-сообщения, переписка в мессенджере, которыми пользуются участники процесса, коммуникации на сайте компании и т. д.). Типы сообщений: инициирующее (полученное) (Properties: isInitiating=true) сообщение-ответ (отправленное) (Properties: isInitiating=false)
Соединительные элементы	
Sequence Flow	Последовательный поток – стрелка, показывающая последовательность действий
Association	Ассоциация – позволяет установить соответствие между артефактом и элементом потока (событие, действие, шлюз)
Message Flow	Поток сообщений – показывает сообщения, которыми обмениваются участники бизнес-процесса. Также Message Flows может связывать два отдельных пула в диаграмме
Data Association	Ассоциация данных – соединяет элемент данных (объект данных, хранилище) с элементом потока (событие, действие, шлюз)
Артефакты (*для добавления дополнительной информации о процессе*)	
Group	Группа объектов – прямоугольник, объединяющий несколько действия, принадлежащих к одной категории, но не являющийся действием (задачей, подпроцессом) и не влияющий на поток управления
Text Annotation	Текстовая аннотация – комментарий, присоединяемый к какому-либо элементу диаграммы для улучшения читабельности диаграммы

Пример разработки функциональной модели ИС «*Библиотека*».

Вначале разрабатывают модель верхнего уровня процессов, отображающую подпроцессы в ИС (рисунок 2.1), затем проводят декомпозицию подпроцессов – строят модели, отображающие детали подпроцессов (рисунки 2.2 и 2.3).

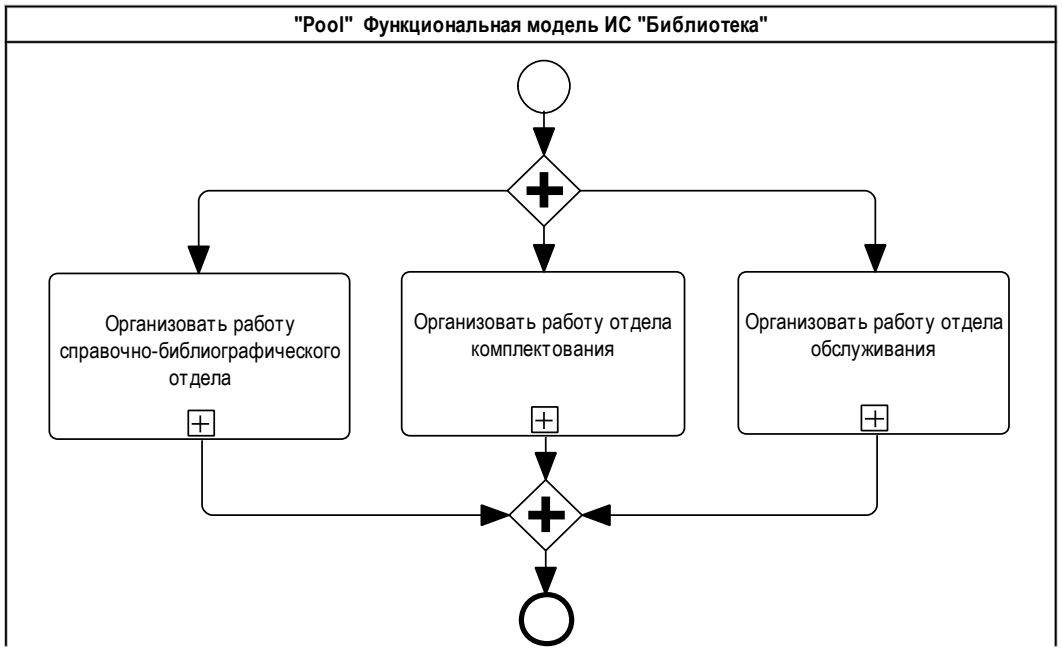


Рисунок 2.1 – Диаграмма подпроцессов функциональной модели (первый уровень)

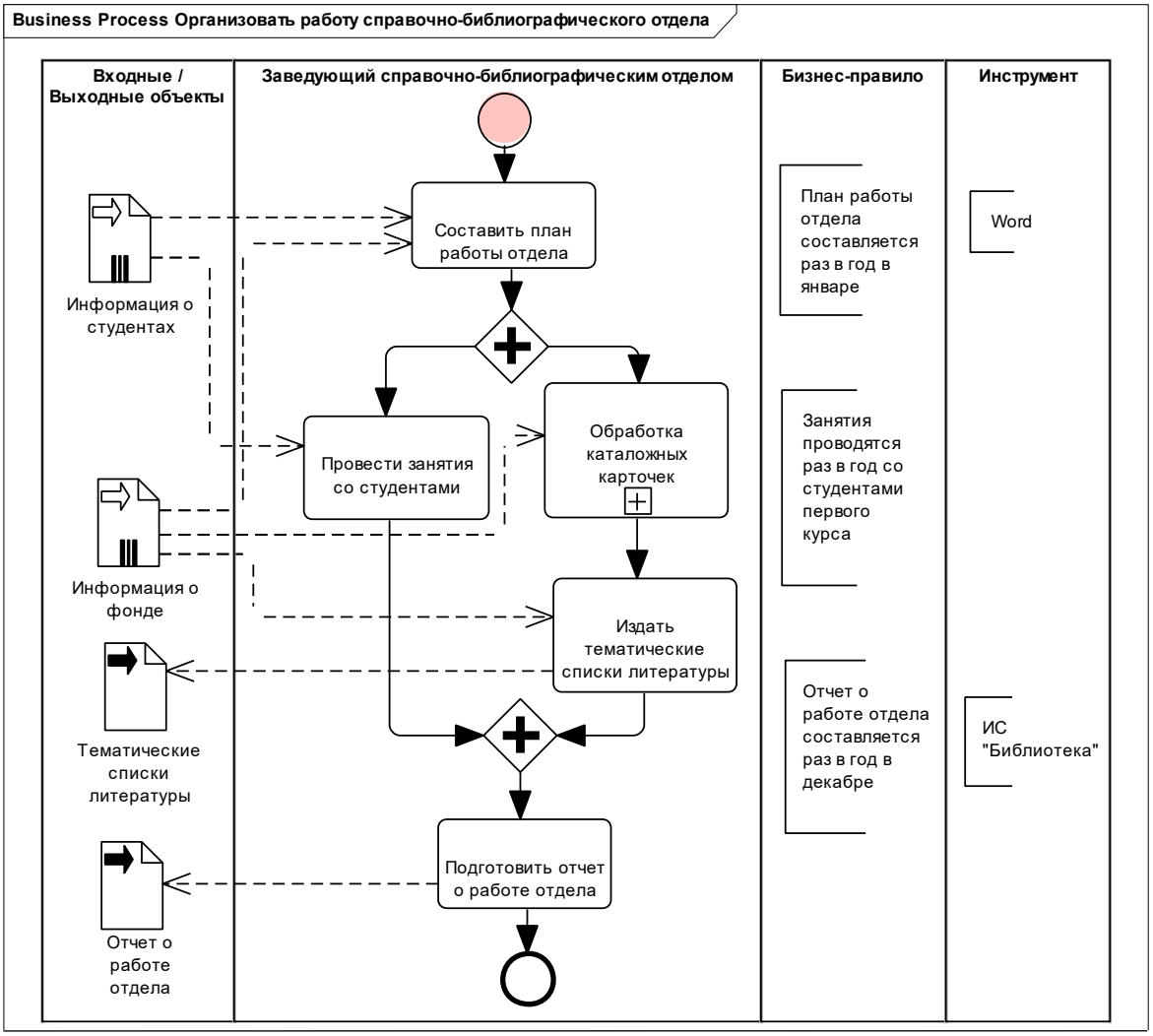


Рисунок 2.2 – Диаграмма декомпозиции подпроцесса «Организовать работу справочно-библиографического отдела» (второй уровень)

Процесс начинается с начального события, состоит из всех действий, которые следует выполнить для достижения цели, и завершается достижением цели или запуском других процессов. Процесс следует понимать, как сквозной набор работ (например, «от поступления заявки до оплаты»), пересекающий всю систему, чтобы выполнить поставленную цель и доставить ценность потребителю.

Для составления функциональной модели выполняют следующие действия.

1 На основе технического задания составляется список действий в моделируемой системе (вначале описывается линейная последовательность действий от начала к результату, далее при необходимости добавляют ветвления).

2 Действия переводятся в задачи (описываемые глаголами в неопределенной форме), т. е. ответы на вопрос «что нужно сделать». Действие может быть сложным и комплексным (например, автоматизировать деятельность отдела), а задача – это простое конкретное действие, выполняемое непосредственно исполнителем (автоматизация деятельности отдела делится на части и выстраивается последовательность). Не рекомендуется использовать в наименовании задачи союз «и», т. к. он объединяет две разные задачи (например, подготовить отчет о деятельности отдела и табель рабочего времени).

3 Определяются исполнители – должностные лица, ответственные за выполнение действий и задач. При необходимости определяются внешние сущности, которые находятся за рамками анализируемой системы (например, клиенты, поставщики) и являются необходимыми для получения результата.

4 Исполнителям назначаются действия.

5 Описываются условия выполнения процессов и задач (шлюзы).

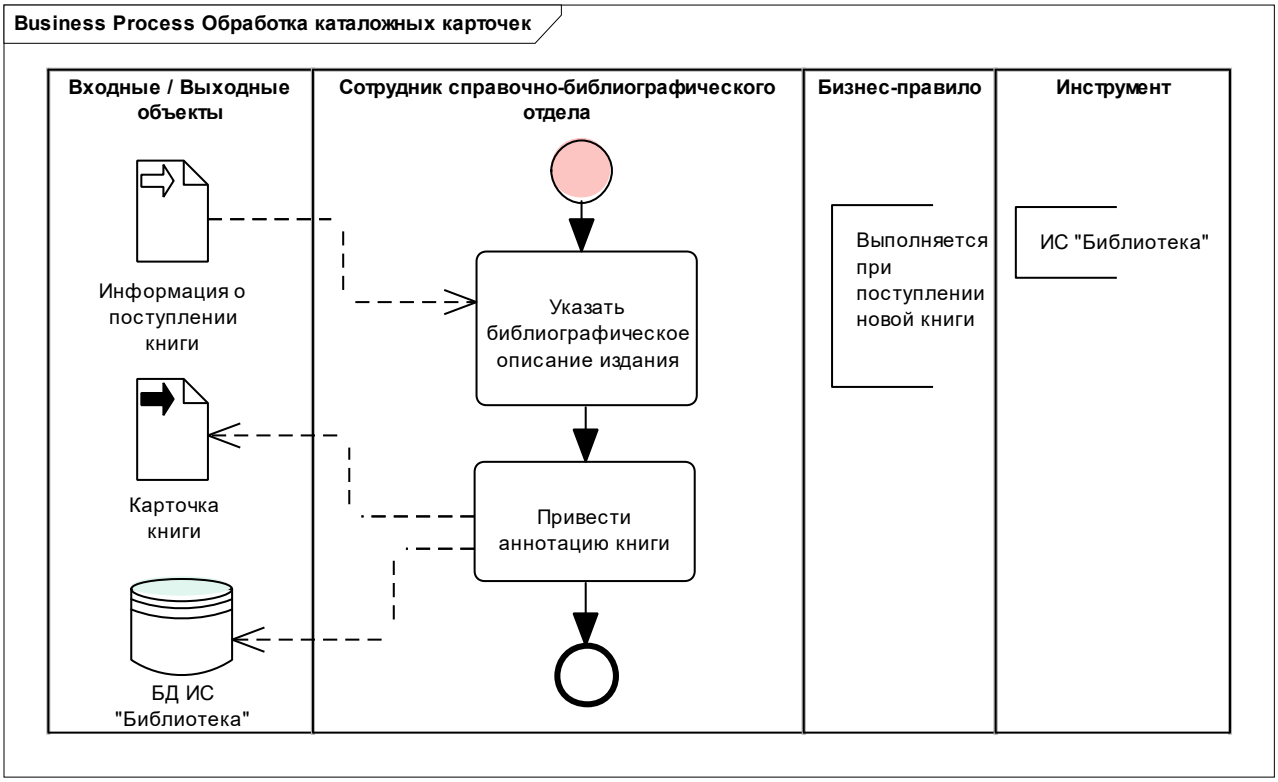


Рисунок 2.3 – Диаграмма декомпозиции подпроцесса «Обработка каталожных карточек» (третий уровень)

Задание

Разработать функциональную модель информационной системы для выбранной предметной области с использованием Enterprise Architect.

Модель должна содержать не менее трех уровней декомпозиции, на последнем уровне должно быть не менее 15 блоков действий (минимум).

Содержание отчета: тема и цель работы; диаграммы декомпозиции.

Контрольные вопросы

1 Перечислить основные этапы моделирования с использованием методологии BPMN.

2 Охарактеризовать элементы нотации BPMN.

3 Чем различаются модели системы AS-IS, TO-BE и SHOULD-BE?

3 Основы использования CASE-средств концептуального проектирования информационной модели системы

Цель: изучить основы применения нотации UML при проектировании баз данных; разработать информационную модель БД ИС.

Теоретические положения

На сегодняшний день существует большое количество инструментов для разработки схем баз данных, например, AllFusion ERwin Data Modeler, DbSchema (https://dbschema.com), DbDiagram.io (https://dbdiagram.io) и т. д. В данной лабораторной работе разработка схемы базы данных будет вестись с использованием CASE-средства Sparx Systems Enterprise Architect [10, с. 56–92, 286–293]. Разработка схемы базы данных в Enterprise Architect может выполняться с использованием нотаций (нотация – система условных графических обозначений и правил их использования для описания различных категорий моделируемой предметной области) UML, IDEF1X, Information Engineering (IE).

При создании новой модели необходимо выбрать File → New project, задать имя файла и тип проекта Enterprise Architect Project (*.eap).

Далее в окне Model Wizard в разделе Database следует выбрать тип добавляемой модели: Data Model – SQLServer.

В браузере проекта появится папка Model, содержащая пакет Data Model, предназначенный для хранения перечня разрабатываемых объектов базы данных, распределенных по двум следующим пакетам:

1) пакет Logical Model – содержит логическую модель БД, состоящую из таблиц, атрибутов и ограничений на их значения, и связей между таблицами;

2) пакет «Database» SQLServer – содержит процедурную часть проектируемой БД, которая в различных СУБД состоит из различных компонентов и реализуется по-разному. Для MS SQL Server данный пакет может содержать хранимые процедуры, функции, запросы, представления, последовательности, триггеры, таблицы (разработанные с помощью Database Builder из папки Tools). Для MS Access пакет может содержать запросы, представления и таблицы.

В папке Model могут храниться пакеты Data Model для различных СУБД, которые можно добавить, нажав правой клавишей мыши по папке Model и выбрав из контекстного меню Add → Add a Model using Wisard.

Добавить сразу несколько пакетов Data Model для различных СУБД можно непосредственно при создании проекта Enterprise Architect Project, отметив галочкой требуемые целевые СУБД. Для выполнения последующих лабораторных работ можно сразу отметить галочками Data Model – MSAccess и Data Model – SQLServer. Сформированные пакеты Data Model – MSAccess и Data Model – SQLServer появятся в браузере проекта.

Далее в пакете Logical Model на диаграмме Logical Model с помощью инструмента Table меню Diagram / Toolbox нужно создать таблицы базы данных и установить для нее целевую СУБД (Access или SQLServer2012).

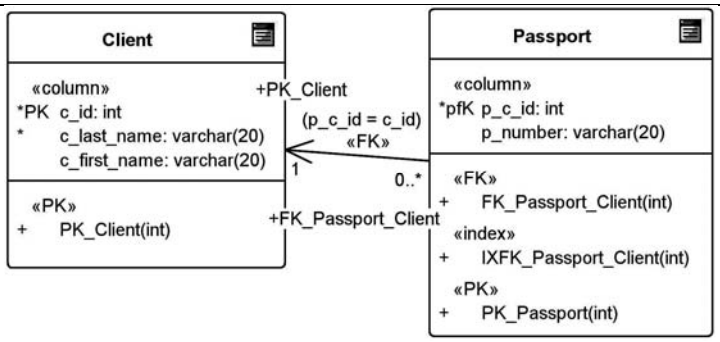
Звездочка слева от названия столбца на схеме таблицы означает, что для такого столбца задано ограничение NOT NULL .

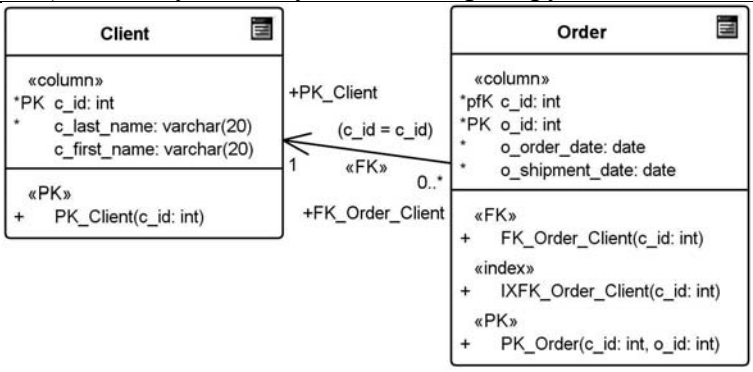
Изменить цвет таблиц схемы базы данных со Standard на белый можно, выбрав Diagram → Appearance → White Board.

Изменить графическую нотацию можно, выбрав в пункте меню Diagram → Properties. Появится окно Data Modeling Diagram: Logical Model, в котором нужно выбрать раздел Connectors и указать выбранный тип Connector Notation: UML 2.1, Information Engineering или IDEF1X.

Связи. В Enterprise Architect на панели инструментов для создания связей между отношениями имеется только один инструмент – ассоциация. **Ассоциация** – это самый общий тип связи, который показывает, что некие сущности взаимосвязаны, при этом тип и особенности этой взаимосвязи не отражаются, хотя некоторые параметры можно указать (направленность связи, мощность связи). Назначение и построение связей основных типов показано в таблице 3.1.

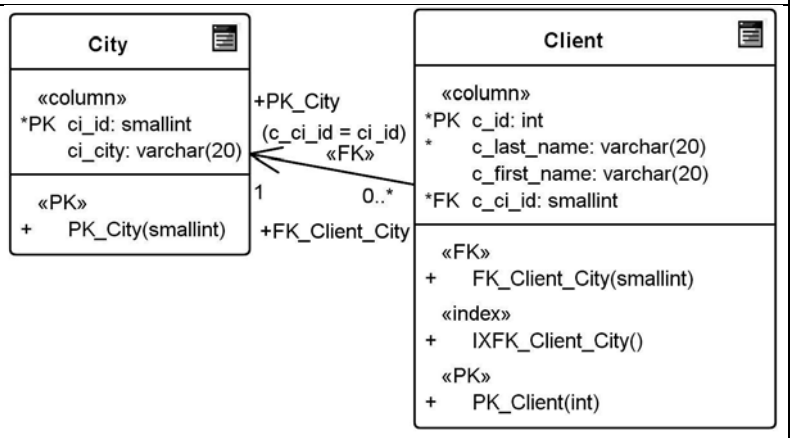
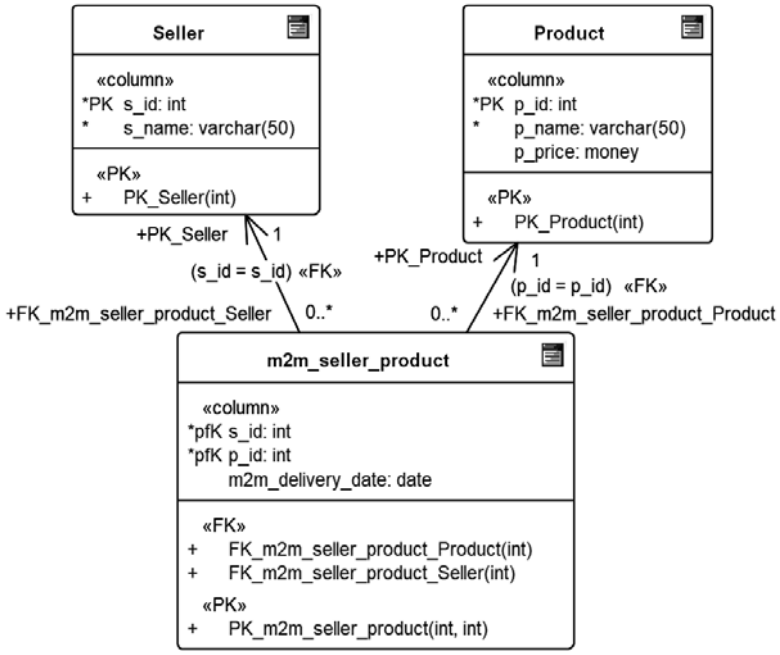
Таблица 3.1 – Создание основных типов связей между таблицами

<table>
<tr><td>Наименование и определение типа связи</td><td>Графическое обозначение типа связи</td></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td rowspan="2">Идентифицирующая – связь между двумя сущностями, в которой каждый экземпляр подчиненной (дочерней) сущности идентифицируется (однозначно определяется) значениями атрибутов родительской сущности. атрибуты первичного ключа родительской сущности мигрируют в состав первичного ключа дочерней сущности и помечаются там как внешний ключ (pfK). При генерации скрипта БД атрибутам внешнего ключа присваивается признак NOT NULL, что означает невозможность внесения записи в дочернюю таблицу без наличия идентификационной информации из родительской таблицы.
Отличие идентифицирующей связи 1:1 от 1:M состоит в числах, указывающих мощность связи, и в том, что первичный ключ в дочерней таблице одновременно является и внешним</td><td> 

Идентифицирующая связь 1:1: например, у каждого клиента может быть только один паспорт, и у каждого паспорта может быть только один владелец. Каждый паспорт может быть определён только через связь с данным клиентом (и не имеет смысла без конкретного клиента), поэтому связь будет идентифицирующей</td></tr>
<tr><td> 

Идентифицирующая связь 1:M: например, каждый клиент может иметь много заказов, и если по бизнес-правилам каждый заказ обязан быть строго привязан к соответствующему клиенту и не имеет смысла без конкретного клиента, связь будет идентифицирующей</td></tr>
</table>

Окончание таблицы 3.1

1	2
Неидентифицирующей называется связь между двумя сущностями, в которой каждый экземпляр подчиненной сущности не зависит от значений атрибутов родительской сущности и может существовать без экземпляра родительской сущности. Поэтому внешний ключ FK будет среди неключевых атрибутов. **Неидентифицирующая** связь называется **обязательной** (No Nulls), если все экземпляры дочерней сущности должны участвовать в связи. При генерации кода БД атрибут внешнего ключа получит признак Not Null. **Неидентифицирующая** связь называется **необязательной** (Nulls Allowed), если некоторые экземпляры дочерней сущности могут не участвовать в связи. В этом случае внешний ключ дочерней сущности может принимать значение NULL	 Сущности Client и City могут идентифицироваться независимо друг от друга, поэтому между ними имеется *неидентифицирующая* связь. Если существует бизнес-правило, в соответствии с которым клиент в обязательном порядке должен соотноситься с городом, то связь между отношениями Client и City будет *обязательной* (столбец внешнего ключа в подчиненной таблице Client c_ci_id помечен звездочкой). Так как один клиент относится к одному городу, а из одного города может быть множество клиентов, то данная связь будет иметь тип 1:M (1 : 1..*)
Связь M:M – ассоциация, связывающая два отношения таким образом, что одному кортежу любого из связанных отношений может соответствовать произвольное количество кортежей второго отношения. Связь M:M может существовать только на даталогическом уровне проектирования. На физическом уровне проектирования, поскольку СУБД не поддерживают связь M:M, такая связь организуется с помощью дополнительной ассоциативной сущности и двух идентифицирующих связей 1:M	 Ассоциативная сущность m2m_seller_product отражает свойства связи M:M. Атрибут «дата поставки товара» не является ни свойством продавца, ни свойством товара, поэтому столбец m2m_delivery_date размещен в ассоциативной сущности

Обеспечение ссылочной целостности базы данных. **Ссылочная целостность** (referential integrity) – свойство реляционной базы данных, заключающееся в том, что для каждого значения внешнего ключа дочернего отношения должно существовать соответствующее значение первичного ключа в родительском отношении (т. е. запись в дочернем отношении не может ссылаться на несуществующую запись в родительском отношении). Либо значение внешнего ключа должно быть неопределенным (т. е. не ссылаться на кортеж родительского отношения) [10, с. 71–77].

Ограничения ссылочной целостности – это правила, которые ограничивают выполнение операций вставки (INSERT), обновления (UPDATE) и удаления (DELETE) экземпляров родительской и дочерней сущностей [3, 6, 7, 10].

Выбор стратегии обеспечения ссылочной целостности определяется бизнес-правилами, действующими в моделируемой предметной области, и типом операции: INSERT, UPDATE или DELETE.

В общем случае (для большинства CASE-средств и СУБД) возможны следующие стратегии обеспечения ссылочной целостности:

1) C – Cascade – разрешить выполнение требуемой операции в **родительском** отношении (затрагивающей первичный ключ), но внести при этом необходимые поправки в дочернем отношении так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Например, при удалении кортежа родительского отношения каскадом удалять все ссылающиеся на него кортежи дочернего отношения. Или при обновлении значения первичного ключа в родительской таблице обновить соответствующее значение внешнего ключа в дочерней таблице. Вставка данных в родительскую или дочернюю таблицу (или выборка данных из любых таблиц) не активирует каскадную операцию. Изменение в родительской таблице полей, не входящих в состав первичного ключа, не активирует каскадную операцию. На одной связи не может быть разрешено несколько различных каскадных операций (например, одновременно и каскадное удаление, и установка значений по умолчанию), кроме каскадного обновления, которое может совмещаться со всеми остальными операциями;

2) Set Null (SN) – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на неопределенные (NULL) значения (установка пустых внешних ключей). Применяется только в случае, если NULL-значения в соответствующем внешнем ключе разрешены;

3) Set Default (SD) (установить по умолчанию) – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию при создании столбца внешнего ключа или вычисляемое при выполнении данной операции;

4) Restrict (R) – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности (например, запретить удаление кортежей в родительском отношении при наличии хотя бы одного ссылающегося кортежа);

5) No Action (NOA) – значение по умолчанию, означает, что никакие действия (UPDATE, DELETE) по модификации строк в родительской таблице не должны быть выполнены при наличии связанных строк в дочерней таблице. Позволяет изменять или удалять только те значения в родительской таблице,

с которыми не связаны соответствующие значения в столбце внешнего ключа дочерней таблицы;

6) NONE (условного обозначения нет) – не требуется специальное определение ссылочной целостности.

Пример разработки информационной модели. В информационной модели библиотеки используются следующие сущности:

– List_of_events – для хранения информации о мероприятиях, проводимых справочно-библиографическим отделом: номер события, дата, описание;

– Department – для хранения информации об отделах библиотеки: номер отдела, название отдела;

– Library_employee – для хранения данных о сотрудниках библиотеки: табельный номер, фамилия, имя, отчество, дата рождения, должность;

– Student – для хранения информации о студентах, которые пользуются библиотекой: номер читательского билета, фамилия, имя, отчество, год поступления, год окончания, факультет, группа, форма обучения;

– Copy_of_book – для хранения информации об экземплярах книг, зарегистрированных в библиотеке: шифр книги, отметка о списании, отметка о замене;

– Book – для хранения информации о книгах: ISBN, фамилия автора, название, предметная область, год издания, издательство, число страниц, цена.

Информационная модель БД ИС отображена на рисунке 3.1.

Связь между сущностями List_of_events и Library_employee неидентифицирующая необязательная, т. к. эти сущности могут существовать независимо друг от друга и за определенным мероприятием необязательно может быть закреплен сотрудник. Тип связи 1:M, т. к. за проведение одного мероприятия могут отвечать несколько сотрудников.

Связь между сущностями Department и Library_employee неидентифицирующая обязательная, т. к. каждый сотрудник закреплен за определенным отделом. Тип связи 1:M, т. к. в одном отделе могут работать много сотрудников.

Связь между сущностями Student и Copy_of_book – M:M, т. к. один студент может пользоваться многими экземплярами, а один экземпляр может быть у многих студентов.

Связь между сущностями Library_employee и Copy_of_book – M:M, т. к. один сотрудник библиотеки может пользоваться многими экземплярами, а один экземпляр может быть у многих сотрудников библиотеки.

Для разрешения связей M:M были введены дополнительные зависимые сущности Issuing_books, Use_of_libr_student.

Ссылочная целостность реализована с учетом следующих ограничений предметной области:

– при изменении информации о каком-либо отделе из таблицы Department в таблице Copy_of_book информация будет автоматически меняться (каскадное обновление), удалять запрещено;

– в таблице Book разрешено изменение записей (каскадное обновление), удаление данных из этой таблицы запрещается (запрет удаления);

– в таблице Copy_of_book разрешено изменение записей (каскадное обновление), удаление данных из этой таблицы запрещается (запрет удаления);

– в таблице Student при изменении информации о студенте происходит каскадное обновление данных. Разрешается удаление информации только в случае, когда на данную информацию нет ссылок в других связанных таблицах.

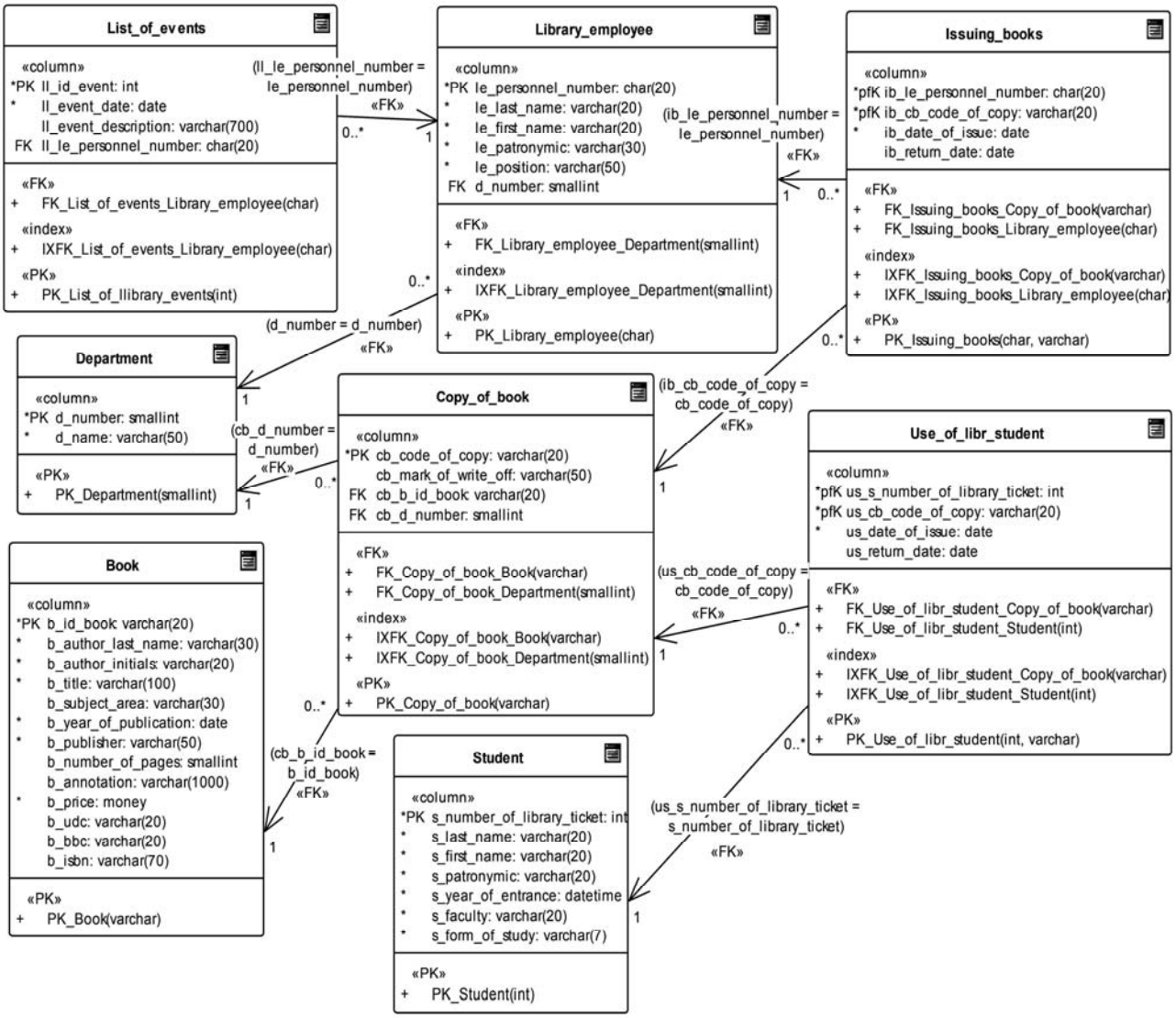


Рисунок 3.1 – Информационная модель

Задание

Разработать информационную модель выбранной предметной области. Информационная модель должна быть представлена схемой БД, разработанной с использованием UML. На схеме БД должны быть отображены сущности (не менее десяти), связи между сущностями (в том числе не менее одной связи M:M), атрибуты, типы данных атрибутов, NULL-значения атрибутов, мощность связей.

Содержание отчета: тема и цель работы; информационная модель; перечень всех сущностей, входящих в модель, с описанием их назначения и указанием атрибутов каждой сущности; подробное обоснование выбранных типов связей (идентифицирующих и неидентифицирующих (обязательных и необязательных)) между сущностями, принятых ограничений ссылочной целостности.

Контрольные вопросы

1 Каковы особенности применения нотации UML при проектировании БД?

2 Охарактеризовать основные понятия модели «сущность – связь»: сущности, атрибуты, виды ключей (первичный, составной, естественный, суррогатный, альтернативный, внешний), идентифицирующие и неидентифицирующие (обязательные и необязательные) связи, направленность и мощность связи, связь категоризации, связь многие-ко-многим и ее разрешение.

3 Как можно выбрать тип данных атрибута?

4 Что такое ссылочная целостность?

4 Взаимодействие CASE-средств информационного моделирования с системами управления базами данных (генерация схемы базы данных)

Цель: получить навыки генерации объектов базы данных из Enterprise Architect в СУБД MS Access; освоить процедуру обратного проектирования.

Теоретические положения

Процедура прямого проектирования (forward engineering) структуры физической БД предусматривает разработку объектов БД в Enterprise Architect и генерацию SQL-скрипта, на основе которого в СУБД, выбранной при проектировании объектов БД, можно воссоздать все спроектированные объекты БД.

Для генерации SQL-скрипта для создания схемы базы данных на основе модели в Enterprise Architect необходимо выбрать пункт меню «Package» → «Database Engineering → Generate Package DDL…». Откроется диалоговое окно «Generate DDL», в котором можно установить ряд параметров. Опции генерации кода можно просмотреть на вкладке «Options». Выбирается способ записи в один файл «Single File» и указывается к нему путь. После этого следует нажать кнопку «Generate» для автоматической генерации SQL-скрипта. Полученный файл можно просмотреть, например, с помощью программы «Блокнот» и использовать в СУБД MS MS Access для автоматического создания схемы БД. При этом нужно учесть, что Access в запросах SQL на создание объектов БД позволяет выполнять только по одной команде создания какого-либо объекта. Для создания таблицы БД в Access нужно открыть вкладку Создание → Конструктор запросов, переключиться из режима конструктора в режим SQL и на появившейся вкладке создания нового запроса выполнить команду CREATE TABLE столько раз, сколько таблиц содержится в разработанной в Enterprise Architect базе данных.

Процедура обратного проектирования (reverse engineering) предусматривает генерацию из функционирующей физической БД соответствующей ей модели данных в Enterprise Architect. Для генерации модели данных нужно в Enterprise Architect создать новый проект New Project, указать его название и путь доступа,

в появившемся окне Model Wizard выбрать Database → Data Model → MSAccess2007. В появившейся модели Data Model – MSAccess выполнить двойной клик по таблице объектов «Database» MSAccess2007 и в появившемся окне MSAccess2007 выбрать из контекстного меню Import DB schema from ODBC. Далее в окне Import DB schema from ODBC source следует выбрать присоединяемую базу данных и путь к ней, источник данных и драйвер MSAccess mdb, *accdb и выполнить импорт базы данных в модель Enterprise Architect.

Задание
Необходимо сгенерировать схему БД из Enterprise Architect в MS Access.

Содержание отчета: тема и цель работы; схема данных БД в MS Access.

Контрольные вопросы

1 Чем различаются понятия «forward engineering» и «reverse engineering»?
2 Что такое «Database Compare»?

5 Синхронизация функциональной и информационной моделей системы

Цель: выполнить синхронизацию функциональной и информационной моделей автоматизированной ИС.

Теоретические положения

После разработки информационной модели ее следует связать с функциональной моделью. Такая связь гарантирует завершенность анализа, обеспечивает наличие источников данных (сущностей) для всех процессов и действий.

На данном этапе производится сопоставление действий в модели BPMN с объектами модели данных (сущностями и атрибутами). При этом нужно учитывать, что действие в функциональной модели процессов может быть связано с несколькими атрибутами различных сущностей. Выполненное сопоставление функциональной и информационной моделей оформляется отчетом о верификации моделей (пример которого представлен таблицей 5.1).

Таблица 5.1 – Пример отчета о верификации моделей

Имя действия	Имя сущности	Имя атрибута
Провести занятия со студентами	List_of_events	ll_id_event ll_event_date ll_event_description ll_le_personnel_number

Отчет о верификации моделей позволяет удостоверить, что в функциональной модели отсутствуют действия, не нашедшие свое отражение в базе данных, а в информационной модели нет сущностей и атрибутов, не связанных ни с какими действиями.

Задание

Необходимо составить таблицу отчета о верификации моделей, в которой должны быть показаны связи всех действий в модели BPMN со всеми сущностями и атрибутами информационной модели. Следует учитывать, что одному и тому же действию в функциональной модели могут соответствовать несколько сущностей в информационной модели и, наоборот, одной сущности могут соответствовать несколько действий.

Содержание отчета: тема и цель работы; таблица отчета о верификации.

Контрольные вопросы

1 Для чего необходимо связывание моделей?
2 Как создать отчет о связывании? Какие поля можно отразить в отчете?

6 Access. Создание и заполнение таблиц

Цель: приобрести навыки работы в СУБД MS Access по созданию таблиц и их заполнению.

Теоретические положения

При разработке структуры таблицы необходимо определить названия ее полей и их типы данных. Каждому полю таблицы присваивается уникальное имя (не более 64 символов). Значение типа поля может быть задано только в режиме конструктора.

В Access существует несколько способов создания пустой таблицы [8]:

– ввод данных непосредственно в пустую таблицу в **режиме таблицы**. При сохранении таблицы Access проанализирует данные и автоматически присвоит каждому полю соответствующий тип данных и формат;

– определение всех параметров макета таблицы в **режиме конструктора**, который позволяет добавлять поля, настроить отображение полей и обработку в них данных, а затем создать первичный ключ.

Схема данных определяет, с помощью каких полей таблицы связываются между собой, как будет выполняться объединение данных этих таблиц, нужно ли проверять связную целостность при добавлении и удалении записей, изменении ключей таблиц.

Типы данных Access приведены в таблице 6.1.

Таблица 6.1 – Типы данных, используемые в Access

Тип данных	Назначение	Размер
Короткий текст (Short Text)	Для хранения текста или комбинаций знаков, не используемых в расчетах	До 255 символов
Длинный текст (Long Text) (ранее Поле MEMO (Memo))	Для хранения обычного текста или комбинаций алфавитно-цифровых знаков длиной более 255 знаков. Поддерживает форматирование текста	1 Гбайт. В элементе управления отображаются первые 64000 символов
Числовой (Number)	Для хранения числовых значений (целых или дробных), которые используются в вычислениях (кроме денежных значений)	1, 2, 4, 8 или 12 байтов (16 байтов, если это поле GUID)
Дата/время (Date/Time)	Для хранения значений даты и времени. Целая часть значения, расположенная слева от десятичной запятой, представляет собой дату. Дробная часть, расположенная справа от десятичной запятой, – это время	8-байтовые числа двойной точности с плавающей запятой
Денежный (Currency)	Для хранения денежных значений (когда значения не должны округляться при вычислениях) с числом знаков до 15 слева от десятичной запятой и с точностью до четырех знаков после десятичной запятой	8 байтов
Логический	Для логических значений: Да/Нет, Истина/Ложь или Вкл/Выкл	1 бит
Счетчик (AutoNumber)	Для формирования уникальных значений	4 байта (16 байтов, если это поле GUID)
Поле объекта OLE	Для хранения объектов OLE (изображений, диаграмм) из других программ Microsoft Windows. Вместо поля объекта OLE лучше использовать поле с типом данных Вложение (Attachment), которое позволяет вкладывать в одну запись несколько файлов и поддерживает больше типов файлов	До 1 Гбайт
Гиперссылка (Hyperlink)	Для хранения ссылок на веб-узлы (URL-адреса), на узлы или файлы интрасети, или локальной сети (UNC-адреса – записи пути стандартного формата), на узлы или файлы локального компьютера, а также на объекты Access, хранящиеся в базе данных	Может содержать до 8192 знаков (каждая часть гиперссылки до 2048 знаков)
Вложение (Attachment)	Для вложения в поле записи файлов изображений, электронных таблиц, документов	Для сжатых вложений – 2 Гбайт
Вычисляемый (Calculated)	Для создания вычисляемых полей: числовых, денежных, дата/время, логических	
Мастер подстановок (Lookup Wizard)	Для запуска мастера подстановок, позволяющего создавать поле, в котором в виде раскрывающегося списка отображаются значения из другой таблицы, запроса или списка значений	При присоединении к полю таблицы или запроса – это размер присоединенного столбца

Для каждого типа данных можно установить значения свойств, которые будут определять особенности обработки соответствующего поля. Среди всех

свойств можно выделить общие, которые можно задать для большинства типов (таблица 6.2).

Таблица 6.2 – Основные свойства большинства типов данных

Свойство	Описание
Маска ввода	Символы редактирования, определяющие способы ввода данных
Подпись	Устанавливается информативное название поля, которое автоматически будет использоваться при создании форм и отчетов
Значение по умолчанию	Значение, которое будет использоваться по умолчанию, т. е. в том случае, если в данное поле не будет введена информация
Условие на значение	Устанавливает ограничение на вводимые данные, т. е. не позволяет вводить в поле данные, не соответствующие указанному условию
Сообщение об ошибке	Задает текст сообщения, который будет отображаться в том случае, если данные, введенные в поле, не соответствуют ограничению, указанному в свойстве Условие на значение
Обязательное поле	Определяет режим обязательного ввода информации в данное поле
Индексированное поле	Устанавливает режим использования индекса для данного поля, что позволяет ускорить доступ к информации в поле, а также задать режим, при котором в поле нельзя вводить повторяющиеся значения

Структура базы данных задается с помощью схемы данных. В ней определяются и запоминаются связи между таблицами. Схема данных открывается командой «Работа с базами данных» → «Схема данных». При этом появляется окно «Добавление таблицы», с помощью которого на схему добавляются таблицы базы данных. Между таблицами устанавливаются связи 1:1 и 1:М. При нажатии левой клавишей мыши по связи из контекстного меню выбирается «Изменить связь», и в появившемся диалоговом окне «Связи» нужно установить флажок «Обеспечение целостности данных».

При построении связи Access называет одну из таблиц главной, а другую – связанной. Если одно из связываемых полей является ключевым или просто имеет уникальный индекс, главной будет таблица, содержащая это поле. Или главной считается таблица, с которой было начато прокладывание связи.

Создание связи «один-к-одному». Оба общих поля (как правило, поля первичного ключа и внешнего ключа) должны иметь уникальный индекс. Это означает, что свойство «Индексированное поле» должно иметь для этих полей значение «Да (Совпадения не допускаются)».

Создание связи «один-ко-многим». Поле на стороне «один» связи (как правило, это первичный ключ) должно иметь уникальный индекс. Это означает, что свойство «Индексированное поле» этого поля должно иметь значение «Да (Совпадения не допускаются)». Полю на стороне «многие» не должен соответствовать уникальный индекс. У него может быть индекс, однако он должен допускать совпадения. Это означает, что его свойство «Индексированное поле» может иметь значение «Нет» либо «Да (Допускаются совпадения)».

При включении обеспечения целостности данных должны выполняться следующие условия:

– общее поле главной таблицы должно быть первичным ключом или иметь уникальный индекс (общие поля могут иметь разные имена);

– общие поля должны иметь одинаковый тип данных. Единственное исключение – поле типа «Счетчик» можно связать с полем типа «Числовой», если свойство «Размер поля» обоих полей одинаково (например, «Длинное целое»).

Задание

Для сгенерированной из Enterprise Architect базы данных нужно создать схему данных, а также заполнить базу 200 записями (в сумме для всех таблиц).

Содержание отчета: тема и цель работы; схема данных, скриншоты таблиц, заполненных данными.

Контрольные вопросы

1 Перечислите все типы данных для полей в Access.

2 Для чего используется тип данных Поле объекта OLE? Как разместить объект OLE?

3 Как отменить ввод записи? Как ввести данные в таблицу?

7 Access. Создание запросов

Цель: приобрести навыки работы в СУБД Access по построению запросов.

Теоретические положения

Запросы создаются для выборки данных из одной или нескольких связанных таблиц по заданным условиям, для проведения вычислений и статистической обработки данных [8]. С помощью запроса можно обновить, удалить, добавить данные в таблицу, создать новые таблицы (таблица 7.1).

При выполнении запроса на выборку Access извлекает записи из таблиц и формирует результирующий набор данных – динамический (или виртуальный) набор записей, не хранящийся в базе данных, т. е. прекращающий свое существование после закрытия запроса. Сохраняется только структура запроса – перечень таблиц, список полей, порядок сортировки, тип запроса и т. д.

Запросы в СУБД Access можно создавать с помощью: режима мастера; режима конструктора, являющегося графическим инструментом языка QBE (Query-by-Example – язык запросов по образцу); языка SQL (используется диалект Jet SQL).

Таблица 7.1 – Виды запросов в Access

Наименование	Назначение	Способ реализации
1	2	3
Запрос на выборку	Позволяет отобразить записи из одной или нескольких таблиц по указанным полям, сгруппировать записи для вычисления сумм, средних значений и т. д.	Вкладка «Создание» → Кнопка «Конструктор запросов». Для шаблонов в поиске используются следующие символы: ? или _ заменяет текстовый символ # – любую одиночную цифру (0-9) * или % – любое количество символов [а-л] – символы в заданном интервале [!м-ю] – символы вне заданного интервала. Например, условие Like "М*" выбирает записи со значениями поля, которые начинаются на букву М
Запрос на выборку с параметром	При выполнении выводит приглашение для ввода данных	В строку «Условие отбора» вводят имя параметра в квадратных скобках, отличающееся от имени поля
Перекрестный запрос	Представляет результаты в виде сводной таблицы с группировкой по строкам и столбцам и вычислениями по заданной функции (Sum, Min, Max, Avg, Count и т. д.). Значения группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй – в верхней строке	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить нужные таблицы или запросы и задать условия отбора → Изменить тип запроса на «Перекрестный». Для полей, значения которых группируются по строкам, в строке «Перекрестная таблица» выбрать «Заголовки строк». Для поля, значения которого представлены в запросе как заголовки столбцов, в строке «Перекрестная таблица» выбрать «Заголовки столбцов». Для поля, по которому производятся вычисления, в строке «Перекрестная таблица» выбрать «Значения», а в строке «Групповая операция» – нужную функцию. Для полей, содержащих условие, но без группировки, в строке «Групповая операция» выбирается «Условие», а строка «Перекрестная таблица» оставляется пустой
Запрос на выборку с групповыми операциями	Позволяет выделить группы записей с одинаковыми значениями в указанных полях группировки и выполнить статистические вычисления по заданной функции (Sum, Min, Max, Avg, Count (подсчет количества непустых значений поля в группе)). Результат содержит по одной записи для каждой группы	В бланк запроса включают поля, по которым производится группировка, и поля, для которых выполняются групповые функции. При нажатии кнопки «Итоги» на вкладке «Конструктор» в бланке появится строка «Групповая операция», в которой для вычисляемых полей вместо «Группировка» указывают функцию
Запрос Записи без подчиненных	Предназначен для поиска записей основной таблицы, у которых нет связанных записей	Вкладка «Создание» → Кнопка «Мастер запросов». Указываются анализируемая (главная) таблица, подчиненная таблица и поля связи

Окончание таблицы 7.1

1	2	3
Запрос на обновление	Используется для обновления значений указанных полей таблицы новыми значениями	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить обновляемую таблицу → Изменить тип запроса на «Обновление». Для обновляемого поля в строке «Обновление» вводится выражение, вычисляющее значение
Запрос на добавление	Предназначен для вставки записей из одной или нескольких таблиц в одну целевую таблицу	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на добавление
Запрос на создание таблицы	Служит для создания новой таблицы на основе записей существующих таблиц	Строится как запрос на выборку, а затем тип запроса меняется на «Создание таблицы» и в появившемся диалоговом окне задается имя новой таблицы
Запрос на удаление	Предназначен для удаления записей, удовлетворяющих заданному условию	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить требуемую таблицу → Изменить тип запроса на «Удаление». В первом столбце в первом поле бланка ставится «ИмяПоля.*» (т. е. «все поля»), после чего в строке «Удаление» будет выведен текст «Из». В следующих столбцах выбирают поля, для которых задают условия отбора

Задание

Для спроектированной базы данных необходимо разработать 15 запросов, представляющих изученные виды запросов (с учетом запросов в техническом задании, разработанном в лабораторной работе № 1).

Содержание отчета: тема и цель работы; SQL-код запросов и скриншоты результатов выполнения запросов.

Контрольные вопросы

1 Что такое запрос? Какие способы создания запросов существуют?

2 Как при создании запроса установить условия отбора? Как осуществить поиск данных в диапазоне значений? Как используются выражения в запросе? Как сортируются данные в запросе? Как установить параметры в запросе? Как суммировать данные в запросе?

3 Каково назначение запросов с групповыми операциями и перекрестных запросов? Какие функции используются в данных запросах?

4 Логические операторы и их назначение. Назначение операторов BETWEEN, IN, LIKE.

5 Встроенные функции даты и времени. Встроенные функции и операторы для работы с текстом. Встроенные функции преобразования типов данных.

6 Как создаются запросы на обновление, добавление, удаление?

8 Access. Создание форм и отчетов

Цель: приобрести навыки работы в Access по созданию форм и отчетов.

Теоретические положения

Формы являются основным средством организации интерфейса пользователя в приложениях Access. Способы создания форм рассматриваются в [8].

Кнопочная форма – это форма, содержащая набор кнопок, направляющих пользователя к другим формам (обычно при щелчке мышью кнопки формы), т. е. своего рода главное меню БД. Кнопки можно связать с какими-либо действиями – открытием другой формы, выполнением запроса, печатью таблицы, выходом из приложения и т. д.

Для автоматического создания кнопочной формы следует выбрать на ленте «Работа с базами данных» → «Диспетчер кнопочных форм» (Database Tools → Switchboard Manager). Если диспетчер кнопочных форм отсутствует, то его надо включить. Для этого следует выбрать пункт меню «Файл», выбрать раздел «Параметры» и в нем – «Панель быстрого доступа». Затем на вкладке «Настройка панели быстрого доступа» выбрать из раскрывающегося списка строку «Вкладка «Работа с базами данных» и в списке команд выделить «Диспетчер кнопочных форм», а дальше нажать на кнопку «Добавить». При первом нажатии этой кнопки Access сообщит, что не может найти кнопочную форму и предложит ее создать. Диспетчер кнопочных форм выводит на экран список страниц. Каждая страница – отдельная часть меню кнопочной формы.

Автоматический запуск главной кнопочной формы при открытии приложения. Предполагается, что форма, которая должна отображаться первой, уже создана. Следует открыть вкладку «Файл» и выбрать «Параметры». В открывшемся окне в левой колонке в группе «Текущая база данных» в списке «Форма просмотра» выбрать форму, отображаемую при запуске БД, и нажать ОК. Чтобы отобразилась начальная форма, нужно закрыть БД и открыть ее повторно.

Макросы – это небольшие программы на языке макрокоманд СУБД Access, состоящие из последовательности определенных команд (одной или нескольких макрокоманд). Макросы являются простейшими средствами автоматизации действий над объектами Access. Если нажать правой кнопкой мыши по полю кнопки и выбрать **Обработка событий → Макросы**, **о**ткроется окно конструктора макросов, в котором надо выбрать макрокоманду из выпадающего списка (всего имеется около 50 различных макрокоманд). Все созданные макросы будут отображаться во вкладке **Макросы.**

Отчеты используются для отображения информации, содержащейся в таблицах, в отформатированном виде, который легко читается как на экране компьютера, так и на бумаге. Отчет можно создать в режиме конструктора, в режиме мастера и в режиме макета. В ходе создания отчета простыми средствами перетаскивания в отчет нужных полей из таблиц БД конструируется запрос – источник записей отчета. Использование свойств WYSIWYG позволяет сразу увидеть,

как именно будут выглядеть данные на странице отчета, и усовершенствовать макет [12]. Разработка и форматирование отчета аналогичны форме.

Задание

Для созданной базы данных необходимо для всех таблиц разработать формы, оформить главную кнопочную форму, реализовать все отчеты, указанные в техническом задании в лабораторной работе № 1.

Содержание отчета: тема и цель работы; скриншоты форм и отчетов.

Контрольные вопросы

1 Какие способы создания форм существуют?

2 Что такое главная кнопочная форма и как организовать автоматический запуск главной кнопочной формы при открытии приложения?

3 Для чего служат отчеты и из каких разделов они могут состоять?

4 Какие элементы форматирования используют для оформления отчетов?

5 Как в отчете задаются уровни группировки и сортировка?

6 С какой целью и в какие разделы отчета добавляются вычисляемые поля?

9 Технология создания баз данных на основе промышленной СУБД MS SQL Server

Цель: получить навыки установки Microsoft SQL Server; научиться создавать базу данных на основе скрипта SQL.

Теоретические положения

Microsoft SQL Server – система управления реляционными базами данных (СУБД), использующая язык структурированных запросов Transact-SQL (T-SQL) [1–11]. Установить Microsoft SQL Server Developer Edition можно с официального сайта (https://docs.microsoft.com/ru-ru/sql/tools/).

Для подключения к серверу баз данных и выполнения большинства действий над базой данных (БД) используется среда Microsoft SQL Server Management Studio (SSMS).

Генерация SQL-скрипта для создания схемы базы данных на основе модели в CASE-средстве. В Sparx Enterprise Architect необходимо выбрать пункт меню «Package» → «Database Engineering → Generate Package DDL…». Откроется диалоговое окно «Generate DDL», в котором можно установить ряд параметров. Опции генерации кода можно просмотреть на вкладке «Options». Выбирается способ записи в один файл «Single File» и указывается к нему путь. После этого следует нажать кнопку «Generate» для автоматической генерации SQL-скрипта.

Полученный файл можно просмотреть с помощью программы «Блокнот» и использовать в СУБД MS SQL Server для автоматического создания схемы БД.

БД в SQL Server состоит из двух частей:

1) файл данных – файл, имеющий расширение .mdf, в котором находятся все основные объекты БД (таблицы, индексы, представления и т. д.);

2) файл журнала транзакций – файл, имеющий расширение .ldf, который содержит журнал, где фиксируются все действия с БД (записываются сведения о процессе работы с транзакциями (контроль целостности данных, состояния базы данных до и после выполнения транзакций). Данный файл предназначен для восстановления БД в случае её выхода из строя.

Создание новой базы данных в SSMS. Создать новую базу данных можно с использованием диалоговых средств SSMS [5, с. 93–163].

Для создания файла БД в обозревателе объектов SSMS следует нажать правую клавишу мыши на папке «Databases» (Базы данных) и в контекстном меню выбрать пункт «New Database» или «Создать базу данных». В появившемся окне нужно указать имя БД, определить ее владельца (по умолчанию), задать путь доступа к файлам БД .mdf и .ldf.

Далее на вкладке создания нового запроса нужно выполнить SQL-скрипт, сгенерированный в Sparx Enterprise Architect.

Один экземпляр SSMS может управлять несколькими базами данных. Вся информация о базах данных, управляемых SSMS, хранится в системной базе данных master.

Создать новую базу данных можно также с использованием оператора T-SQL CREATE DATABASE [4, с. 93–104, 116–118].

Удалить базу данных можно с помощью контекстного меню, выбрав пункт *Удалить*, или с использованием оператора T-SQL DROP DATABASE.

Отсоединение и *присоединение* БД используются для переноса БД между различными экземплярами SSMS. Разработанная под управлением сервера S1 БД может быть отсоединена от S1, скопирована на диск другого компьютера и присоединена к серверу S2 на втором компьютере. Для присоединения БД нужно в обозревателе объектов SSMS выбрать «Базы данных» → «Присоединить», для отсоединения БД – выбрать «Базы данных», выделить имя отсоединяемой БД и в контекстном меню выбрать «Задачи» → «Отсоединить».

Создать таблицу в SSMS можно на диаграмме баз данных либо с помощью обозревателя объектов. В обозревателе нужно раскрыть вкладку с созданной БД, раскрыть вкладку «Таблицы» и в появившемся после нажатия правой клавиши мыши контекстном меню выбрать «Создать таблицу». В появившемся окне определения полей новой таблицы указать следующее:

– Column Name – имя поля, начинающееся с буквы и не содержащее различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки-ограничители;

– Data Type – тип данных поля [4, 5, 11];

– Allow NULL – разрешить значения NULL. Если эта опция поля включена, то в случае незаполнения поля в него будет подставлено значение NULL.

После создания таблиц можно перейти к построению схемы БД. Для этого

в обозревателе объектов нужно выбрать «Диаграммы баз данных» и в контекстном меню – «Создать диаграмму базы данных» добавить все таблицы в окно схемы БД. Для определения связей следует «ухватиться» за поле главной таблицы и «перетащить» его на поле подчиненной таблицы. При определении связи появляется окно «Create Relationship», в котором задается название связи в поле «Relationship name».

Задание

Необходимо сгенерировать схему базы данных из Enterprise Architect в MS SQL Server.

Содержание отчета: тема и цель работы; схема БД в MS SQL Server.

Контрольные вопросы

1 Перечислить типы файлов БД в SQL Server и пояснить их назначение.
2 С помощью каких операторов T-SQL создается и удаляется БД?
3 Для чего применяется отсоединение и присоединение БД?
4 Сколько схем БД можно создать? Что такое ограничения целостности БД?

10 Создание sql-скрипта заполнения базы данных

Цель: получить навыки создания sql-скриптов заполнения БД.

Теоретические положения

Сценарий (скрипт) – последовательность операторов T-SQL. Для подготовки сценария, отладки и выполнения используется SSMS.

Автоматическая генерация скриптов из обозревателя объектов.

Для таблицы можно сгенерировать скрипты для создания таблицы, удаления таблицы, выборки данных из таблицы, скрипты для добавления новых данных в таблицу, а также для изменения и удаления существующих записей.

Для генерации скрипта таблицы текущей БД из контекстного меню выбирается команда «Создать сценарий для таблицы» → «Используя CREATE».

Для генерации скрипта БД после выделения имени БД из контекстного меню выбирается команда «Задачи» → «Сформировать скрипты» → «Создать скрипт для всей базы данных и всех ее объектов» → «Указание порядка сохранения скриптов: Открыть в новом окне запроса» → «Дополнительные параметры создания скрипта: Типы данных для внесения в скрипт – Схема и данные».

При запуске SSMS и подключении к SQL-серверу по умолчанию выполняется команда USE master, т. е. работа идет с системной БД master. Для выбора иной БД следует выполнить команду USE Имя_Базы_Данных. Перед запуском на исполнение сгенерированного скрипта БД аналогичным образом в первой строке скрипта нужно указывать USE Имя_Базы_Данных.

Для создания скрипта для административных операций (например, резервное копирование базы данных, создание учетной записи и т. д.) можно воспользоваться контекстным меню «Задачи» → «Создать резервную копию…» → «Скрипт», что позволит автоматически создать скрипт, в который будут подставлены введенные в полях формы на экране значения.

Задание

Необходимо, используя команды INSERT, внести в базу данных не менее 200 записей. Пример команды INSERT:

INSERT INTO Table1(t_id, product, date) VALUES (3, 'Коробка', 2022-12-26)

Для заполненной БД сгенерировать скрипт, содержащий схему базы данных и данные в таблицах.

Содержание отчета: тема и цель работы; скрипт со схемой БД и данными.

Контрольные вопросы

1 Что такое скрипт (сценарий) и для решения каких задач он используется?
2 Какие виды скриптов существуют?
3 Для чего предназначен оператор GO [5, с. 108–109]?
4 Перечислить основные характеристики базы данных [5, с. 153–162].

11 Язык SQL. Добавление, изменение и удаление данных в таблицах средствами SQL

Цель: научиться добавлять, изменять и удалять данные в таблицах с помощью операторов T-SQL.

Теоретические положения

Для вставки данных в таблицу средствами T-SQL используются операторы INSERT, UPDATE, DELETE, синтаксис которых рассмотрен в [4, 5, 7, 9–12].

В таблице 11.1 приведены примеры **оператора INSERT** для вставки данных в таблицу, SQL-код определения которой приведен ниже:

```
CREATE TABLE Book
(
    b_id_book int PRIMARY KEY IDENTITY (1, 1),
    b_title varchar (100) NULL,
    b_publisher varchar (50) NOT NULL DEFAULT 'ИНФРА-М'
)
```

Таблица 11.1 – Примеры инструкций INSERT

Задача	SQL-код решения
Добавление в таблицу строки с указанием имен столбцов	INSERT INTO Book (b_title, b_publisher) VALUES ('Война и мир', 'BHV');
Добавление в таблицу строки без указания имен столбцов	INSERT INTO Book VALUES ('Война и мир', 'BHV');
Добавление в таблицу строки с подстановкой значения, заданного ограничением DEFAULT	INSERT INTO Book VALUES ('Анна Каренина', DEFAULT);
Добавление в таблицу строки с указанием измененного порядка следования столбцов	INSERT INTO Book (b_publisher, b_title) VALUES ('BHV', 'Война и мир');
Добавление в таблицу строки в том случае, если все столбцы имеют значения по умолчанию (второй столбец в таблице Book тоже имеет значение по умолчанию – NULL)	INSERT INTO Book DEFAULT VALUES;
Добавление в таблицу нескольких строк	INSERT INTO Book VALUES ('Исповедь', DEFAULT), ('Воскресение', DEFAULT);
Добавление в таблицу строки с заданным значением в столбце идентификаторов	SET IDENTITY_INSERT Book ON; INSERT INTO Book (b_id_book, b_title, b_publisher) VALUES (17285, 'Детство', 'BHV')
Добавление строк, значения которых определяются на основе подзапроса	INSERT INTO Copy_Book VALUES ('Исповедь', (SELECT b_publisher FROM Book WHERE b_id_book = 1)), ('Воскресение', (SELECT b_publisher FROM Book WHERE b_id_book = 2));
Добавление в копию таблицы строк, отобранных в подзапросе на основе некоторого условия	INSERT INTO Copy_Book SELECT * FROM Book WHERE b_title = 'Воскресение';
Добавление данных во вновь создаваемую таблицу Book_copy – с помощью инструкции SELECT INTO, которая является вариацией простой инструкции SELECT	SELECT b_id_book, b_publisher, b_title INTO Book_copy FROM Book WHERE b_id_book = 1

Для создания набора вставляемых данных можно использовать инструкцию выполнения вместе с хранимой процедурой или пакетом SQL.

Если столбец имеет свойство IDENTITY (столбец идентификаторов, счетчик), при вставке строки имя этого столбца и значение поля для этого столбца в команде INSERT не указывают. Для такого столбца сервер автоматически вычисляет новое значение. Оператор SET IDENTITY_INSERT < имя таблицы > { ON | OFF } отключает (ON) или включает (OFF) автоинкремент.

В таблице 11.2 приведены примеры использования **оператора UPDATE**.

Операция обновления столбца со свойством IDENTITY представляет собой комбинацию инструкции DELETE с удалением ненужного значения из ячейки столбца идентификаторов и инструкции INSERT со вставкой требуемого значения в ячейку столбца идентификаторов.

Таблица 11.2 – Примеры инструкций UPDATE

Задача	SQL-код решения
Обновление в таблице Book_copy всех значений столбца b_publisher на основе выражения для определения новых значений	UPDATE Book_copy SET b_publisher = CONCAT('издательство ', b_publisher)
Обновление в таблице Book значений столбца с указанием условия отбора строк для обновления	UPDATE Book SET b_publisher = 'AZ' WHERE b_publisher = 'Лира';
Обновление в таблице Book значений нескольких столбцов с указанием условия отбора строк для обновления	UPDATE Book SET b_publisher = 'AZ', b_title = 'Букварь' WHERE b_publisher = 'Лира';
Обновление в столбце b_publisher таблицы Book значения 'ИНФРА-М' на значение 'Y' во всех строках, где b_publisher='ИНФРА-М'	UPDATE Book SET Book.b_publisher = 'Y' FROM (SELECT * FROM Book WHERE b_publisher='ИНФРА-М') AS BS WHERE Book.b_id_book = BS.b_id_book

В таблице 11.3 приведены примеры использования **оператора DELETE**.

Примечание – Во избежание нежелательных последствий с помощью инструкции SELECT INTO создается копия таблицы «Book» и выполняется работа уже с копией:

```
SELECT * INTO Book_copy FROM Book
DELETE FROM Book_copy /* удаление всех строк из таблицы Book_copy */
DROP TABLE Book_copy  /* удаление таблицы Book_copy */
```

Таблица 11.3 – Примеры инструкций DELETE

Задача	SQL-код решения
1	2
Удаление из таблицы Book строк с указанием условия отбора удаляемых строк	DELETE FROM Book_copy WHERE b_publisher LIKE 'A%';
Удаление первых двух строк таблицы Book_copy. Подзапрос (инструкция SELECT в круглых скобках) возвращает базовый набор строк для инструкции DELETE. Результату этого подзапроса присваивается псевдоним bc, а директива WHERE задает параметры сравнения строк из bc с базовой таблицей. Затем директива DELETE автоматически удаляет все совпавшие строки	SELECT * INTO Book_copy FROM Book DELETE Book_copy FROM (SELECT TOP 2 * FROM Book_copy) AS bc WHERE Book_copy.b_id_book = bc.b_id_book
Удаление тех строк из таблицы Book_copy, для которых нет соответствующих строк в таблице Book, с использованием стандартного синтаксиса оператора DELETE, при этом для определения удаляемых строк используется подзапрос	DELETE FROM Book_copy WHERE b_publisher = 'Лира' AND b_id_book NOT IN (SELECT b_id_book FROM Book);

Окончание таблицы 11.3

1	2
Удаление тех строк из таблицы Book_copy, для которых нет соответствующих строк в таблице Book, с использованием дополнительного предложения FROM, которое вводит источник табличного типа, конкретизирующий данные, удаляемые из таблицы в первом предложении FROM	DELETE FROM Book_copy FROM Book_copy AS bc LEFT JOIN Book ON bc.b_id_book = Book.b_id_book WHERE bc.b_publisher = 'Лира' AND Book.b_id_book IS NULL;

Инструкция TRUNCATE TABLE ИмяЦелевойТаблицы удаляет все строки в целевой таблице, не записывая в журнал транзакций удаление отдельных строк, за счет чего выполняется быстрее и требует меньших ресурсов системы.

В таблице 11.4 представлены допустимые подстановочные символы для шаблонов LIKE (см. таблицу 11.3), их значения и примеры использования.

Таблица 11.4 – Подстановочные символы, используемые в шаблонах LIKE

Подстановочный знак	Значение	Пример	Результат
%	Символ-шаблон, заменяющий любую последовательность символов	WHERE [title] LIKE 's%'	Строка, начинающаяся с s: Samantha или sven
_ (подчеркивание)	Символ-шаблон, заменяющий любой одиночный символ	WHERE [titleofcourtesy] LIKE 'm _.'	Ms. Mr.
[<список символов>]	Один символ из списка	WHERE [firstname] LIKE '[sp]%'	Строка, в которой первый символ s или p: Sara или Paul
[<диапазон символов>]	Один символ из диапазона. При этом можно перечислить сразу несколько диапазонов (например, [0-9a-z])	WHERE [freight] LIKE '6[5-7]%'	Строка, в которой второй символ – цифра 5, 6 или 7: 65,83 или 677,54
[^<список или диапазон символов>]	В сочетании с квадратными скобками исключает из поискового образца символы из списка или диапазона	WHERE [shipaddress] LIKE '[^0-9]%'	Строка, в которой первый символ не цифра
ESCAPE ' '	Для поиска символа, который является подстановочным знаком, его указывают после Escape-символа с помощью ключевого слова ESCAPE	WHERE col1 LIKE '!_%' ESCAPE '!'	Поиск строки, которая начинается со знака подчеркивания (_), используя в качестве Escape-символа (!)
'ymd'	Для поиска даты используется форма без разделителей, которая не зависит от языка входа в систему для всех типов данных даты и времени	WHERE [birthdate] = '19581208'	1958-12-08 00:00:00.000

Задание

Необходимо для разрабатываемой БД написать для каждой таблицы по три различных команды INSERT, UPDATE, DELETE с использованием различных функций (не менее 15 различных функций) и предикатов в условии отбора. Перечень функций T-SQL содержится в [4, 5].

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Объяснить синтаксис операторов INSERT, SELECT INTO, UPDATE, DELETE и указать ограничения для данных операторов.

2 Привести примеры использования INSERT, UPDATE, DELETE.

3 Охарактеризовать особенности использования операторов INSERT, UPDATE, DELETE по отношению к столбцу идентификаторов IDENTITY.

4 Указать назначение и ограничения оператора TRUNCATE TABLE.

5 Перечислить подстановочные символы, используемые в шаблонах LIKE.

6 Перечислить основные группы функций T-SQL [4, 5] и охарактеризовать наиболее часто используемые функции.

12 Язык SQL. Работа с представлениями

Цель: научиться создавать представления в SQL Server средствами T-SQL.

Теоретические положения

Представление – это именованный запрос на выборку, сохраненный в БД, который выглядит и работает как таблица, при обращении по имени создает виртуальную таблицу, наполняя ее актуальными данными из БД, с которой можно работать как и с реально существующей на диске таблицей. Физически представление реализовано в виде SQL-запроса, на основе которого производится выборка данных из одной или нескольких таблиц или представлений. Представление часто применяется для ограничения доступа пользователей к конфиденциальным данным в таблице. Синтаксис представлений рассмотрен в [5, 9].

Чтобы выполнить представление, т. е. получить данные в виде виртуальной таблицы, необходимо выполнить запрос SELECT к представлению так же, как и к обычной таблице: SELECT * FROM view_name.

Для удаления представления используется команда T-SQL DROP VIEW{view […n]}. За один раз можно удалить несколько представлений.

В качестве примера приведено представление, предоставляющее информацию об экземплярах книг, которые были изданы за последние пять лет.

```
CREATE VIEW Get_full_info_ copy_of_book
AS
SELECT        /*Указываем, какие поля будут выбраны*/
Copy_of_book.cb_code_of_copy,
Book.b_author_last_name,        Book.b_title,        Book.b_year_of_publication,
Book.b_publisher,
Copy_of_book.cb_d_number,
Copy_of_book.cb_mark_of_write_off, Copy_of_book.cb_mark_of_replacement
FROM          /*Указываем таблицу и связанные с ней таблицы, из которых выбираются связанные данные*/
Book      INNER JOIN Copy_of_book
            ON Book.b_id_book = Copy_of_book.cb_b_id_book
WHERE    YEAR(Book.b_year_of_publication)    BETWEEN    YEAR(GETDATE()-5)    AND    YEAR(GETDATE())
/* GETDATE()  возвращает текущую дату, YEAR(<дата>) – год <даты> */
```

Задание

В разрабатываемой базе данных необходимо реализовать 15 представлений с использованием стандартных функций T-SQL. При этом должно быть использовано не менее 10 различных функций.

Содержание отчета: тема и цель работы; SQL-код 15 представлений.

Контрольные вопросы

1 Что такое представление и в каких случаях целесообразно его использовать? Перечислить способы создания представлений.

2 Какие виды представлений различают? Что такое обновляемые представления? Что такое кэширующие (материализованные, индексированные) представления?

3 Перечислить операторы SQL, с помощью которых представления создаются, удаляются и изменяются.

13 Язык SQL. Создание хранимых процедур

Цель: научиться создавать хранимые процедуры в СУБД MS SQL Server для вставки, удаления, изменения данных с использованием команд T-SQL.

Теоретические положения

Хранимая процедура – это скомпилированный набор SQL-предложений, сохраненный на сервере баз данных как именованный объект и выполняющийся как единый фрагмент кода. Хранимые процедуры могут принимать и возвращать параметры. При этом клиент осуществляет только вызов хранимой процедуры

по ее имени, затем сервер базы данных выполняет блок команд, составляющих тело вызванной процедуры, и возвращает клиенту результат [5, 9]. По сравнению с обычными SQL-запросами, посылаемыми из клиентского приложения, они требуют меньше времени для подготовки к выполнению, поскольку скомпилированы и сохранены. Синтаксис хранимых процедур описан в [4, 5, 9].

Для вызова хранимой процедуры используется оператор EXECUTE (сокращенно EXEC). Здесь задается имя процедуры и список значений передаваемых процедуре параметров. Если параметр является выходным, то он задается в виде имени локальной переменной, за которым идет ключевое слово OUTPUT (OUT).

Далее приведен пример хранимой процедуры, возвращающей количество экземпляров какой-либо книги.

```
/* проверяется, существует ли хранимая процедура Count_number_of_copies, и при необходимости она удаляется */
DROP PROCEDURE IF EXISTS Count_number_of_copies;
GO
/* проверяется, существует ли временная таблица Temp1, и при необходимости она удаляется */
DROP TABLE IF EXISTS TEMP1;
GO

CREATE PROCEDURE Count_number_of_copies
@b_id_book varchar(20)      /*Объявляем входную переменную*/
@number int  OUTPUT         /*Объявляем выходную переменную*/
AS
/* Следующая конструкция проверяет, существуют ли записи в таблице «Book» с заданным b_id_book*/
IF NOT EXISTS (SELECT * FROM Book WHERE b_id_book = @b_id_book)
RETURN 0       /* Вызывает конец процедуры Count_number_of_copies */
SELECT Copy_of_book.cb_b_id_book
INTO TEMP1   /*Сохраняет выбранные поля во временной таблице Temp1*/
FROM Copy_of_book
WHERE cb_b_id_book = @b_id_book
SELECT @number = COUNT(cb_b_id_book)       /* COUNT подсчитывает количество неповторяющихся записей поля b_id_book */
FROM TEMP1
```

Пример хранимой процедуры на удаление из таблицы «Student». Допустимо, если в таблице «Use_of_libr_student» нет ссылающихся записей.

```
CREATE  PROCEDURE   Delete_student
@num int            /* Объявляем входные переменные */
AS                  /* Проверяем, есть ли ссылающиеся записи, если записей нет, разрешается удаление */
IF NOT EXISTS  (SELECT * FROM Use_of_libr_student
```

```
                    WHERE  us_s_number_of_library_ticket =@num)
    DELETE          /* Оператор удаления */
    FROM Student    /* Имя таблицы, откуда нужно удалить */
    WHERE           /* Условие удаления – удаляем строку, для которой значение поля us_s_number_of_library_ticket совпадает с нужным */
    us_s_number_of_library_ticket = @num
```

Пример хранимой процедуры на обновление таблицы «Student» (изменение фамилии студента).

```
CREATE PROCEDURE Update_student
@number int,        /* Объявляем входные переменные */
@lname varchar(20)
AS
IF EXISTS (SELECT * FROM Student    /* Проверяем, существуют ли студенты, */
   WHERE s_number_of_library_ticket = @number)    /* номер читательского билета которых равен искомому */
UPDATE Student      /* Если такие есть, обновляем таблицу «Student» */
SET s_last_name=@lname    /* полю фамилия присваиваем новое значение */
   WHERE s_number_of_library_ticket = @number    /* если номер читательского билета записи равен искомому */
```

Задание

В разрабатываемой БД необходимо реализовать 20 хранимых процедур, в том числе для вставки, удаления, изменения данных. Должны быть использованы стандартные функции SQL (не менее 20 различных функций), а также выходные параметры процедур.

Содержание отчета: тема и цель работы; SQL-код 20 хранимых процедур, SQL-код вызова хранимых процедур на исполнение.

Контрольные вопросы

1 Что такое хранимая процедура? Для чего используется?

2 Какие виды параметров могут использоваться в процедуре?

3 Как производится средствами T-SQL создание, модификация и удаление хранимых процедур? Как создаются хранимые процедуры на вставку, изменение и удаление данных?

4 Как вызвать средствами T-SQL процедуру с входными и выходными параметрами на исполнение?

5 Описать управление процессом компиляции хранимой процедуры.

14 Язык SQL. Работа с курсорами

Цель: научиться создавать курсоры в MS SQL Server Management Studio.

Теоретические положения

Курсоры в SQL Server представляют собой механизм обмена данными между сервером и клиентом. Курсор позволяет клиентским приложениям работать не с полным набором данных, а только с одной или несколькими строками.

Существует четыре основных типа курсоров, различающихся по предоставляемым возможностям, – статические, динамические, последовательные и ключевые. Тип курсора определяется на стадии его создания и не может быть изменен. Более подробно курсоры описаны в конспекте лекций и в [5, 9].

Далее рассмотрен пример создания курсора для просмотра информации о студентах и выдачи информации об их числе.

```
DECLARE curs1 CURSOR
GLOBAL          /* Создается глобальный курсор, который
                   будет существовать до закрытия данного соединения*/
SCROLL          /* Создает прокручиваемый курсор */
KEYSET          /* Будет создан ключевой курсор */
TYPE_WARNING
FOR
SELECT          /* Какие поля будут показаны в курсоре */
Student.s_number_of_library_ticket, Student.s_last_name, Student.s_first_name,
Student.s_patronymic, Student.s_year_of_entrance, Student.s_faculty
FROM Student                    /* Из какой таблицы выбираются данные */
FOR READ ONLY                   /* Только для чтения */
OPEN GLOBAL curs1               /* открываем глобальный курсор */
DECLARE @@Counter int           /* объявляем переменную*/
SET @@Counter =@@CURSOR_ROWS    /*присваиваем ей число рядов
курсора */
Select @@Counter                /* выводим результат на экран */
CLOSE curs1                     /* закрываем курсор */
DEALLOCATE curs1                /* освобождаем курсор */
```

Задание

В разрабатываемой БД необходимо реализовать пять курсоров статического и динамического типов, содержащих не менее пяти различных функций.

Содержание отчета: тема и цель работы; SQL-код пяти курсоров.

Контрольные вопросы

1 Что такое курсор? Какие типы курсоров различают?
2 Что такое полный и результирующий наборы строк?
3 Какие основные операции выделяют при работе с курсором? С помощью каких команд T-SQL реализуются основные операции?

15 Язык SQL. Работа с триггерами

Цель: научиться создавать триггеры в MS SQL Server Management Studio.

Теоретические положения

Триггер – это специальный тип хранимых процедур, который запускается автоматически на стороне сервера при выполнении тех или иных действий с данными таблицы. Каждый триггер привязывается к конкретной таблице [4, 5, 9].

Напрямую обратиться к триггеру нельзя. Он вызывается автоматически при наступлении соответствующего события в БД – добавление новой строки в таблицу, изменение или удаление строки таблицы. Триггер может срабатывать, когда соответствующее действие с БД выполняет клиентское приложение, хранимая процедура или триггер (другой или тот же самый).

Триггеры DML вызываются при выполнении операторов INSERT, UPDATE или DELETE. Можно указать время вызова триггера:

– AFTER – триггер вызывается после всех действий оператора, если оператор был выполнен успешно. Синонимом в синтаксисе оператора создания триггера является FOR. Также до запуска триггера должны успешно завершиться все каскадные действия и проверки ограничений, на которые есть ссылки;

– INSTEAD OF – триггер вызывается вместо действий, заданных оператором INSERT, UPDATE или DELETE.

Триггеры DDL активируются в ответ на разные события языка описания данных DDL. Эти события прежде всего соответствуют инструкциям Transact-SQL CREATE, ALTER, DROP и некоторым системным хранимым процедурам, которые выполняют схожие с DDL операции.

Триггеры входа могут срабатывать в ответ на событие LOGON, которое возникает при создании пользовательского сеанса.

Для создания триггера используется следующая инструкция T-SQL.

```
CREATE TRIGGER  [ schema_name . ] Trigger_name
ON Table_name | View_name
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
AS
sql_statement [...n] }
```

Trigger_name – задает имя триггера, с помощью которого он будет распознаваться хранимыми процедурами и командами T-SQL. Имя триггера должно быть уникальным в пределах БД. Оно должно характеризовать действие, выполняемое им, записываться в формате <глагол_объект> и содержать в себе имя таблицы и название момента срабатывания.

Table_name | View_name – имя таблицы БД (или представления), к которой будет привязан триггер.

[DELETE] [,] [INSERT] [,] [UPDATE] – определяет инструкции изменения данных, при применении которых к таблице или представлению срабатывает триггер DML. При создании триггера должно быть указано хотя бы одно из этих ключевых слов, и допускается создание триггера, реагирующего на две или три команды в любом сочетании.

WITH APPEND – указание этого ключевого слова требуется для обеспечения совместимости с более ранними версиями MS SQL Server.

sql_statement – определяет набор команд, которые будут выполняться при запуске триггера.

Далее приведен пример триггера, который будет запрещать удаление записей таблицы «Issuing_books», если текущий пользователь не владелец базы данных и если поле «дата выдачи» содержит какое-либо значение.

```
CREATE TRIGGER Ondelete_issuing_books   /* Объявляем имя триггера */
ON Issuing_books                        /* имя таблицы, с которой связан триггер */
FOR DELETE  /*Указываем операцию, на которую будет срабатывать триггер (здесь на  удаление)*/
AS
IF ( SELECT COUNT(*)        /*проверяет*/
FROM Issuing_books          /*записи из таблицы «Issuing_books»*/
WHERE Issuing_books.ib_date_of_issue IS NOT NULL) > 0   /*условие проверяет наличие записи в поле «ib_date_of_issue». Если COUNT(*) возвращает значение, отличное от нуля (т. е. запись есть), то первое условие IF не выполнено*/
AND (CURRENT_USER <> 'dbo')         /*вызывается функция определения имени текущего пользователя и проверяется, владелец ли он*/
BEGIN
PRINT 'у вас нет прав на удаление этой записи'      /*выдача сообщения о неудаче операции*/
ROLLBACK TRANSACTION    /*откат (отмена) транзакции*/
END
```

Задание

В разрабатываемой БД необходимо реализовать 10 триггеров, реагирующих на различные команды (INSERT, DELETE, UPDATE) и содержащих не менее 10 различных стандартных функций SQL.

Содержание отчета: тема и цель работы; SQL-код 10 триггеров.

Контрольные вопросы

1 Что такое триггер? Какие типы триггеров различают?
2 Как создать триггер?
3 С помощью каких команд T-SQL можно изменить или удалить триггер?

16 Создание и изменение таблиц средствами SQL

Цель: получить навыки создания и изменения таблиц средствами T-SQL.

Теоретические положения

Процесс создания таблицы начинается с проектирования ее будущей структуры. В процессе проектирования необходимо определить:

– для хранения каких данных предназначена создаваемая таблица;

– каким образом будет обеспечиваться целостность данных в ней. Для этого следует определить ограничения на значения столбцов (CONSTRAINTS).

Синтаксис создания, изменения и удаления таблицы рассмотрен в [4, 5, 9].

Пример создания таблицы с ограничениями на значения столбцов:

```
CREATE TABLE Production.OrderDetail
(
order_id int CONSTRAINT PK_Order_Id  PRIMARY KEY  IDENTITY(1,1),
unit_price money NULL,
order_qty smallint NULL
      CONSTRAINT   CK_Order_Order_qty  CHECK(order_qty > 5),
total_qty AS (unit_price * order_qty),    /* пример вычисляемого столбца */
rowguid uniqueidentifier ROWGUIDCOL NOT NULL
      CONSTRAINT DF_Order_rowguid DEFAULT (NEWID()),
my_user varchar(20) DEFAULT USER, /* имя пользователя, вставившего строку */
)
```

Пример удаления таблицы после проверки ее существования (IF EXISTS):

```
DROP TABLE IF EXISTS Production.[OrderDetail]
```

В следующем примере к столбцу в таблице добавляется ограничение. В столбце имеется значение, нарушающее это ограничение. Поэтому во избежание проверки ограничения относительно существующих строк, а также для того, чтобы разрешить добавление ограничения, применяется WITH NOCHECK.

```
CREATE TABLE dbo.example (column_a  int);
GO
```

```
INSERT INTO dbo.example VALUES (-1) ;
GO
ALTER TABLE dbo.example WITH NOCHECK
ADD CONSTRAINT ex_check CHECK (column_a > 1) ;
GO
```

Задание

Необходимо создать средствами T-SQL три обычных таблицы и три дополнительных таблицы, обеспечивающие возможность сохранения копий строк из обычных таблиц при удалении данных. При создании обычных таблиц необходимо реализовать все изученные механизмы управления значениями столбцов, а также установить связи между таблицами. Необходимо изменить структуру дополнительных таблиц так, чтобы в них можно было хранить информацию о времени удаления данных из таблицы.

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Перечислить этапы проектирования структуры таблиц.

2 Охарактеризовать основные символы нотации Бэкуса – Наура.

3 С помощью каких команд T-SQL можно создать или изменить таблицу?

4 Охарактеризовать механизмы управления значениями столбцов (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, DEFAULTS, NULL).

5 Хранится ли в таблице значение вычисляемого столбца?

6 Указать способы обеспечения уникальности значений в столбце таблицы.

7 Для чего нужна схема базы данных? Как создать новую схему базы данных? К какой схеме относятся по умолчанию создаваемые таблицы?

8 Как изменить определение столбца таблицы?

9 Как добавить новый столбец или удалить столбец из таблицы?

17 Создание индексов средствами языка SQL

Цель: получить навыки определения индексируемых столбцов таблиц.

Теоретические положения

Индекс представляет собой дополнение к таблице, помогающее ускорить поиск необходимых данных за счет физического или логического их упорядочивания. Он является набором ссылок, упорядоченным по определенному (индексируемому) столбцу таблицы. Физически индекс представляет собой упорядоченный набор значений из индексированного столбца с указателями на места физического размещения исходных строк в структуре базы данных. Использование

индексов позволяет избежать полного сканирования таблицы. На сегодняшний день существует большое количество разновидностей индексов, основными из которых являются кластерный, некластерный, уникальный, покрывающий, полнотекстовый. Более подробно они описаны в [4, 5, 10, 12].

При выборе столбца для индекса следует проанализировать, какие типы запросов чаще всего выполняются и какие столбцы являются ключевыми.

Ключевые столбцы – это такие колонки, которые задают критерии выборки данных, например порядок сортировки. Не стоит индексировать столбцы, которые только считываются и не играют никакой роли в определении порядка выполнения запроса. Не следует индексировать слишком длинные столбцы, например столбцы с адресами или названиями компаний, достигающие длины несколько десятков символов. В крайнем случае, можно создать укороченный вариант такого столбца, выбрав из него до десяти первых символов, и индексировать его. Индексирование длинных столбцов может существенно снизить производительность работы сервера.

Индексы создаются в следующих случаях:

– если операции чтения из таблицы выполняются гораздо чаще, чем операции модификации;

– если поле (или совокупность полей) часто используется в запросах в разделе WHERE;

– если нужно обеспечить уникальность значения поля (или совокупности полей), не являющегося первичным ключом (в этом случае создается уникальный индекс);

– если поле (или совокупность полей) является внешним ключом (т. к. в таком случае индексы могут существенно ускорить выполнение JOIN-запросов).

Формат команды CREATE INDEX на T-SQL имеет вид:

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
INDEX index_name
ON table_or_view_name ( column [...n] )
[ INCLUDE ( column_name [ ,...n ] ) ]
[ WITH { PAD_INDEX = { ON | OFF } | FILLFACTOR = f
    | IGNORE_DUP_KEY } [ ,...n ] ) ]
```

При указании ключевого слова UNIQUE будет создан уникальный индекс, который гарантирует уникальность значений в индексируемом столбце и может быть реализован как для кластерного, так и для некластерного индексов. В одной таблице могут существовать один уникальный кластерный индекс и множество уникальных некластерных индексов.

CLUSTERED – создаваемый индекс будет кластерным. Если аргумент CLUSTERED не указан, создается некластерный индекс. В качестве кластерного индекса следует выбирать наиболее часто используемые и редко изменяемые столбцы.

NONCLUSTERED – создаваемый индекс будет некластерным, и, в отличие

от кластерных, он не перестраивает физическую структуру таблицы, а лишь организует ссылки на соответствующие строки. Каждая таблица может содержать до 999 некластерных индексов независимо от способа их создания: неявно с помощью ограничений PRIMARY KEY и UNIQUE или явно с помощью инструкции CREATE INDEX. Однако лучше ограничиться 4–5 индексами.

index_name – имя индекса, по которому он будет распознаваться командами Transact-SQL. Имя индекса должно быть уникальным в пределах таблицы.

table_or_view_name(column [...n]) – имя таблицы или представления, в которой содержатся один или несколько индексируемых столбцов. В скобках указываются имена столбцов, на основе которых будет построен индекс. Не допускается построение индекса на основе столбцов с типом данных ntext, text, varchar(max), nvarchar(max), varbinary(max), xml или image. В один составной индекс можно включить до 16 столбцов.

Параметр INCLUDE позволяет создать покрывающий индекс, т. е. некластерный индекс, содержащий в своих листовых узлах информацию из дополнительного неключевого поля, не использующегося при создании самого индекса.

В предложении WITH перечисляются параметры создаваемого индекса.

Коэффициент заполнения FILLFACTOR = f задает заполнение в процентах каждой страницы индекса во время его первоначального создания или пересоздания. Значение f можно установить в диапазоне от 1 до 100 (по умолчанию f = 0). Значения коэффициентов заполнения 0 и 100 идентичны. Если f = 100, компонент Database Engine создает индексы с полностью заполненными страницами конечного уровня. Чем больше значение f, тем меньше свободного места в страницах листьев индекса. Например, при значении f = 60 каждая страница листьев индекса будет иметь 40 % свободного места для вставки строк индекса в дальнейшем. Производительность операций считывания снижается обратно пропорционально значению коэффициента заполнения. Для редко изменяемых таблиц рекомендуется принимать f = 100; для часто изменяемых таблиц f = 50–70; в случае промежуточной ситуации f = 80–90 [4].

Параметр PAD_INDEX = {ON | OFF} задает возможность использования разреженного индекса. ON – допустим разреженный индекс, OFF (значение по умолчанию) – недопустим. В случае разреженного индекса должен присутствовать и коэффициент заполнения FILLFACTOR, задающий процент разрежения.

Параметр IGNORE_DUP_KEY = { ON | OFF } определяет ответ на ошибку, случающуюся, когда операция вставки пытается вставить в уникальный индекс повторяющиеся значения ключа. Применяется только к операциям вставки, производимым после создания или перестроения индекса. Параметр не работает во время выполнения инструкции CREATE INDEX, ALTER INDEX или UPDATE. Значение по умолчанию – OFF, которое означает, что если в уникальный индекс вставляются повторяющиеся значения ключа, выводится сообщение об ошибке и будет выполнен откат всей операции INSERT. Значение ON означает, что, если в уникальный индекс вставляются повторяющиеся значения ключа, выводится предупреждающее сообщение и с ошибкой завершаются только строки, нарушающие ограничение уникальности.

Для удаления индекса используется команда DROP INDEX, имеющая следующий синтаксис: DROP INDEX 'table.index' [...n]. Аргумент 'table.index' определяет удаляемый индекс в таблице.

Задание

Необходимо обосновать выбор колонок таблиц для создания индексов в разрабатываемой БД и создать пять индексов для таблиц БД.

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Что такое кластерный, некластерный и уникальный индексы?
2 Какие критерии учитываются при выборе столбцов для индексирования?
3 С помощью каких команд T-SQL задаются различные виды индексов?
4 Перечислить общие рекомендации при планировании стратегии индексирования (назвать не менее восьми рекомендаций).
5 Когда использование индексов нецелесообразно?

18 Назначение прав доступа пользователям к объектам базы данных средствами T-SQL

Цель: изучить основы управления правами доступа к объектам базы данных в СУБД MS SQL Server.

Теоретические положения

Вопросы управления правами доступа к объектам БД подробно описаны в [2, 11]. Здесь же рассмотрен SQL-код основных инструкций.

Для предоставления разрешений permission на защищаемый объект securable участнику principal используется инструкция GRANT, общая концепция которой имеет следующий вид: GRANT <some permission> ON <some object> TO <some user, login or group>.

Синтаксис инструкции GRANT:

```
GRANT { ALL [ PRIVILEGES ] }   | permission [ ( column [ ,...n ] ) ] [ ,...n ]
    [ ON [ class :: ] securable ]
TO principal [ ,...n ]
    [ WITH GRANT OPTION ] [ AS principal ]
```

Охарактеризуем назначение параметров инструкции.

ALL – этот параметр устарел и используется только для поддержки обрат-

ной совместимости. Он не предоставляет все возможные разрешения (см. документацию https://docs.microsoft.com/). Вместо ALL следует предоставлять конкретные разрешения.

PRIVILEGES – включено для обеспечения совместимости с требованиями ISO. Не изменяет работу ALL.

permission – имя разрешения, которое предоставляется пользователю. Можно предоставлять одновременно несколько разрешений.

column – указывает имя столбца таблицы, на который предоставляется разрешение.

class – указывает класс защищаемого объекта, для которого предоставляется разрешение. Квалификатор области :: является обязательным.

securable – указывает защищаемый объект, на который предоставляется разрешение.

TO principal – имя участника. Состав участников, которым можно предоставлять разрешения, меняется в зависимости от защищаемого объекта.

WITH GRANT OPTION – позволяет пользователю, получающему разрешение, получить также возможность предоставлять данное разрешение другим участникам.

Инструкция DENY запрещает разрешение для участника. Предотвращает наследование разрешения участником через его членство в группе или роли. DENY имеет приоритет над всеми разрешениями, но не применяется к владельцам объектов или членам с предопределенной ролью сервера sysadmin (членам роли сервера sysadmin и владельцам объектов не может быть отказано в разрешениях).

Инструкция REVOKE удаляет разрешение, выданное или запрещенное ранее (иногда говорят, что REVOKE используется для неявного отклонения доступа к объектам БД). Синтаксис инструкций DENY и REVOKE сходен с синтаксисом инструкция GRANT и подробно изложен в [2, 11].

Предоставление разрешения удаляет DENY или REVOKE для этого разрешения на данный защищаемый объект. Если то же разрешение запрещено для более высокой области действия, которая содержит данный защищаемый объект, то DENY имеет более высокий приоритет.

Задание

Для разрабатываемой БД необходимо реализовать систему безопасности: создать одну роль и трех пользователей. В созданную роль нужно включить двух пользователей. Третьего пользователя следует включить в одну из стандартных ролей SQL Server.

Далее необходимо назначить права на все ранее разработанные объекты БД. Следует обратить внимание, что не должно быть объектов базы данных, на которые ни у кого из пользователей прав нет. В отчете указывается, с какими объектами базы данных позволяет пользователям работать каждая роль, а также приводятся фрагменты инструкций SQL, позволяющих создать роли, пользователей и предоставить этим пользователям права доступа к объектам базы данных.

Назначение прав доступа на объекты базы данных (таблицы, представления,

хранимые процедуры) должно быть проиллюстрировано таблицей 18.1 с помощью общепринятых сокращений, произошедших от аббревиатуры CRUD – Create, Read, Update, Delete (C – право на создание записи, R – право на чтение записи, U – право на обновление записи, D – право на удаление записи). Также можно использовать сокращение E (Execute), например, для хранимых процедур. Иные виды назначенных разрешений прописываются указанием их полных наименований (общее количество разрешений для SQL Server, начиная с выпуска 2019 г., составляет 248).

Таблица 18.1 – Перечень прав пользователей (категорий пользователей) относительно объектов базы данных

Имя объекта базы данных	Пользователь 1	Пользователь 2	Пользователь *N*	Приложение	Гость	Администратор
Table1	CRU	R	U	R	–	CRUD
View1	R	R	CRU	R	R	CRUD

Содержание отчета: тема и цель работы; описание распределения прав доступа пользователей; SQL-код выполнения задания.

Контрольные вопросы

1 На какие категории можно разделить права в SQL Server?

2 Перечислить стандартные роли сервера и базы данных.

3 Какие команды T-SQL используются для предоставления прав доступа, запрещения и неявного отклонения доступа?

Список литературы

1 **ГОСТ 34.602–89**. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Электронный ресурс]. – Москва : Стандартинформ, 2009. – Режим доступа: http://docs.cntd.ru/document/gost-34-602-89. – Дата доступа: 20.09.2022.

2 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: в 2 т. Т. 2: Распределенные и удаленные базы данных / В. П. Агальцов. – Москва: ФОРУМ; ИНФРА-М, 2021. – 271 с. – Режим доступа: https://znanium.com/catalog/product/1514118. – Дата доступа: 20.09.2022.

3 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: в 2 кн. Кн. 1: Локальные базы данных – Москва: ФОРУМ; ИНФРА-М, 2020. – 352 с.: ил. – Режим доступа: https://znanium.com/catalog/product/1068927. – Дата доступа: 20.09.2022.

4 **Бен-Ган, И.** Microsoft SQL Server 2012. Создание запросов: учебный

курс Microsoft : пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. – Москва: Русская редакция, 2015. – 720 с.: ил.

5 **Бондарь, А. Г.** Microsoft SQL Server 2014 / А. Г. Бондарь. – Санкт-Петербург : БХВ-Петербург, 2015. – 592 с. : ил.

6 **Гвоздева, Т. В.** Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум : учебно-справочное пособие / Т. В. Гвоздева, Б. А. Баллод. – Санкт-Петербург ; Москва ; Краснодар : Лань, 2018. – 156 с. : ил.

7 **Голицына, О. Л.** Базы данных [Электронный ресурс]: учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва: ФОРУМ; ИНФРА-М, 2020. – 400 с. – Режим доступа: https://znanium.com/catalog/product/1053934. – Дата доступа: 20.09.2022.

8 **Кузин, А. В.** Разработка баз данных в системе Microsoft Access [Электронный ресурс]: учебник / А. В. Кузин, В. М. Демин. – 4-е изд. – Москва: ФОРУМ; ИНФРА-М, 2020. – 224 с. – Режим доступа: https://znanium.com/catalog/product/1058247. – Дата доступа: 20.09.2022.

9 **Куликов, С. С.** Работа с MySQL, MS SQL Server и Oracle в примерах [Электронный ресурс]: практическое пособие / С. С. Куликов. – Минск: БОФФ, 2016. – 556 с. – Режим доступа: http://svyatoslav.biz/database_book/. – Дата доступа: 20.09.2022.

10 **Куликов, С. С.** Реляционные базы данных в примерах [Электронный ресурс]: практическое пособие для программистов и тестировщиков / С. С. Куликов. – Минск: Четыре четверти, 2020. – 424 с. – Режим доступа: http://svyatoslav.biz/relational_databases_book/. – Дата доступа: 20.09.2022.

11 **Полищук, Ю. В.** Базы данных и их безопасность [Электронный ресурс]: учебное пособие / Ю. В. Полищук, А. С. Боровский. – Москва: ИНФРА-М, 2020. – 210 с. – Режим доступа: https://znanium.com/catalog/product/1011088. – Дата доступа: 20.09.2022.

12 **Шустова, Л. И.** Базы данных [Электронный ресурс]: учебник / Л. И. Шустова, О. В. Тараканов. – Москва : ИНФРА-М, 2021. – 304 с. – Режим доступа: https://znanium.com/catalog/product/1362122. – Дата доступа: 20.09.2022.