МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

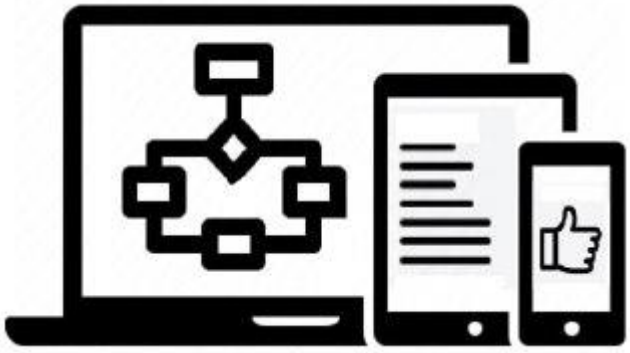
Кафедра «Программное обеспечение информационных систем»

БАЗЫ И БАНКИ ДАННЫХ

Методические рекомендации
к лабораторным работам
для студентов специальности
1-53 01 02 «Автоматизированные системы
обработки информации»
дневной и заочной форм обучения

Часть 1

1 модуль



Могилев 2021

Содержание

1 Разработка технического задания на проектирование информационной системы

4 часа

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, основные элементы которой представлены 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.

Отделы обслуживания работают с читателями дневного отделения по читательским билетам, заочного – по зачетной книжке. Номер билета соответствует номеру формуляра, который содержит информацию о выданных книгах.

После окончания каждого курса студент должен сдать все неиспользуемые книги. По завершении обучения в университете студент сдает все библиотечные книги, подписывает обходной лист.

Библиотекой также могут пользоваться преподаватели – сотрудники университета, которым также выдаются читательские билеты. Кроме того, преподаватели могут заказывать учебную литературу.

Каждый отдел библиотеки выполняет свои функции.

Справочно-библиографический отдел 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.

Отдел комплектования осуществляет следующие функции:

– заказ литературы по тематическим планам издательств;

– обработка новой литературы;

– классификация литературы по ББК;

– списание ветхой, устаревшей литературы;

– замена утерянных книг на новую литературу.

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

В функции отделов обслуживания входят:

– запись студентов в библиотеку;

– выдача книг читателям и прием книг;

– ведение картотек читателей.

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

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

В функции директора при работе с ИС входят:

– координация полной работы всех отделов библиотеки;

– составление отчетности и плана работы библиотеки.

Информация, подлежащая хранению: имя и фамилия сотрудника, табельный номер, дата вступления в должность, дата рождения, отдел, в котором работает, фамилия начальника отдела, размер книгообеспеченности фонда, посещаемость, читаемость.

Директор библиотеки (как и заместитель директора библиотеки) имеет доступ ко всей информации в БД ИС.

В функции заведующего отделом входит:

– координация работы отдела библиотеки;

– составление отчета о работе отдела (раз в год в декабре) и плана работы отдела (раз в год в январе).

Информация, подлежащая хранению: имя и фамилия сотрудника, табельный номер, стаж работы, квалификация, дата рождения.

Заведующий отделом имеет доступ к информации о сотрудниках своего отдела, о читателях, о книжном фонде.

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)).

Функции, выполняемые подсистемами объекта автоматизации:

4.2.1 запрос информации о книгах, выданных студенту, сотруднику;

4.2.2 добавление нового читателя в библиотеку (в БД), проверка его данных;

4.2.3 запрос информации о наличии книг в книжном фонде;

4.2.4 формирование тематических списков литературы;

4.2.5 формирование отчетов о работе отделов;

4.2.6 добавление новых книг в каталог библиотеки.

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 подразделе 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 Средства концептуального проектирования функциональных моделей информационных систем

4 часа

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.

Бизнес-процесс (ГОСТ Р ИСО 19440) представляет собой набор видов деятельности предприятия или набор действий людей и (или) машин), направленных на реализацию одной или более задач предприятия для достижения цели, создание определённого продукта или услуги для потребителей.

В соответствии с глоссарием стандарта BPMN 2.0 бизнес-процесс (Process) представляется «графом Flow-элементов (набором активностей, событий, шлю-

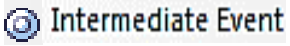
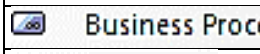
зов) и отношений Sequence Flow, связывающих их в исполняемый поток» (https://www.omg.org/spec/BPMN/2.0).

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).

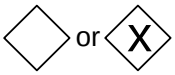
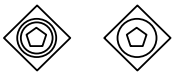
Далее в окне Model Wizard в разделе BPMN следует выбрать тип модели BPMN 2.0 Business Process, добавляемой в основной пакет Model.

В таблице 2.1 приведен перечень основных элементов диаграммы BPMN Business Process.

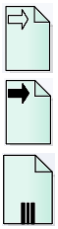
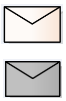
Таблица 2.1 – Элементы диаграммы BPMN Business Process в Enterprise Architect

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 автоматизированных механизмов или 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 предоставленной этим механизмом информации об изменениях в бизнес-процессе)

Продолжение таблицы 2.1

1	2
	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)
	script task (сценарий, скрипт – задача, выполняемая автоматизированной системой без участия человека)
	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)
	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. В случае запуска 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. При ветвлении активируется одна или более ветвей. При слиянии все выполняющиеся входящие ветви должны быть завершены
	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. При синхронизации (слиянии) параллельных ветвей оператор ждет завершения всех входящих ветвей, и затем активирует исходящий поток
Зоны ответственности	
	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

Окончание таблицы 2.1

1	2
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
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) сообщение (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)
Conversation Link	Связь диалога
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, пояснения, улучшающие читабельность 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).

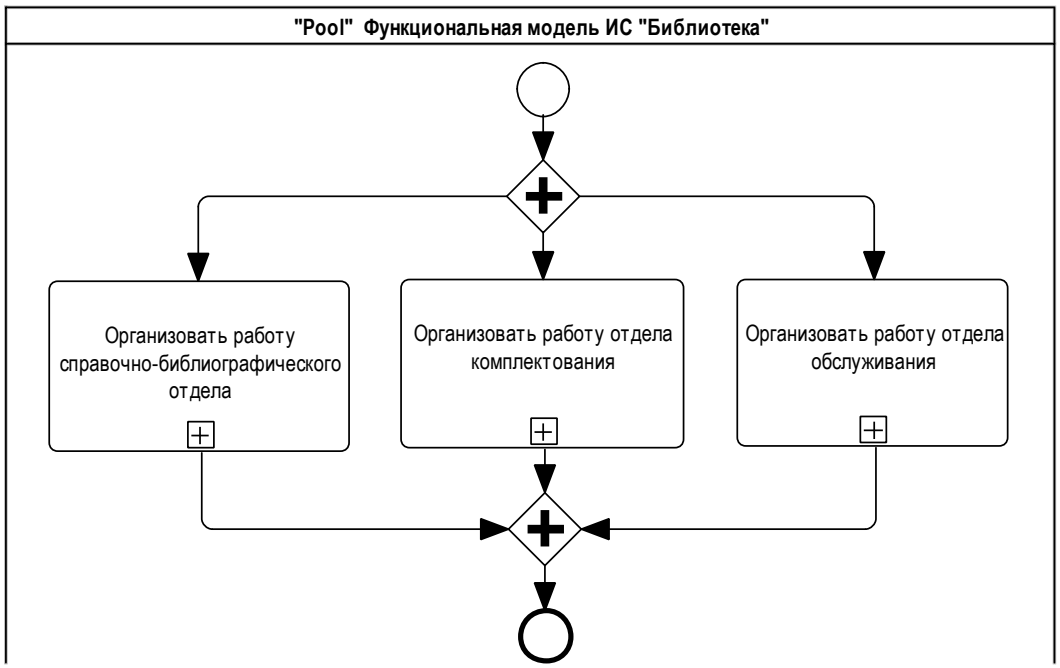


Рисунок 2.1 – Диаграмма подпроцессов функциональной модели (первый уровень)

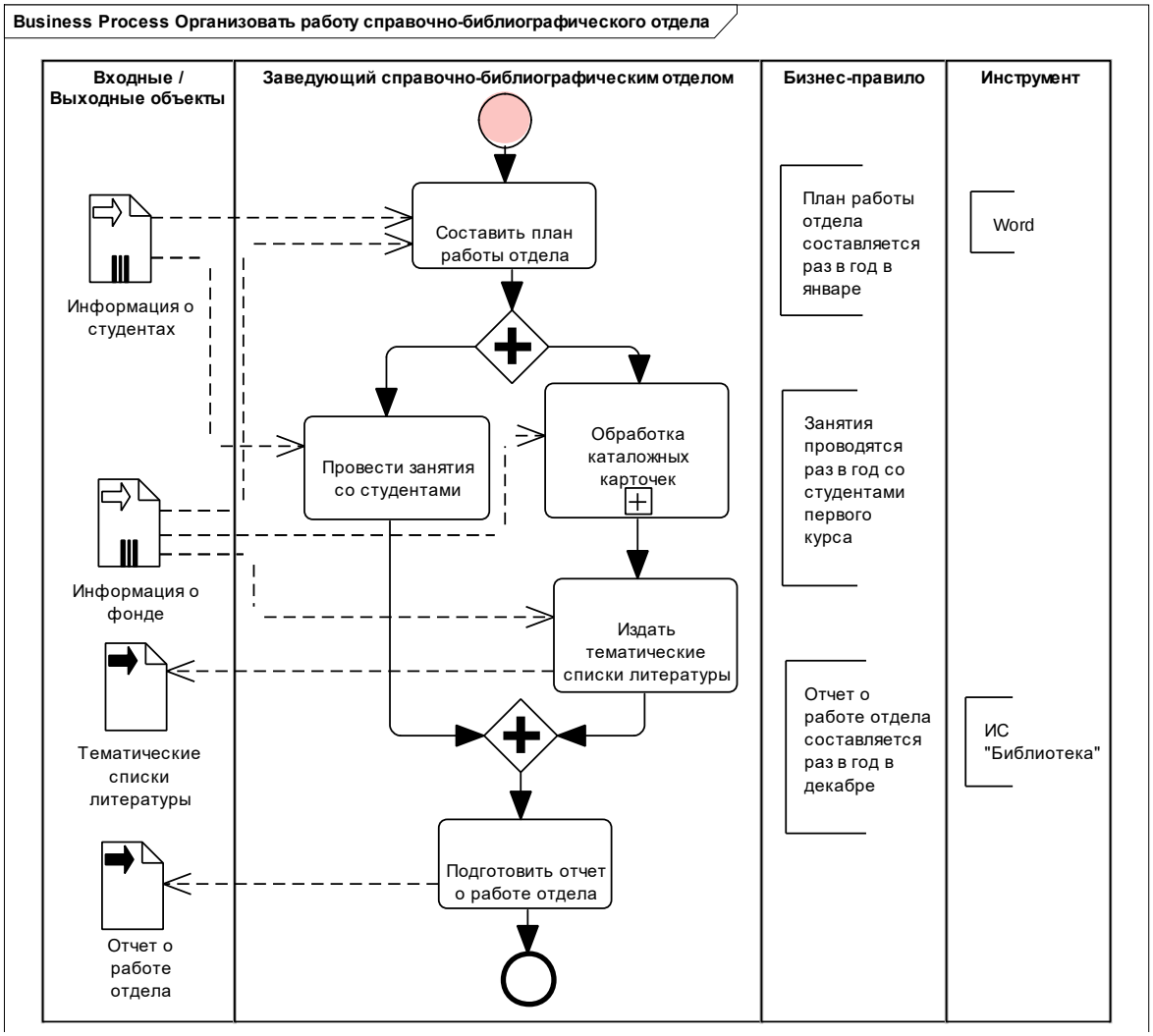


Рисунок 2.2 – Диаграмма декомпозиции подпроцесса «Организовать работу справочно-библиографического отдела» (второй уровень)

Процесс начинается с определенного (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).

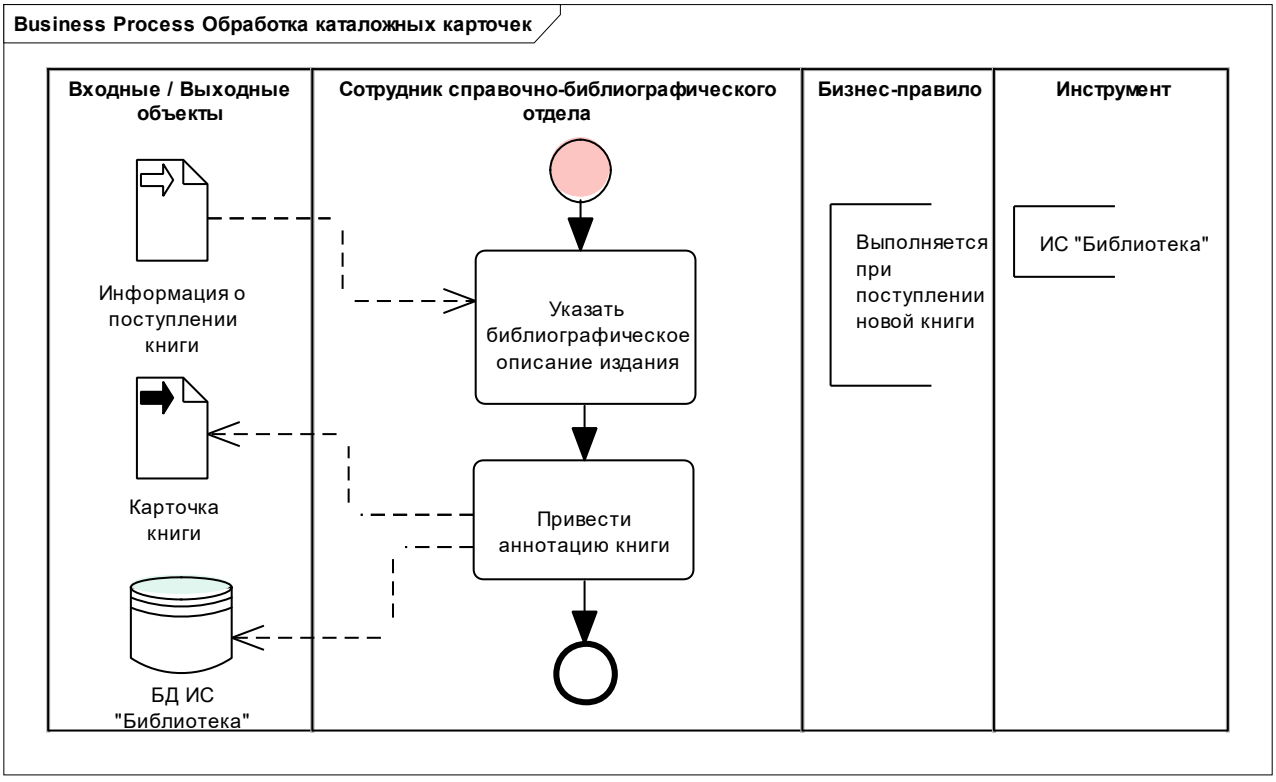


Рисунок 2.3 – Диаграмма декомпозиции подпроцесса «Обработка каталожных карточек» (третий уровень)

Задание

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

Модель должна содержать не менее 3 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 средств концептуального проектирования информационной модели системы в Enterprise Architect

4 часа

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

Теоретические положения

Взаимосвязи между основными понятиями БД представлены в таблице 3.1.

Таблица 3.1 – Основные понятия баз данных

Точка зрения	Предметная область	Реляционная теория	Физическое хранение	Стандарт SQL, SQL Server	Объектная модель
Взаимное соответствие терминов	Сущность	Отношение (Relation)	Файл	Таблица (Table)	Класс
	Свойство	Атрибут (Attribute)	Поле (Field)	Столбец (Column)	Свойство
	Экземпляр	Кортеж (Tuple)	Запись (Record)	Строка (Row)	Объект

Первичный ключ (Primary Key – PK) – это один или несколько атрибутов, однозначно идентифицирующих каждый экземпляр сущности.

Первичный ключ должен удовлетворять двум требованиям – уникальности и безызбыточности (минимальности).

Уникальность ключа означает, что в любой момент времени таблица БД не может содержать никакие две различные записи, имеющие одинаковые значения ключевых полей.

Требование безызбыточности ключевых полей означает, что только сочетание значений выбранных полей отвечает требованиям уникальности записей таблицы БД. Это означает также, что ни одно из входящих в ключ полей не может быть исключено из него без нарушения уникальности.

Исходя из требований уникальности и безызбыточности атрибут (атрибуты), выбираемый для первичного ключа, должен:

1) уникально идентифицировать значение сущности;

2) не содержать неопределенное значение NULL;

3) сохранять уникальность с течением времени;

4) содержать как можно меньше символов, что облегчает индексацию и восстановление данных.

Составной ключ – первичный ключ, состоящий из нескольких атрибутов.

При формировании составного ключа не следует включать в состав ключа поля таблицы, значения которых сами по себе однозначно идентифицируют записи в таблице. Например, не стоит создавать ключ, содержащий одновременно поля «номер паспорта» и «уникальный номер налогоплательщика», т. к. каждый из этих атрибутов может однозначно идентифицировать записи в таблице.

В качестве первичного ключа может использоваться естественный ключ или суррогатный.

Естественный ключ – это атрибут или набор атрибутов, которые однозначно идентифицируют значение сущности (записи таблицы) и имеют некий физический смысл вне БД (номер детали и т. д.). Уникальность записи может достигаться не только значением уникального кода внешнего объекта, но, например, и датой вставки записи в таблицу (первичный ключ РК состоит из значения кода объекта и даты вставки записи в БД).

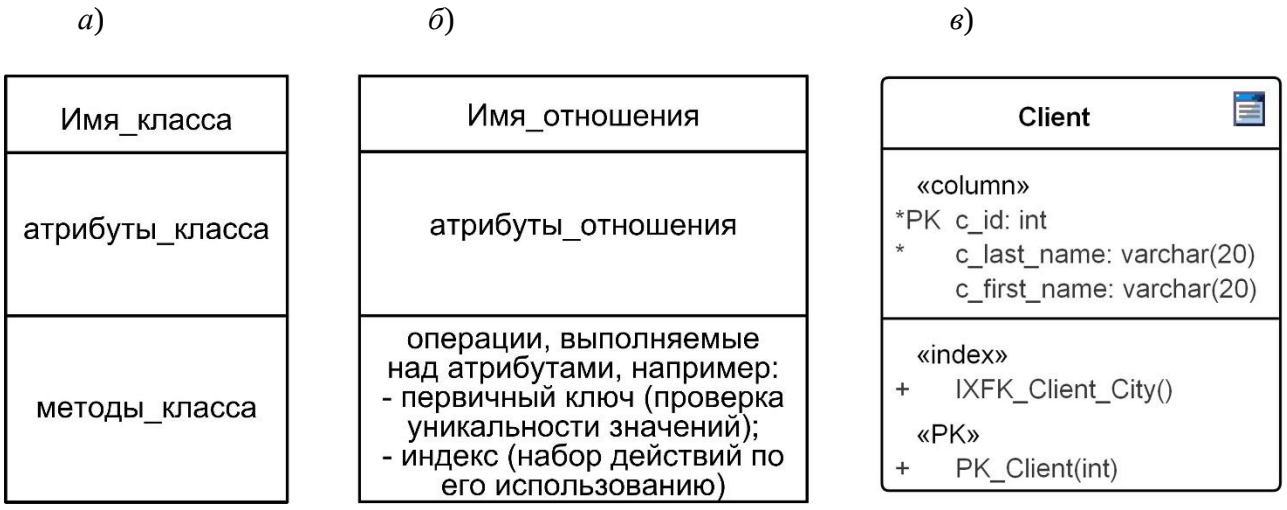
Суррогатный ключ – автоматически сгенерированное значение внутреннего ID – идентификатора сущности, никак не связанное с информационным содержанием записи, которое имеет некий физический смысл только внутри БД и уникально на всем протяжении жизни сущности; обычно в роли суррогатного ключа могут использоваться данные типа INT, SMALLINT, TINYINT.

Внешний ключ (Foreign Key – FK) – это столбец (или группа столбцов), содержащий значения, совпадающие со значениями первичного ключа в другой таблице. Он обеспечивает связь сущностей главной и подчиненной таблиц. Внешний ключ не обладает свойством уникальности и обычно является частью составного первичного ключа или неключевым столбцом.

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 [9, с. 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). Во всех указанных нотациях разработка схемы базы данных в Enterprise Architect ведется с использованием основных понятий языка UML «класс», «атрибут» и «метод».

На рисунке 3.1 показано, как основные понятия языка UML (Unified Modeling Language – унифицированный язык моделирования, язык графического описания для объектного моделирования в области разработки программного обеспечения) связаны с разработкой баз данных. Прямоугольник обозначает класс или отношение, атрибуты класса – атрибуты отношения, к методам отношения относят первичный ключ, внешний ключ, индексы, т. к., например, первичный ключ – это не просто поле, обладающее определенными свойствами, но и наличие необходимости выполнять проверку уникальности значений



а – графическое изображение структуры класса в UML; *б* – графическое изображение структуры отношения в Enterprise Architect; *в* – пример отношения в Enterprise Architect

Рисунок 3.1 – Структура отношения в Enterprise Architect

Выбор той или иной нотации (UML, IDEF1X, IE) может определяться как корпоративными стандартами разработчика, так и требованиями заказчика.

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 – включает логическую модель БД, состоящую из таблиц, атрибутов, ограничений на значения атрибутов 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. Так, для MS SQL Server данный пакет может содержать хранимые процедуры, функции, запросы, представления, последовательности, триггеры, таблицы (разработанные с помощью инструмента Database Builder из папки Tools). Для MS Access данный пакет может содержать запросы, представления и таблицы (разработанные с помощью инструмента Database Builder из папки Tools).

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.

Вначале на диаграмме Logical Model создадим таблицу Client и установим для нее целевую СУБД SQLServer2012.

Выделив созданную таблицу и выбрав в контекстном меню пункт Columns, для таблицы Client создадим первичный ключ c_id. В меню Client Columns and Constraints в разделе Columns необходимо указать тип данных int столбца c_id, задать галочкой ограничение первичного ключа PK. Одновременно в разделе Constraints появится автоматически сгенерированное название ограничения первичного ключа PK_Client.

Зададим для первичного ключа свойство IDENTITY, т. е. сделаем столбец первичного ключа идентификатором (счетчиком со свойством AutoNum = True) с начальным значением StartNum = 1 и приращением Increment = 1. Добавим в таблицу Client столбцы last_name и first_name. Звездочка слева от названия столбца на схеме таблицы (* PK id_client, * last_name) означает, что для такого столбца задано ограничение Not Null (рисунок 3.2).

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.

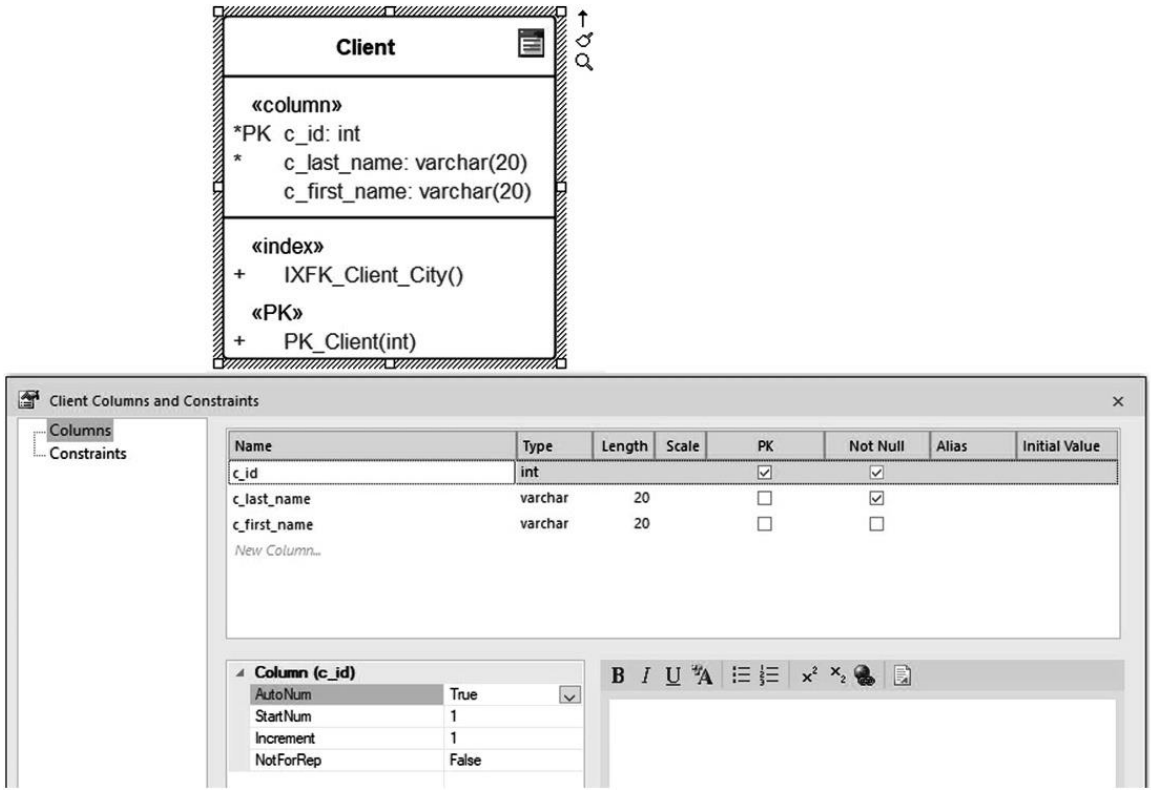


Рисунок 3.2 – Создание таблицы Client

Связи. Типы данных столбца первичного ключа и соответствующего столбца внешнего ключа должны полностью совпадать (кроме автоинкрементируемости, которой не должно быть у внешнего ключа). Это позволяет СУБД не затрачивать дополнительные ресурсы на преобразование данных при сравнении значений первичного и внешнего ключа.

В Enterprise Architect на панели инструментов для создания связей между отношениями имеется только один инструмент – ассоциация. Поэтому для того, чтобы реализовать требуемый тип связи (1:1, 1:M, M:M) между отношениями с помощью одной лишь ассоциации, нужно хорошо понимать устройство и смысл различных типов связей.

Ассоциация – это самый общий тип связи, который просто показывает, что некие сущности взаимосвязаны, при этом тип и особенности этой взаимосвязи не отражаются, хотя некоторые параметры можно указать (направленность связи, мощность связи).

Связь, т. е. ассоциация, позволяющая установить некоторые взаимосвязи между сущностями, имеют следующие свойства:

1) **направленность**. Большинство средств проектирования схем баз данных (не только Enterprise Architect) при создании связи придерживается правила «от дочерней к родительской» таблице. В Enterprise Architect в нотациях UML, IDEF1X, IE стрелки связей на схемах изображаются просто **от дочернего отношения к родительскому**, а не от конкретного поля дочерней таблицы к конкретному полю родительской таблицы. Понять, по каким полям связаны дочернее и родительское отношения, можно из надписи на самой связи.

Главная (родительская) таблица находится на стороне «один» связи, а подчинённая (дочерняя) таблица находится на стороне «много». Связи многие-ко-многим симметричны;

2) **мощность** (кардинальность, cardinality) связи – свойство связи, которое служит для указания количества взаимосвязанных строк в таблицах, объединённых связью (например: 1:M, 1:5 (один к пяти) и т. д.).

Мощность связи задается при ее создании в окне Foreign Key Constraint и может принимать следующие значения: 1..*, 1.., 1, 0..1, 0..*, 0, *.

Значения мощности связи определяются исключительно требованиями предметной области, которую описывает база данных. Например:

– в системе обязательно должен быть хотя бы один модератор, но их может быть не более двух → «1 к 1..2»;

– сотрудник при отсутствии заказов может не вести ни один договор, но если заказы есть, то не более пяти договоров → «1 к 0..5»;

– детское автокресло может быть никем не арендовано, и каждый человек может арендовать не более трех автокресел → «0…1 к 0..3».

Ограничения мощности связей можно контролировать с помощью триггеров или из клиентского приложения;

3) **обязательность** (Nulls Allowed или No Nulls) показывает, может ли атрибут внешнего ключа принимать значение «Null» в таблице БД;

4) **тип**. Существует три классических вида связей между таблицами [2–11].

В связи 1:1 (один к одному) может участвовать не больше одного экземпляра каждой из связываемых сущностей, т. е. одной записи в таблице *A* может соответствовать не более одной записи в таблице *B*. Например, каждый преподаватель ведет не более одной дисциплины, и каждая дисциплина ведется не более чем одним преподавателем.

В связи 1:M (один ко многим) один экземпляр (или даже нуль экземпляров) родительской сущности может быть связан с любым количеством экземпляров дочерней сущности. И наоборот – для связи M:1 (многие к одному). Например, в связи 1:M один преподаватель может преподавать любое количество учебных дисциплин, но каждая дисциплина преподается не более чем одним преподавателем.

В связи M:M (многие-ко-многим или неспецифическая связь) каждый из экземпляров обеих сущностей может быть связан с любым числом экземпляров другой сущности. Например, каждый преподаватель может вести любое количество дисциплин, и каждая дисциплина может преподаваться любым количеством преподавателей.

Связь между сущностями может быть одного из следующих типов:

а) идентифицирующая связь;

б) неидентифицирующая (обязательная или необязательная);

в) категоризации (типизирующая);

г) рекурсивная (связь сущности самой с собой).

Внешний ключ в зависимости от типа связи может стать частью составного ключа дочерней сущности или неключевым атрибутом дочерней сущности.

С помощью внешнего ключа экземпляр дочерней сущности ссылается на соответствующий экземпляр родительской сущности.

Создание идентифицирующей связи.

Идентифицирующей является связь между двумя сущностями, в которой каждый экземпляр подчиненной (дочерней) сущности идентифицируется (однозначно определяется) значениями атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности зависит от независимой родительской сущности и не может существовать без экземпляра родительской сущности. Соответственно, запись в дочерней таблице не может существовать без соответствующей записи в родительской таблице. Для гарантированного обеспечения этого свойства атрибуты первичного ключа родительской сущности мигрируют в состав *первичного ключа* дочерней сущности и помечаются там как внешний ключ (FK). И 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.

В качестве примера рассмотрим взаимосвязь между сущностями «Клиент» и «Заказ», которые представлены на рисунке двумя таблицами Client и Order.

Каждый клиент может иметь много заказов, потому на схеме базы данных это можно показать связью типа 1:M, а т. к. по бизнес-правилам каждый заказ обязан быть строго привязан к соответствующему клиенту и может быть определён только через связь с данным клиентом (и не имеет смысла без конкретного клиента), связь будет идентифицирующей.

Создадим связь типа 1:M. Для этого **от дочернего отношения** Order протянем ассоциацию **к родительскому отношению** Client.

В появившемся окне Foreign Key Constraint будет предложено создать в дочернем отношении Order столбец внешнего ключа c_id (*) и задать имя для ограничения внешнего ключа FK_Order_Client.

После нажатия OK ограничение внешнего ключа FK_Order_Client и связь «FK» будут созданы.

В соответствии с определением идентифицирующей связи внешний ключ нужно сделать частью составного первичного ключа. Открыв из контекстного меню таблицы Order окно Columns, нужно для столбца o_c_id поставить галочку в чекбоксе PK.

В результате будет создана идентифицирующая связь (типа 1:M). Столбец внешнего ключа o_c_id в составе первичного ключа будет помечен обозначением pfK (рисунок 3.3).

Если выделить созданную таблицу Client, выбрать в контекстном меню пункт Code Engineering → Generate DDL, в появившемся окне Generate DDL задать Generate To DDL Execution Engine и нажать кнопку Generate, то будет автоматически сгенерирован код SQL, описывающий таблицу Client. Указанный код будет размещен на вкладке Execute DDL инструмента Database Builder создания физической модели базы данных.

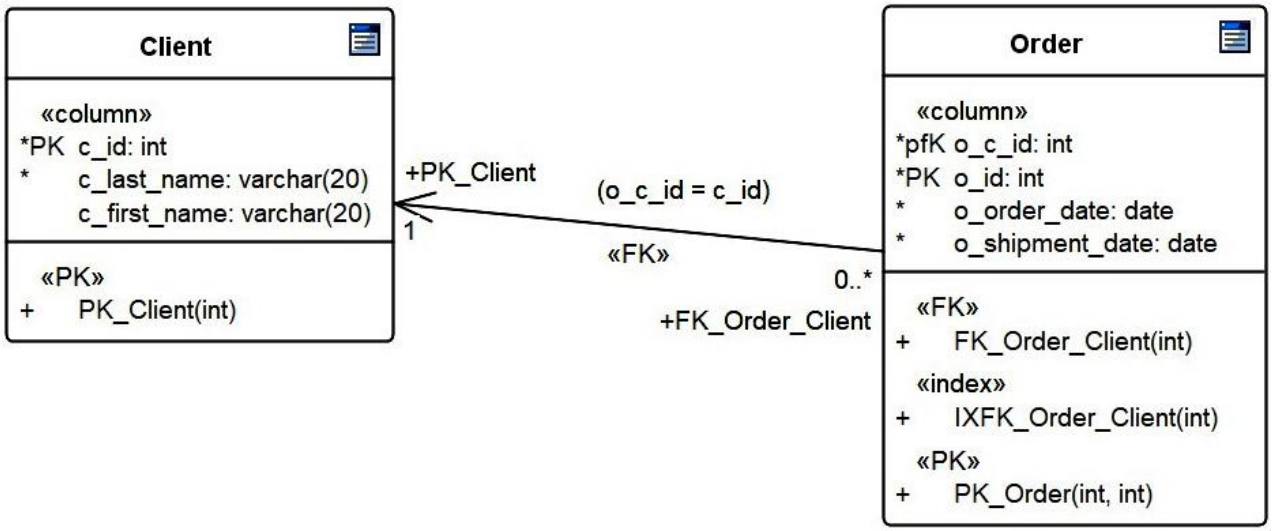


Рисунок 3.3 – Идентифицирующая связь 1:M

Приведем SQL-код таблицы Client:

```
CREATE TABLE [Client]
(
	[c_id] int NOT NULL IDENTITY (1, 1),
	[c_last_name] varchar(20) NOT NULL,
	[c_first_name] varchar(20) NULL
)
```

В разделе Client.PK.CIient вкладки Execute DDL находится автоматически сгенерированный SQL-код ограничения первичного ключа PK_Client.

```
ALTER TABLE [Client]
ADD CONSTRAINT [PK_Client]
	PRIMARY KEY CLUSTERED ([c_id] ASC)
```

SQL-код таблицы Order:

```
CREATE TABLE [Order]
(
	[o_c_id] int NOT NULL,
	[o_id] int NOT NULL,
	[o_order_date] date NOT NULL,
	[o_shipment_date] date NOT NULL
)
```

SQL-код ограничения первичного ключа таблицы Order:

```
ALTER TABLE [Order]
```

```
ADD CONSTRAINT [PK_Order]
    PRIMARY KEY CLUSTERED ([o_c_id] ASC, [o_id] ASC)
```

SQL-код ограничения внешнего ключа таблицы Order:

```
ALTER TABLE [Order] ADD CONSTRAINT [FK_Order_Client]
    FOREIGN KEY ([o_c_id]) REFERENCES [Client] ([c_id])
```

SQL-код, обеспечивающий ссылочную целостность данных в обеих связанных таблицах в ограничении внешнего ключа:

```
ON DELETE No Action
ON UPDATE No Action
```

SQL-код некластерного индекса, созданного для столбца внешнего ключа таблицы Order:

```
CREATE NONCLUSTERED INDEX [IXFK_Order_Client]
ON [Order] ([o_c_id] ASC)
```

Приведем пример **идентифицирующей связи типа 1:1**.

На рисунке у каждого клиента может быть только один паспорт, и у каждого паспорта может быть только один владелец, поэтому между сущностями Client и Passport связь 1:1 (рисунок 3.4). Так как каждый паспорт обязан быть строго привязан к соответствующему клиенту и может быть определён только через связь с данным клиентом (и не имеет смысла без конкретного клиента), связь будет идентифицирующей.

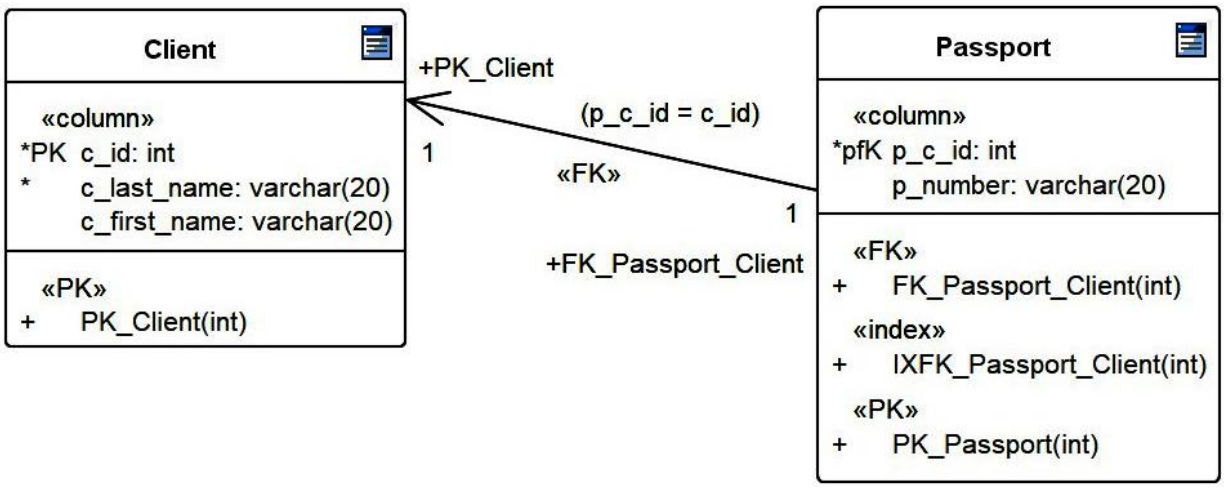


Рисунок 3.4 – Идентифицирующая связь 1:1

Идентифицирующая связь «один к одному» на рисунках изображается практически так же, как 1:M (отличие состоит в числах, указывающих мощность

связи, и в том, что первичный ключ в дочерней таблице одновременно является и внешним).

Создание неидентифицирующей связи.

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. Кортежу дочернего отношения может не соответствовать ни одного кортежа родительского отношения, а сам кортеж дочернего отношения может быть полностью определён без использования значения первичного ключа соответствующего кортежа родительского отношения [9]. Атрибуты первичного ключа родительской сущности мигрируют в подчиненную, чтобы стать там *внешним ключом* (неключевыми атрибутами).

Для неидентифицирующей связи необходимо указать **обязательность связи** (Nulls Allowed или No Nulls), которая показывает, может ли атрибут внешнего ключа принимать значение «Null» в таблице БД.

Неидентифицирующая связь называется **обязательной** (No Nulls), если все экземпляры дочерней сущности должны участвовать в связи. В случае обязательной связи несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности, 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 случае необязательной связи внешний ключ дочерней сущности может принимать значение NULL. Это означает, что экземпляр дочерней сущности не будет связан ни с одним экземпляром родительской сущности.

При создании неидентифицирующей связи первичный ключ родительского отношения в процессе миграции в дочернее отношение становится неключевым атрибутом и внешним ключом.

На рисунке 3.5 показаны сущности Client и City. Поскольку эти сущности могут идентифицироваться независимо друг от друга, 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*. Поскольку один клиент относится к одному городу, а из одного города может быть множество клиентов, то устанавливаемая связь будет иметь тип 1:M.

Для того, чтобы создаваемую связь определить как **неидентифицирующую**, внешний ключ не включен в состав первичного ключа, а размещен среди неключевых атрибутов.

Для того, чтобы создаваемую неидентифицирующую связь определить как **обязательную**, для столбца внешнего ключа c_ci_id в отношении Client установлен признак Not Null (c_ci_id помечен звездочкой).

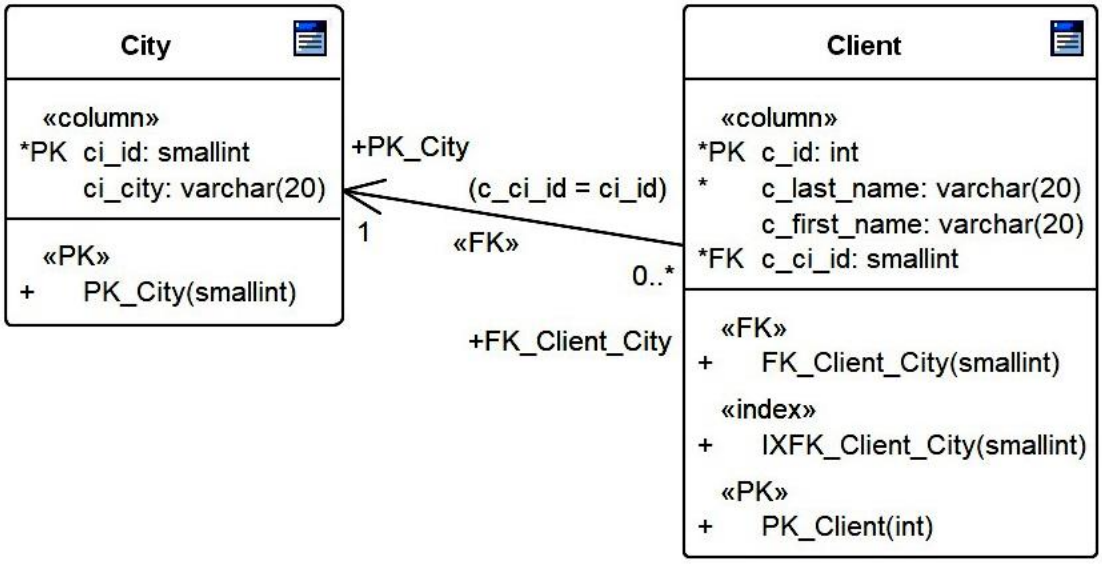


Рисунок 3.5 – Неидентифицирующая обязательная связь

Если существует бизнес-правило, в соответствии с которым клиент необязательно должен сопоставляться с городом, то неидентифицирующая связь между отношениями Client и City будет *необязательной*. Для того, чтобы создаваемую связь определить как **необязательную**, столбцу внешнего ключа c_ci_id в отношении Client разрешено принимать значения Null (рисунок 3.6).

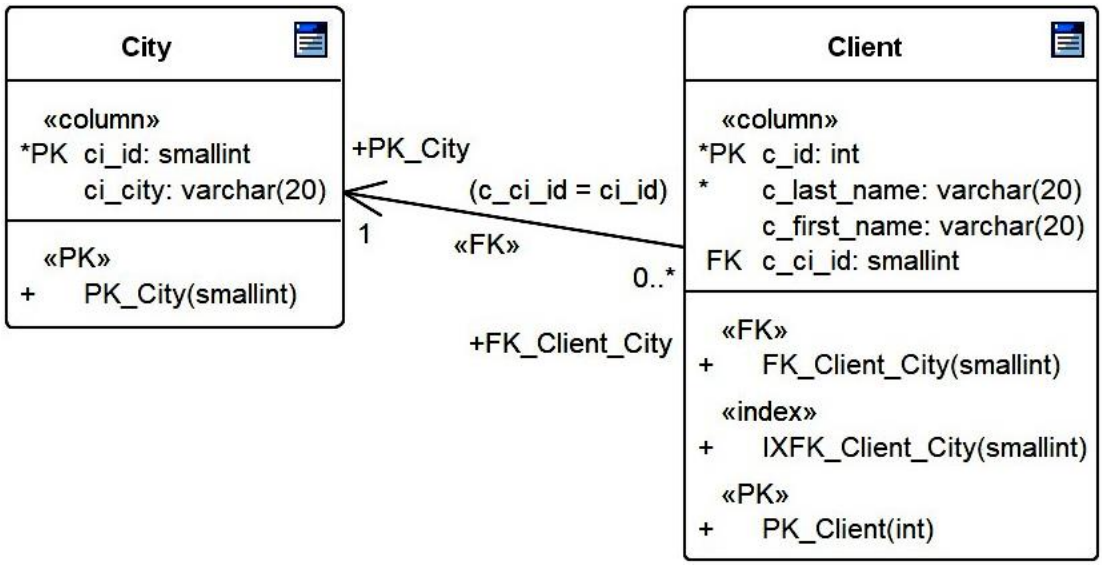


Рисунок 3.6 – Неидентифицирующая необязательная связь

Создание связи M:M.

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 поддержи-

вают связь M:M, такая связь организуется с помощью дополнительной ассоциативной сущности и двух идентифицирующих связей 1:M.

На рисунке 3.7 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 дополнительной ассоциативной сущности.

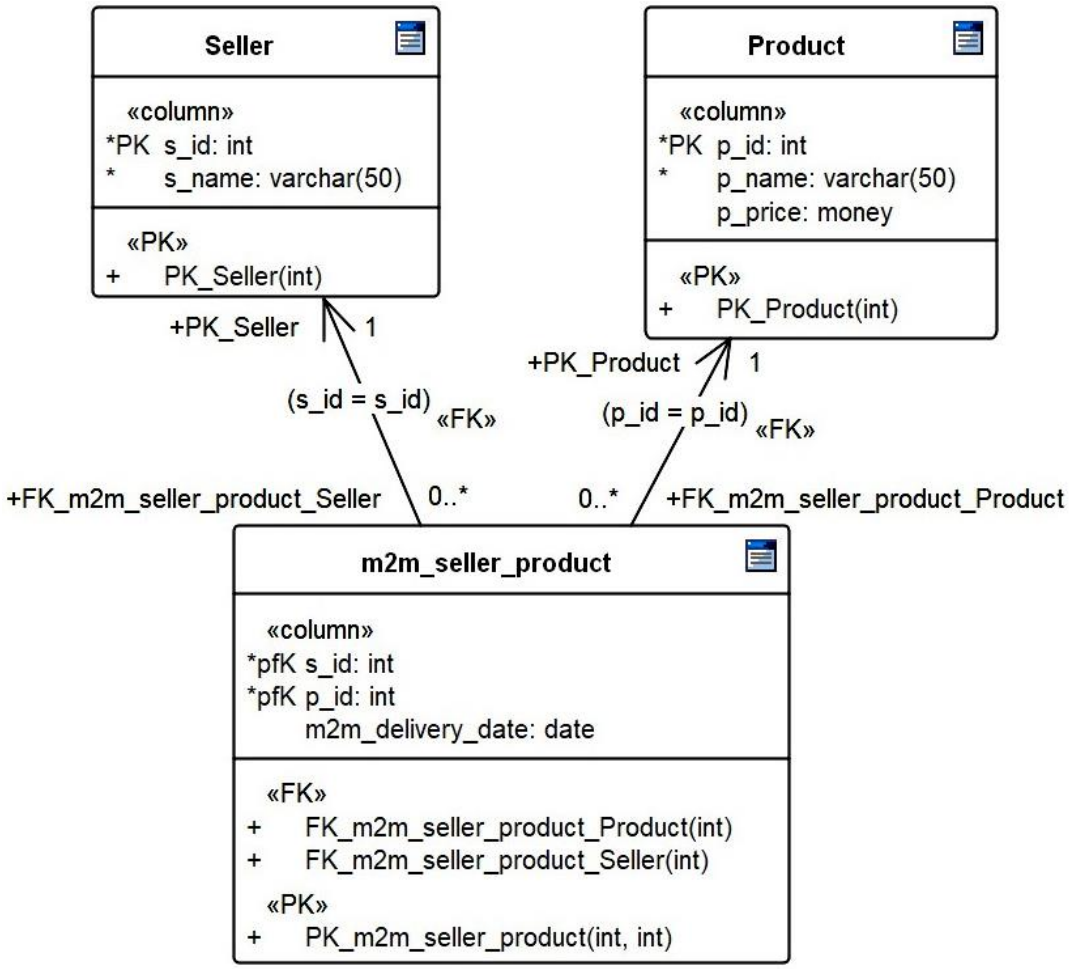


Рисунок 3.7 – Создание связи M:M

Создание рекурсивной связи. Рекурсивная связь – это неидентифицирующая необязательная связь сущности с самой собой (одна и та же сущность является и родительской, и дочерней одновременно), которая отражает тот факт, что экземпляр сущности может быть связан с другим экземпляром той же самой сущности. Рекурсивная связь создают с помощью рекурсивного внешнего ключа.

Рекурсивный внешний ключ – это разновидность внешнего ключа, использующая в случае, когда таблица ссылается на саму себя. Атрибут рекурсивного внешнего ключа размещен в той же таблице, что и первичный ключ, и содержит в себе копии значений первичного ключа этой же таблицы.

Мигрировавший в качестве внешнего ключа атрибут не может появиться дважды в одной сущности под одним именем, поэтому должен получить другое имя (имя роли). На рисунке 3.8 показан пример создания рекурсивной связи типа «руководит / подчиняется», когда информация о руководителе (senior manager)

содержится в той же сущности, что и информация о подчиненных (managers). Чтобы показать присутствие руководителя в таблице, создается рекурсивная связь, и в окне создания внешнего ключа в разделе Child определяется новый атрибут для внешнего ключа с именем m_senior_manager_id (при этом создавать индекс для внешнего ключа здесь не нужно).

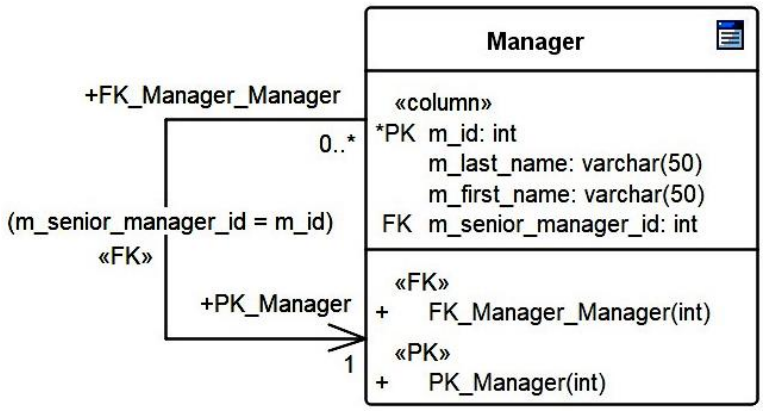


Рисунок 3.8 – Рекурсивная связь

Имя роли (Rolename, функциональное имя) – синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут внешнего ключа в дочерней сущности.

Имя роли в обязательном порядке используется в двух случаях: если используется рекурсивная связь или если два или более атрибута одной сущности имеют одинаковые домены значений, но разный смысл. Например, на рисунке 3.9 отображены сущности, характеризующие процесс обмена валюты, в котором участвуют проданная и купленная валюты.

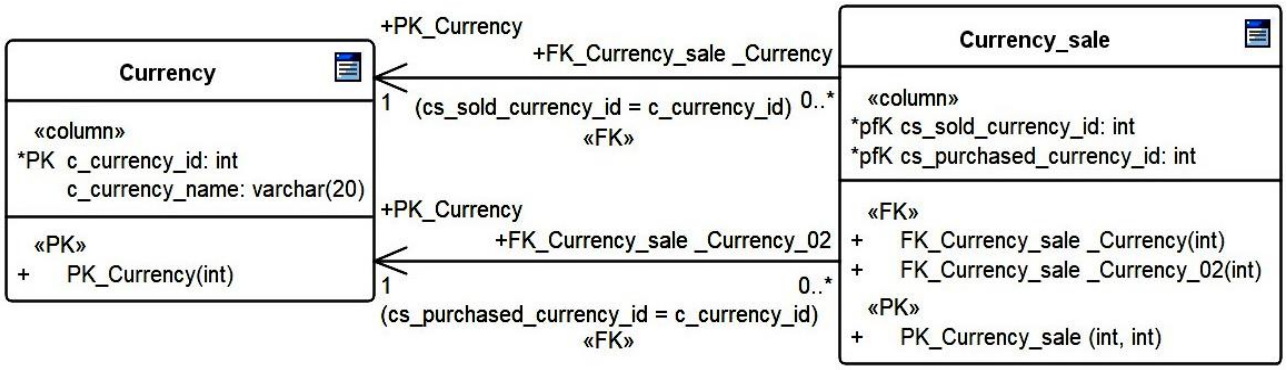


Рисунок 3.9 – Использование атрибутов, имеющих общий домен значений, но разный смысл

Сущности Currency (валюта) и Currency_sale (продажа валюты) должны быть связаны дважды, и первичный ключ сущности Currency дважды мигрирует в качестве внешнего ключа в сущность Currency_sale. Один мигрировавший атрибут будет содержать идентификатор проданной валюты, а другой – идентификатор купленной валюты, т. е. оба атрибута имеют общий домен значений,

но разный смысл. Поэтому эти атрибуты имеют различные имена – sold (проданная) и (purchased).

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) [9, с. 71–77].

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

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.

Консистентность базы данных (database consistency) – свойство реляционной базы данных, заключающееся в строгом соблюдении в любой момент времени всех ограничений, заданных неявно реляционной моделью или явно конкретной схемой базы данных. Консистентность контролирует СУБД (контролирует типы данных, ограничения на значения столбцов, ссылочную целостность, корректность выполнения триггеров, корректность запросов и т. д.).

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 отделах библиотеки: шифр книги, отметка о списании, отметка о замене;

– Replacement_of_copies – хранит номера актов замены книг;

– Lecturer – для хранения информации о преподавателях-пользователях библиотеки: читательский номер, фамилия, имя, отчество, кафедра, должность;

– Decommissioned_book – хранит информацию о протоколах списания книг: шифр книги, табельный номер списавшего книгу сотрудника библиотеки, причина списания, номер протокола списания;

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

– Order_of_book – заказ преподавателей новой литературы: количество, дата заказа.

Информационная модель БД ИС отображена на рисунке 3.10.

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.

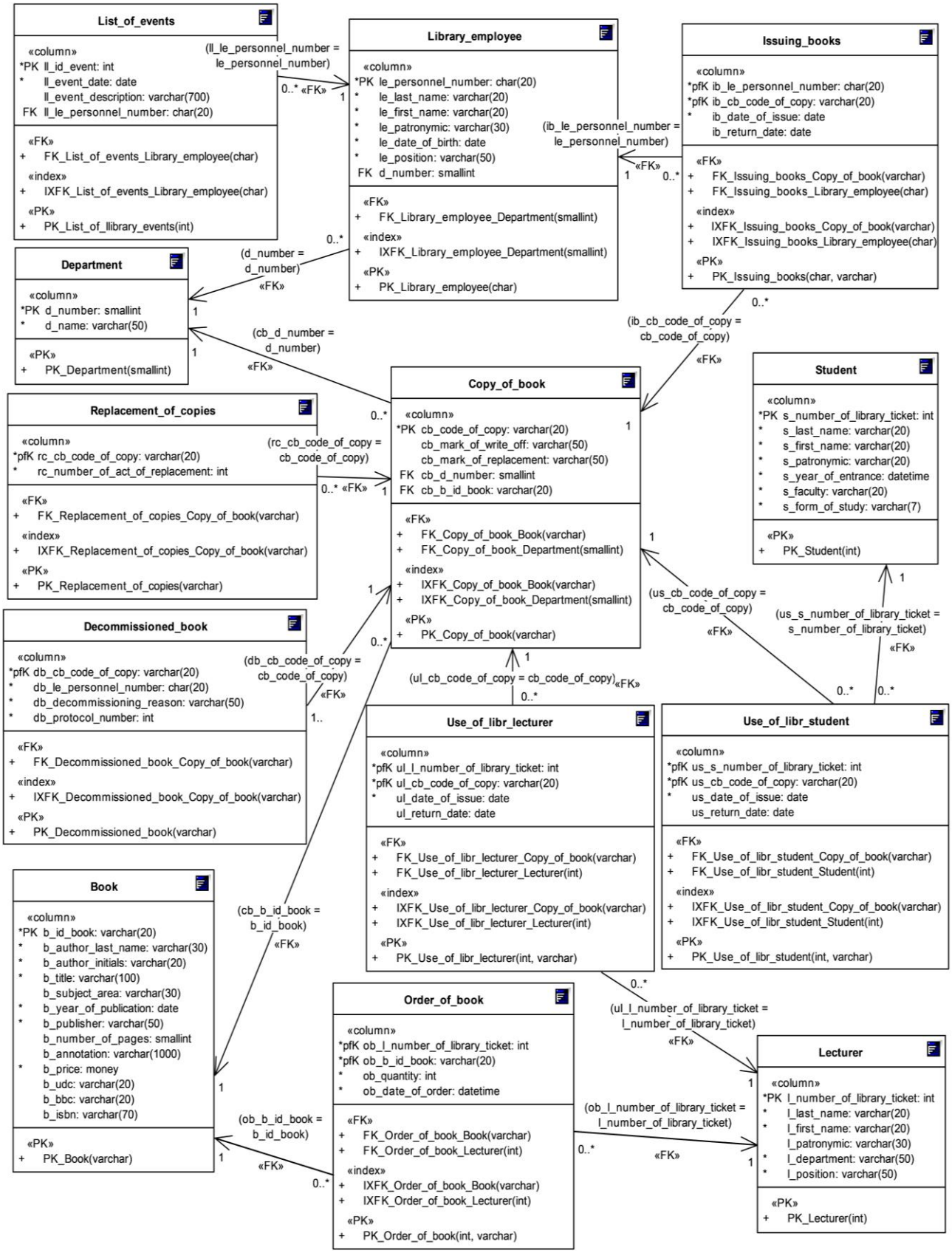


Рисунок 3.10 – Информационная модель

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 Department и Copy_of_book неидентифицирующая обязательная, т. к. каждый экземпляр закреплен за определенным отделом. Тип связи 1:M, т. к. в одном отделе может храниться много экземпляров.

Связь между сущностями Book и Copy_of_book неидентифицирующая обязательная, т. к. каждый экземпляр – это зарегистрированная книга. Тип связи 1:M, т. к. одна книга может быть представлена несколькими экземплярами.

Связь между сущностями Replacement_of_copies и Copy_of_book идентифицирующая, т. к. для замены экземпляров необходима информация о нем. Тип связи 1:M, т. к. замена осуществляется для одного экземпляра.

Связь между сущностями Decommissioned_book и Copy_of_book идентифицирующая, т. к. для списания экземпляров необходима информация о нем. Тип связи 1:1, т. к. списание осуществляется для одного экземпляра.

Связь между сущностями Lecturer и Order_of_book идентифицирующая, т. к. для заказа книг необходима информация о заказчике. Тип связи 1:M, т. к. один преподаватель может заказать много книг.

Связь между сущностями Book и Order_of_book идентифицирующая, т. к. для заказа книг необходима информация о заказе. Тип связи 1:M, т. к. одна книга может быть во многих заказах.

Связь между сущностями Lecturer и Copy_of_book – M:M, т. к. один преподаватель может пользоваться многими экземплярами, а один экземпляр может быть у многих преподавателей.

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, Use_of_libr_lecturer.

Ссылочная целостность реализована с учетом следующих ограничений предметной области:

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

– при изменении информации о каком-либо сотруднике из таблицы Lecturer в таблице Use_of_libr_lecturer информация будет автоматически изменяться (каскадное обновление);

– 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;

– в таблице Lecturer при изменении данных происходит каскадное обновление. Разрешается удаление информации только в случае, если на данную информацию нет ссылок в других связанных таблицах;

– в таблице Order_of_book разрешается обновление.

Задание

Разработать информационную модель выбранной предметной области. Информационная модель должна быть представлена схемой БД, разработанной с использованием методологии 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 Enterprise Architect с системами управления базами данных (генерация схемы базы данных)

4 часа

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 программной системы

2 часа

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, подпроцессов 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 (таблица 5.1), который 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.

Таблица 5.1 – Отчет о верификации моделей

Имя действия	Имя сущности	Имя атрибута

В таблице 5.2 приведен пример заполнения таблицы отчета о верификации.

Таблица 5.2 – Пример отчета о верификации моделей

Имя действия	Имя сущности	Имя атрибута
Провести занятия со студентами	List_of_events	Il_id_event ll_event_date ll_event_description ll_le_personnel_number
Обработка каталожных карточек	Book	b_id_book b_author_last_name b_author_initials b_title b_subject_area b_year_of_publication b_publisher b_number_of_pages b_annotation b_price b_udc b_bbc b_isbn

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.

2 часа

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:

– 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.

Существуют следующие основные типы объектов базы данных Access:

– **таблица** – совокупность записей, где хранится основная информация;

– **форма** – представляет собой специальный формат экрана, используется для ввода данных в таблицу и просмотра одной записи;

– **запрос** – инструмент для анализа, выбора и изменения данных;

– **отчет** – средство организации данных при выводе на печать;

– **макрос** – программа, состоящая из последовательности макрокоманд, выполняющаяся по вызову или при наступлении некоторого события;

– **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
1	2	3
Короткий текст (Short Text)	Для хранения текста или комбинаций алфавитно-цифровых 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 гигабайт. 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 поле используется как 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. Хранение значений даты и времени в числовом формате позволяет выполнять различные вычисления с этими данными	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, которые могут применяться в качестве первичного ключа	4 байта (16 байтов, если поле используется как GUID)
Поле объекта OLE	Для хранения объектов OLE (изображений, диаграмм) из других программ Microsoft Windows. Поля объекта OLE позволяют вкладывать в одну запись только один файл. Рекомендуется вместо поля объекта OLE использовать поле с типом данных Вложение (Attachment), которое позволяет вкладывать в одну запись несколько файлов и поддерживает больше типов файлов	До 1 гигабайта
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, диаграмм и др.	для сжатых вложений – 2 Гбайт

Окончание таблицы 6.1

1	2	3
Вычисляемый (Calculated)	Для создания вычисляемых полей: числовых, текстовых, денежных, дата/время, логических. Значение определяется выражением, записанным в поле и использующим другие поля текущей записи, некоторые встроенные функции и константы, связанные арифметическими, логическими или строковыми операторами	
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)».

Если перетащить поле, не являющееся ключевым и не имеющее уникального индекса, на другое поле, которое также не является ключевым и не имеет уникального индекса, создается неопределенное отношение. В запросах, содержащих таблицы с неопределенным отношением, Access по умолчанию отображает линию объединения между таблицами, но условия целостности данных при этом не накладываются и нет гарантии уникальности записей в любой из таблиц.

Для связей можно установить следующие свойства:

– «Обеспечение целостности данных» – запрещает помещать в поле связанной таблицы значения, которые отсутствуют в поле главной таблицы;

– «Каскадное обновление связанных полей» – при изменении значения поля в главной таблице будут автоматически меняться соответствующие значения в связанной таблице;

– «Каскадное удаление связанных записей» – если удаляется запись из главной таблицы, должны автоматически удалиться связанные с ней записи из связанной таблицы.

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

4 часа

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 [12]. 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	Способы 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 любой 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

Продолжение таблицы 7.1

1	2	3
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
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

Окончание таблицы 7.1

1	2	3
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?

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 : 28.05.2021.

2 **Агальцов, В. П.** Базы данных [Электронный ресурс]: в 2-х т. Т. 2. Распределенные и удаленные базы данных : учебник / В. П. Агальцов. – Москва : ФОРУМ : ИНФРА-М, 2018. – 271 с. – Режим доступа : URL: https://znanium.com/catalog/product/1068927]. – Дата доступа : 28.05.2021.

3 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: В 2-х кн. Книга 1. Локальные базы данных – Москва: ИД «ФОРУМ» : ИНФРА-М, 2020. – 352 с. : ил. – Режим доступа : URL: https://znanium.com/catalog/product/1068927. – Дата доступа : 28.05.2021.

4 **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.

5 **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 : URL: https://znanium.com/catalog/product/1053934. – Дата доступа : 28.05.2021.

6 **Дадян, Э. Г.** Данные: хранение и обработка [Электронный ресурс]: учебник / Э. Г. Дадян. – Москва : ИНФРА-М, 2020. – 205 с. – Режим доступа : URL: https://znanium.com/catalog/product/1045133. – Дата доступа : 28.05.2021.

7 **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 : URL: https://znanium.com/catalog/product/1058247. – Дата доступа : 28.05.2021.

8 **Кузин, А. В.** Базы данных : учебное пособие для студентов высших учебных заведений / А. В. Кузин, С. В. Левонисова. – 6-е изд., стер. – Москва : Академия, 2016. – 320 с.

9 **Куликов, С. С.** Реляционные базы данных в примерах : практическое пособие для программистов и тестировщиков [Электронный ресурс] / С. С. Куликов. – Минск: Четыре четверти, 2020. – 424 с. – Режим доступа : http://svyatoslav.biz/relational_databases_book/. – Дата доступа : 25.05.2021.

10 **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 : URL: https://znanium.com/catalog/product/1011088. – Дата доступа : 25.05.2021.

11 **Шустова, Л. И.** Базы данных : учебник [Электронный ресурс] / Л. И. Шустова, О. В. Тараканов. – Москва : Инфра-М, 2017. – 304 с. – Режим доступа : www.dx.doi.org/10.12737/11549. – Дата доступа : 29.04.2021.

12 **Бекаревич, Ю. Б.** Самоучитель MS Office Access 2016 / Ю. Б. Бекаревич, Н. В. Пушкина. СПб.: БХВ-Петербург, 2017. – 480 с: ил. – Режим доступа : https://obuchalka.org/20190328108046/samouchitel-ms-office-access-2016-bekarevich-u-b-pushkina-n-v-2017.html. – Дата доступа : 29.04.2021.