Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Хоружий Людмила Ивановна
Должность: Директор института экономики и управления АПК
Дата подписания: 15.07.2023 19:28:57
Уникальный программный ключ:
1e90b132d9b04dce67585160b015dddf2cb1e6a9

УТВЕРЖДАЮ:
Директор института
экономики и управления АПК
_____ Хоружий Л.И.
«__» _____ 2022 г.

Лист актуализации рабочей программы дисциплины
ФТД.02 «Разработка баз данных в СУБД PostgreSQL»

для подготовки бакалавров
Направление: 09.03.03 Прикладная информатика
Направленность: Прикладная информатика в экономике
Форма обучения очная
Год начала подготовки: 2019 г.
Курс 4
Семестр 7

А) В рабочую программу не вносятся изменения. Рабочая программа актуализирована для 2022 г. начала подготовки.
Б) Программа будет распространена при организации учебного процесса на направленность (профиль): Системы искусственного интеллекта.

Разработчики: Никаноров М.С. ст. преп.
Греченева А.В., к.т.н.

«22» 08 2022 г.

Рабочая программа пересмотрена и одобрена на заседании кафедры прикладной информатики протокол № 1 от «29» 08 2022 г.
И.о. заведующий кафедрой Худякова Е.В. д.э.н., профессор

Лист актуализации принят на хранение:

И.о. заведующий выпускающей кафедрой
прикладной информатики
Худякова Е.В. д.э.н., профессор
_____

«29» 08 2022 г.

МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ – МСХА имени К. А. ТИМИРЯЗЕВА»
(ФГБОУ ВО РГАУ - МСХА имени К. А. Тимирязева)

Институт экономики и управления АПК
Кафедра прикладной информатики

УТВЕРЖДАЮ:
И.о. директора института экономики и управления АПК
_____ Л.И. Хоружий
"11" июня 20 19 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ФТД.02 «Разработка баз данных в СУБД PostgreSQL»

для подготовки бакалавров

ФГОС ВО

Направление: 09.03.03 «Прикладная информатика»
Направленности: «Прикладная информатика в экономике»

Курс: 4
Семестры: 7

Форма обучения: очная

Год начала подготовки: 2019

Регистрационный номер _____

Москва, 2019

Разработчик: Худякова Е.В., д.э.н., проф., Ханжиян К.И., ст. преподаватель
«02» 03 2020 г.

Рецензент: Харитонова А.Е., к.э.н.
(ФИО, ученая степень, ученое звание)
(подпись)
«02» 03 2020 г.

Программа составлена в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.03 «Прикладная информатика» и учебного плана по данному направлению.

Программа обсуждена на заседании кафедры прикладной информатики
протокол № 7 от «02» 03 2020 г.

Зав. кафедрой: Худякова Е.В., д.э.н., профессор
(ФИО, ученая степень, ученое звание)
(подпись)
«02» 03 2020 г.

Согласовано:
Председатель учебно-методической
комиссии института: экономики и управления АПК
Корольков А.Ф., к.э.н., доцент
(ФИО, ученая степень, ученое звание)
(подпись)
«15» 05 2020 г.

Заведующий выпускающей кафедрой прикладной информатики
Худякова Е.В., д.э.н., профессор
(ФИО, ученая степень, ученое звание)
(подпись)
«02» 03 2020 г.

Зав. отделом комплектования ЦНБ
(подпись)

Бумажный экземпляр РПД, копии электронных вариантов РПД и оценочных материалов получены:
Методический отдел УМУ
«__» _____20__ г.

СОДЕРЖАНИЕ

АННОТАЦИЯ 4
1. ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ 4
2. МЕСТО ДИСЦИПЛИНЫ В УЧЕБНОМ ПРОЦЕССЕ 4
3. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ 5
4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ 7
4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам 7
4.2 Содержание дисциплины 7
4.3 Лекции/практические занятия 9
5. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ 10
6. ТЕКУЩИЙ КОНТРОЛЬ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ 11
6.1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности 11
6.2. Описание показателей и критериев контроля успеваемости, описание шкал оценивания 23
7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 25
7.1 Основная литература 25
7.2 Дополнительная литература 25
7.3 Нормативные правовые акты 26
8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ 26
9. ПЕРЕЧЕНЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 27
10. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ 27
11. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ СТУДЕНТАМ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ 28
Виды и формы отработки пропущенных занятий 28
12. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПРЕПОДАВАТЕЛЯМ ПО ОРГАНИЗАЦИИ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ 28

Аннотация

рабочей программы учебной дисциплины ФТД.02 «Разработка баз данных в СУБД PostgreSQL» для подготовки бакалавров по направлению 09.03.03 «Прикладная информатика» направленности «Прикладная информатика в экономике»

Цель освоения дисциплины: получение системных знаний о технологиях разработки реляционных баз данных и обеспечение фундаментальной подготовки по использованию современной СУБД PostgreSQL.

Место дисциплины в учебном плане: дисциплина включена в перечень факультативных дисциплин учебного плана по направлению подготовки 09.03.03 «Прикладная информатика», дисциплина осваивается во 7 семестре.

Требования к результатам освоения дисциплины: в результате освоения дисциплины формируются следующие компетенции (индикаторы): ПКос-9.1; ПКос-9.2; ПКос-9.3

Краткое содержание дисциплины:

Введение в PostgreSQL. Основные понятия реляционной модели. Создание рабочей среды, установка СУБД и работа с интерактивным терминалом PostgreSQL. Основные типы данных в СУБД PostgreSQL и их использование. Основы языка SQL применительно к СУБД PostgreSQL. Ограничение целостности данных, построение схем базы данных. Построение запросов разного уровня сложности. Изменение структур таблиц, изменение данных в таблицах. Использование индексов для повышения уровня производительности. Механизм выполнения транзакций.

Общая трудоемкость дисциплины: 36/1 (часы/зач. ед.).

Промежуточный контроль: зачет в 7 семестре.

1. Цель освоения дисциплины

Целью дисциплины является: получение системных знаний о технологиях разработки реляционных баз данных и обеспечение фундаментальной подготовки по использованию современной СУБД PostgreSQL.

Задачи:

- изучить модели и технологию построения баз данных;
- изучить СУБД PostgreSQL и область ее применения;
- получить навыки разработки баз данных в рамках СУБД PostgreSQL;
- получить практические навыки проектирования, администрирования и использования баз данных в СУБД PostgreSQL.

2. Место дисциплины в учебном процессе

Дисциплина «Разработка баз данных в СУБД PostgreSQL» включена в перечень факультативных дисциплин учебного плана. Дисциплина «Разработка баз данных в СУБД PostgreSQL» реализуется в соответствии с требованиями ФГОС ВО, ОПОП ВО и Учебного плана по направлению 09.03.03 «Прикладная информатика».

Данная дисциплина базируется на знаниях, полученных студентами в ходе изучения дисциплин бакалавриата, таких как «Базы данных», «Информационные системы и технологии».

Дисциплина «Разработка баз данных в СУБД PostgreSQL» может быть использована для подготовки бакалавров к ГИА.

Рабочая программа дисциплины «Разработка баз данных в СУБД PostgreSQL» для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается индивидуально с учетом особенностей психофизического развития индивидуальных возможностей и состояния здоровья таких обучающихся.

3. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся компетенций, представленных в таблице 1.

Таблица 1

Требования к результатам освоения учебной дисциплины

№ п/п	Код компетенции	Содержание компетенции (или её части)	Индикаторы компетенций	В результате изучения учебной дисциплины обучающиеся должны:		
				знать	уметь	владеть
1	ПК-9	Способность осуществлять ведение базы данных и поддержку информационного обеспечения решения прикладных задач	ПКос-9.1 Знать модели баз данных и их особенности, современные системы управления базами данных	модели баз данных и их особенности при построении баз данных в СУБД PostgreSQL	-	-
			ПКос-9.2 Уметь создавать информационные базы и их резервные копии, находить и исправлять технические сбои	-	проектировать базы данных в СУБД PostgreSQL, создавать резервные копии, находить и исправлять технические сбои и ошибки работы БД	
			ПКос-9.3 Владеть навыками управления и ведения базы данных и их поддержку для решения прикладных задач			приемами работы с инструментальными средствами для разработки и управления базами данных в PostgreSQL

4. Структура и содержание дисциплины

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам

Общая трудоёмкость дисциплины составляет 1 зач.ед. (36 часов), их распределение по видам работ представлено в таблице 2.

Таблица 2

Распределение трудоёмкости дисциплины по видам работ по семестрам

Вид учебной работы	Трудоёмкость	
	час.	в т.ч. по семестрам
		№ 7
Общая трудоёмкость дисциплины по учебному плану	**36**	**36**
1. Контактная работа:	**18,25**	**18,25**
Аудиторная работа	**18,25**	**18,25**
в том числе:		
лекции (Л)	8	8
практические занятия (ПЗ)	10	10
контактная работа на промежуточном контроле (КРА)	0,25	0,25
2. Самостоятельная работа (СРС)	**17,75**	**17,75**
самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к практическим занятиям, тестированию и т.д.)	8,75	8,75
подготовка к зачету	9	9
Вид промежуточного контроля:		Зачет

4.2 Содержание дисциплины

Таблица 3

Тематический план учебной дисциплины

Наименование разделов и тем дисциплин (укрупнённо)	Всего	Аудиторная работа			Внеаудиторная работа СР
		Л	ПЗ	ПКР	
Раздел 1 Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL	11	3	4	-	4
Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL	15,75	5	6	-	4,75
Подготовка к зачету	9	-	-	-	9
Контактная работа на промежуточном контроле (КРА)	0,25	-	-	0,25	-
Итого по дисциплине	**36**	**8**	**10**	**0,25**	**17,75**

Раздел 1 Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL

Тема 1.1 Модели данных. Основные понятия реляционной модель данных. Средства концептуального моделирования

Основные модели данных, их применение. Модель данных «сущность-связь». Описание реляционной модели данных: домены, отношения, атрибуты и кортежи.

Тема 1.2 Введение в СУБД PostgreSQL. Создание рабочей среды, установка PostgreSQL

История развития СУБД PostgreSQL. Особенности использования. Создание рабочей среды на локальном компьютере, установка полной версии СУБД PostgreSQL (сервер и клиентские программы).

Тема 1.3 Основные операции с таблицами в СУБД PostgreSQL

Подключение к базе данных. Использование утилиты psql. Работа с командами SQL и psql. Создание таблиц, удаление таблиц, ввод данных. Изменение данных.

Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL

Тема 2.1 Типы данных СУБД PostgreSQL. Основы языка определения данных

Набор встроенных типов данных. Числовые типы: целочисленные типы, числа фиксированной точности, типы данных с плавающей точкой, последовательные типы (serial). Символьные (строковые) типы. Типы «дата/время». Логический тип. Массивы. Типы JSON. Значения по умолчанию и ограничения целостности. Назначение первичных и внешних ключей. Модификация таблиц (ALTER TABLE). Создание представлений (CREATE VIEW).

Тема 2.2 Создание запросов. Подзапросы. Изменение данных в таблицах

Дополнительные возможности команды SELECT. Использование основных операторов в извлечение данных (диапазоны, вычисляемые столбцы, упорядочивание данных и др). Операции соединение (join) таблиц. Объединения множеств строк — UNION. Скалярные подзапросы. Подзапросы в предикате IN. Некоррелированные подзапросы. Вложенные подзапросы. Вставка строк в таблицы. Обновление строк в таблицах.

Тема 2.3 Общее понятие индексов, разновидности. Работа с транзакциями. Примеры использования

Описание и использование индексов. Создание индексов по нескольким столбцам. Индексы на основе выражений. Частичные индексы. Обработка транзакций.

Тема 2.4 Повышение производительности работы БД

Методы формирования соединения наборов строк. Управление планировщиком. Оптимизация запросов.

4.3 Лекции/практические занятия

Таблица 4

Содержание лекций/ практических занятий и контрольные мероприятия

№ п/п	№ раздела	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
1.	**Раздел 1** Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL				7
	Тема 1 Модели данных. Основные понятия реляционной модель данных. Средства концептуального моделирования	**Лекция №1** Модели данных. Основные понятия реляционной модель данных.	ПКос-9.1	-	1,5
		Практическое занятие №1 Средства концептуального моделирования		Защита практической работы	1
	Тема 2 Введение в СУБД PostgreSQL. Создание рабочей среды, установка PostgreSQL	**Лекция №2** Введение в СУБД PostgreSQL.	ПКос-9.2 ПКос-9.3	-	1,5
		Практическое занятие №2 Создание рабочей среды, установка PostgreSQL		Защита практической работы	1
	Тема 3 Основные операции с таблицами в СУБД PostgreSQL	**Практическое занятие №3** Основные операции с таблицами в СУБД PostgreSQL		Защита практической работы	2
2.	**Раздел 2** Проектирование и управление базами данных в СУБД PostgreSQL				**11**
	Тема 1 Типы данных СУБД PostgreSQL. Основы языка определения данных	**Лекция №3** Типы данных СУБД PostgreSQL. Основы языка определения данных	ПКос-9.1 ПКос-9.2 ПКос-9.3	-	1
	Тема 2 Создание запросов. Подзапросы. Изменение данных в таблицах	**Лекция №4** Создание запросов. Подзапросы. Изменение данных в таблицах		-	1,5
		Практическое занятие №4 Создание запросов. Подзапросы. Изменение данных в таблицах		Защита практической работы	3
	Тема 3 Общее понятие индексов, разновидности. Работа с транзакциями. Примеры использования	**Лекция №5** Общее понятие индексов, разновидности. Работа с транзакциями.	ПКос-9.2 ПКос-9.3	-	1,5
		Практическое занятие №5 Использование индексов		Защита практической работы	3

№ п/п	№ раздела	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
	Тема 4 Повышение производительности работы БД	**Лекция №6** Повышение производительности работы БД	ПКос-9.2 ПКос-9.3	-	1

Таблица 5

Перечень вопросов для самостоятельного изучения дисциплины

№ п/п	№ раздела и темы	Перечень рассматриваемых вопросов для самостоятельного изучения	Формируемые компетенции (индикаторы)
1	**Раздел 1** Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL		
	Тема 1 Модели данных. Основные понятия реляционной модель данных. Средства концептуального моделирования	Слабоструктурированные модели данных. Модели для представления знаний	ПКос-9.1
	Тема 2 Введение в СУБД PostgreSQL. Создание рабочей среды, установка PostgreSQL	Программа psql — интерактивный терминал PostgreSQ	ПКос-9.2
2	**Раздел 2** Проектирование и управление базами данных в СУБД PostgreSQL		
	Тема 2 Создание запросов. Подзапросы. Изменение данных в таблицах	Теоретико-множественные операции	ПКос-9.2 ПКос-9.3
	Тема 4 Повышение производительности работы БД	Репликация баз данных	ПКос-9.2 ПКос-9.3

5. Образовательные технологии

Таблица 6

Применение активных и интерактивных образовательных технологий

№ п/п	Тема и форма занятия		Наименование используемых активных и интерактивных образовательных технологий
1.	Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL Тема 2 Создание запросов. Подзапросы. Изменение данных в таблицах	ПЗ	Выполнение практических заданий на ПК Мастер-классы
2.	Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL	ПЗ	Выполнение практических заданий на ПК Мастер-классы

№ п/п	Тема и форма занятия		Наименование используемых активных и интерактивных образовательных технологий
	Тема 3 Общее понятие индексов, разновидности. Работа с транзакциями. Примеры использования		

6. Текущий контроль успеваемости и промежуточная аттестация по итогам освоения дисциплины

6.1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

1). Типовые практические работы

Тема «Модели данных. Основные понятия реляционной модель данных. Средства концептуального моделирования»

Построить ER-модель и логическую модель данных для описания предметной области «Пассажирские авиаперевозки».

Российская авиакомпания выполняет пассажирские авиаперевозки. Она обладает своим парком самолетов различных моделей. Каждая модель самолета имеет определенный код, который присваивает Международная ассоциация авиаперевозчиков (IATA). При этом будем считать, что самолеты одной модели имеют одинаковые компоновки салонов, т. е. порядок размещения кресел и нумерацию мест в салонах бизнес-класса и экономического класса. Например, если это модель Sukhoi SuperJet-100, то место 2A относится к бизнес-классу, а место 20D — к экономическому классу. Бизнес-класс и экономический класс — это разновидности так называемого класса обслуживания.

Наша авиакомпания выполняет полеты между аэропортами России. Каждому аэропорту присвоен уникальный трехбуквенный код, при этом используются только заглавные буквы латинского алфавита. Эти коды присваивает не сама авиакомпания, а специальные организации, управляющие пассажирскими авиаперевозками. Зачастую название аэропорта не совпадает с названием того города, которому этот аэропорт принадлежит. Например, в городе Новосибирске аэропорт называется Толмачево, в городе Екатеринбурге — Кольцово, а в Санкт-Петербурге — Пулково. К тому же некоторые города имеют более одного аэропорта. Добавим еще одну важную деталь: каждый аэропорт характеризуется географическими координатами — долготой и широтой, а также часовым поясом.

Формируются маршруты перелетов между городами. Конечно, каждый такой маршрут требует указания не только города, но и аэропорта, поскольку, как мы уже сказали, в городе может быть и более одного аэропорта. В качестве упрощения реальности мы решим, что маршруты не будут иметь промежуточных посадок, т. е. у них будет только аэропорт отправления и аэропорт назначения. Каждый маршрут имеет шестизначный номер, включающий цифры и буквы латинского алфавита.

На основе перечня маршрутов формируется расписание полетов (или рейсов). В расписании указывается плановое время отправления и плановое время прибытия, а также тип самолета, выполняющего этот рейс.

При фактическом выполнении рейса возникает необходимость в учете дополнительных сведений, а именно: фактического времени отправления и фактического времени прибытия, а также статуса рейса. Статус рейса может принимать ряд значений:

– Scheduled (за месяц открывается возможность бронирования);
– On Time (за сутки открывается регистрация);
– Delayed (рейс задержан);

– Departed (вылетел);
– Arrived (прибыл);
– Cancelled (отменен).

Теперь обратимся к пассажирам. Полет начинается с бронирования авиабилета. В настоящее время общепринятой практикой является оформление электронных билетов. Каждый такой билет имеет уникальный номер, состоящий из 13 цифр. В рамках одной процедуры бронирования может быть оформлено несколько билетов, но каждая такая процедура имеет уникальный шестизначный номер (шифр) бронирования, состоящий из заглавных букв латинского алфавита и цифр. Кроме того, для каждой процедуры бронирования записывается дата бронирования и рассчитывается общая стоимость оформленных билетов.

В каждый билет, кроме его тринадцатизначного номера, записывается идентификатор пассажира, а также его имя и фамилия (в латинской транскрипции) и контактные данные. В качестве идентификатора пассажира используется номер документа, удостоверяющего личность. Конечно, пассажир может сменить свой документ, а иной раз даже фамилию и имя, за время, прошедшее между бронированием билетов в разные дни, поэтому невозможно наверняка сказать, что какие-то конкретные билеты были оформлены на одного и того же пассажира.

В каждый электронный билет может быть вписано более одного перелета. Специалисты называют эти записи о перелетах сегментами. В качестве примера наличия нескольких сегментов можно привести такой: Красноярск — Москва, Москва — Анапа, Анапа — Москва, Москва — Красноярск. При этом возможно в рамках одного бронирования оформить несколько билетов на различных пассажиров. Для каждого перелета указывается номер рейса, аэропорты отправления и назначения, время вылета и время прибытия, а также стоимость перелета. Кроме того, указывается и так называемый класс обслуживания: экономический, бизнес и др.

Когда пассажир прибывает в аэропорт отправления и проходит регистрацию билета, оформляется так называемый посадочный талон. Этот талон связан с авиабилетом: в талоне указывается такой же номер, который имеет электронный авиабилет данного пассажира. Кроме того, в талоне указывается номер рейса и номер места в самолете. Указывается также и номер посадочного талона — последовательный номер, присваиваемый в процессе регистрации билетов на данный рейс.

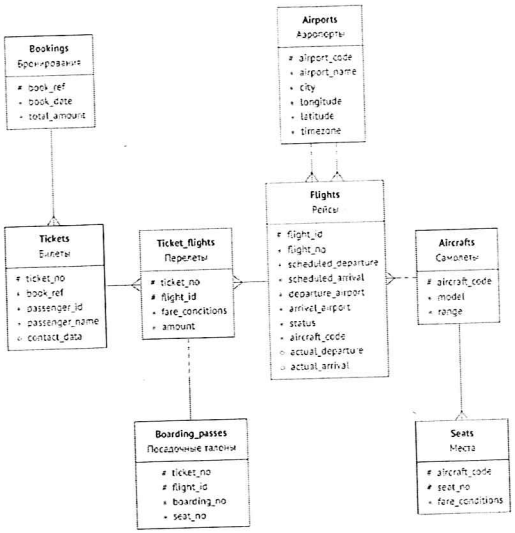
Напомним, что каждому креслу в салоне самолета соответствует конкретный класс обслуживания. Данная информация учитывается при регистрации билетов и оформлении посадочных талонов. Если, например, пассажир приобрел билет с экономическим классом обслуживания, то в его посадочном талоне будет указан номер места в салоне экономического класса, но не в салоне бизнес-класса.

Тема «Введение в СУБД PostgreSQL. Создание рабочей среды, установка PostgreSQL»

1. Выполните процедуру установки СУБД PostgreSQL в среде выбранной вами операционной системы.
2. Ознакомьтесь с утилитой psql с помощью встроенной справки, а также с помощью справки, вызываемой по команде psql –help
3. Кроме утилиты psql существуют и другие универсальные программы для работы с сервером баз данных PostgreSQL, например, pgAdmin. Это мощная утилита с графическим интерфейсом.
   Самостоятельно установите программу pgAdmin и изучите основные приемы работы с ней.
4. Разверните учебную базу данных. Попробуйте подключиться к ней с помощью утилиты psql. Для выхода из утилиты используйте команду \q.

Тема «Основные операции с таблицами в СУБД PostgreSQL»

1. Запустите утилиту psql и подключитесь к базе данных с учетной записью пользователя postgres: psql -d demo -U postgres
2. Создайте таблицы БД «Авиаперевозки», представленные на схеме данных используя команду CREATE TABLE.



3. Заполните БД данными.
4. Изучите и проведите ряд операций по удалению таблиц, изменению данных, приведенных в документации к PostgreSQL.

Тема «Создание запросов. Подзапросы. Изменение данных в таблицах»

1. В документации сказано, что служебный символ «%» в шаблоне оператора LIKE соответствует любой последовательности символов, в том числе и пустой последовательности, однако ничего не сказано насчет правил обработки пробелов.
В таблице «Билеты» (tickets) столбец passenger_name содержит имя и фамилию пассажира, записанные заглавными латинскими буквами и разделенные одним пробелом.
Выясните правила обработки пробелов самостоятельно, выполнив следующие команды и сравнив полученные результаты:

```
SELECT count( * ) FROM tickets;
SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% %';
SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% % %';
SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% %%';
```

2. Этот запрос выбирает из таблицы «Билеты» (tickets) всех пассажиров с именами, состоящими из трех букв (в шаблоне присутствуют три символа «_»):

```
SELECT passenger_name
FROM tickets
WHERE passenger_name LIKE '___ %';
```


Предложите шаблон поиска в операторе LIKE для выбора из этой таблицы всех пассажиров с фамилиями, состоящими из пяти букв.
3. В разделе документации 9.7.2 «Регулярные выражения SIMILAR TO» рассматривается оператор SIMILAR TO. Он работает аналогично оператору LIKE, но использует шаблоны, соответствующие определению регулярных выражений, приведенному в стандарте SQL. Регулярные выражения SQL представляют собой комбинацию синтаксиса LIKE с синтаксисом обычных регулярных выражений. Самостоятельно ознакомьтесь с оператором SIMILAR TO.
4. В разделе документации 9.2 «Функция и операторы сравнения» представлены различные предикаты сравнения, кроме предиката BETWEEN, рассмотренного в этой главе. Самостоятельно ознакомьтесь с ними.
5. В разделе документации 9.17 «Условные выражения» представлены условные выражения, которые поддерживаются в PostgreSQL. В тексте главы была рассмотрена конструкция CASE. Самостоятельно ознакомьтесь с функциями COALESCE, NULLIF, GREATEST и LEAST.
6. Выясните, на каких маршрутах используются самолеты компании Boeing. В выборке вместо кода модели должно выводиться ее наименование, например, вместо кода 733 должно быть Boeing 737-300.
Указание: можно воспользоваться соединением представления «Маршруты» (routes) и таблицы «Самолеты» (aircrafts).
7. Самые крупные самолеты в нашей авиакомпании — это Boeing 777-300. Выяснить, между какими парами городов они летают, поможет запрос:

```
SELECT DISTINCT departure_city, arrival_city
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code
WHERE a.model = 'Boeing 777-300'
ORDER BY 1;
```

```
departure_city | arrival_city
---------------+--------------
Екатеринбург | Москва
Москва | Екатеринбург
Москва | Новосибирск
Москва | Пермь
Москва | Сочи
Новосибирск | Москва
Пермь | Москва
Сочи | Москва
(8 строк)
```

К сожалению, в этой выборке информация дублируется. Пары городов приведены по два раза: для рейса «туда» и для рейса «обратно». Модифицируйте запрос таким образом, чтобы каждая пара городов была выведена только один раз:

```
departure_city | arrival_city
---------------+--------------
Москва | Екатеринбург
Новосибирск | Москва
Пермь | Москва
Сочи | Москва
(4 строки)
```

8. Мы рассматривали различные примеры использования левого и правого внешних соединений: LEFT OUTER JOIN и RIGHT OUTER JOIN. Напишите запрос, в котором использовалось бы полное внешнее соединение — FULL OUTER JOIN.

9. Для ответа на вопрос, сколько рейсов выполняется из Москвы в Санкт-Петербург, можно написать совсем простой запрос:

```
SELECT count( * )
FROM routes
WHERE departure_city = 'Москва'
AND arrival_city = 'Санкт-Петербург';
count
-------
12
(1 строка)
```

А с помощью какого запроса можно получить результат в таком виде?

```
departure_city | arrival_city | count
---------------+--------------+-------
Москва | Санкт-Петербург | 12
(1 строка)
```

10. Выяснить, сколько различных рейсов выполняется из каждого города, без учета частоты рейсов в неделю, можно с помощью обращения к представлению «Маршруты» (routes):

```
SELECT departure_city, count( * )
FROM routes
GROUP BY departure_city
ORDER BY count DESC;

departure_city | count
---------------------+-------
Москва | 154
Санкт-Петербург | 35
Новосибирск | 19
...
Благовещенск | 1
Братск | 1
(101 строка)
```

Модифицируйте этот запрос так, чтобы он выводил число направлений, по которым летают самолеты из каждого города. Например, из Москвы в Санкт-Петербург летает несколько различных рейсов, но все эти рейсы относятся к одному направлению. Указание: нужно передать параметр в функцию count.

11. В материализованном представлении «Маршруты» (routes) имеется столбец days_of_week, который содержит списки (массивы) номеров дней недели, когда выполняется каждый рейс.

Для оптимизации расписания вылетов из Москвы нужно выявить пять городов, в которые из столицы отправляется наибольшее число ежедневных рейсов (маршрутов). Строки в выборке следует расположить в убывающем порядке числа выполняемых рейсов. Указание: воспользуйтесь функцией array_length.

12.* Предположим, что служба материального снабжения нашей авиакомпании запросила информацию о числе рейсов, выполняющихся из Москвы в каждый день недели.

Результат можно получить путем выполнения семи аналогичных запросов: по одному для каждого дня недели. Начнем с понедельника:

```
SELECT 'Понедельник' AS day_of_week, count( * ) AS num_flights
FROM routes
WHERE departure_city = 'Москва'
AND days_of_week @> '{ 1 }'::integer[];
```


В этом запросе используется оператор @>, который проверяет, содержатся ли все элементы массива, стоящего справа от него, в том массиве, который находится слева. В правом массиве всего один элемент — номер интересующего нас дня недели.

```
day_of_week | num_flights
------------+------------
Понедельник | 131
(1 строка)
```

Запрос для вторника отличается лишь номером дня недели в массиве.

```
SELECT 'Вторник' AS day_of_week, count( * ) AS num_flights
FROM routes
WHERE departure_city = 'Москва'
AND days_of_week @> '{ 2 }'::integer[];

day_of_week | num_flights
------------+------------
Вторник | 134
(1 строка)
```

Нужно выполнить еще пять аналогичных команд, чтобы получить результаты для всех дней недели. Очевидно, что это нерациональный способ. Получить требуемый результат можно с помощью одного запроса:

```
SELECT unnest( days_of_week ) AS day_of_week,
count( * ) AS num_flights
FROM routes
WHERE departure_city = 'Москва'
GROUP BY day_of_week
ORDER BY day_of_week;

day_of_week | num_flights
------------+------------
1 | 131
2 | 134
3 | 126
4 | 136
5 | 124
6 | 133
7 | 124
(7 строк)
```

Задание 1. Самостоятельно разберитесь, как работает приведенный запрос. Выясните, что делает функция unnest. Для того чтобы найти ее описание, можно воспользоваться теми разделами документации, которые были указаны в главе 4. Однако можно воспользоваться и предметным указателем (Index), ссылка на который находится в самом низу оглавления документации. В качестве вспомогательного запроса, проясняющего работу функции unnest, можно выполнить следующий:

```
SELECT flight_no, unnest( days_of_week ) AS day_of_week
FROM routes
WHERE departure_city = 'Москва'
ORDER BY flight_no;
```

Задание 2. Использование номеров дней недели в предыдущей выборке не должно вызывать затруднений. Но все-таки предположим, что нас попросили модифицировать запрос, чтобы результат выводился в таком виде:

```
name_of_day | num_flights
------------+------------
Пн. | 131
Вт. | 134
Ср. | 126
Чт. | 136
Пт. | 124
Сб. | 133
Вс. | 124
(7 строк)
```

Покажем одно из возможных решений задачи. Оно основано на использовании специальной табличной функции unnest в предложении FROM. Подробно об этом написано в документации в разделе 7.2.1.4 «Табличные функции». Функция может принимать любое число параметров-массивов, а возвращает набор строк, которые могут использоваться в запросах как обычные таблицы. В этих наборах строк столбцы формируются из значений, содержащихся в массивах.

```
SELECT dw.name_of_day, count( * ) AS num_flights
FROM (
SELECT unnest( days_of_week ) AS num_of_day
FROM routes
WHERE departure_city = 'Москва'
) AS r,
unnest( '{ 1, 2, 3, 4, 5, 6, 7 }'::integer[],
'{ "Пн.", "Вт.", "Ср.", "Чт.", "Пт.", "Сб.", "Вс."}'::text[]
) AS dw( num_of_day, name_of_day )
WHERE r.num_of_day = dw.num_of_day
GROUP BY r.num_of_day, dw.name_of_day
ORDER BY r.num_of_day;
```

Этот запрос можно упростить. Предложение WITH ORDINALITY позволяет в нашем примере избавиться от массива целых чисел, обозначающих дни недели, поскольку автоматически формируется столбец целых чисел, нумерующих строки результирующего набора. По умолчанию этот столбец называется ordinality. Это имя можно использовать в запросе. Самостоятельно модифицируйте запрос с применением предложения WITH ORDINALITY.

13. Ответить на вопрос о том, каковы максимальные и минимальные цены билетов на все направления, может такой запрос:

```
SELECT f.departure_city, f.arrival_city,
max( tf.amount ), min( tf.amount )
FROM flights_v f
JOIN ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

```
departure_city | arrival_city | max | min
---------------+--------------+-----+-----
Абакан | Москва | 101000.00 | 33700.00
Абакан | Новосибирск | 5800.00 | 5800.00
Абакан | Томск | 4900.00 | 4900.00
Анадырь | Москва | 185300.00 | 61800.00
Анадырь | Хабаровск | 92200.00 | 30700.00
...
```



```
Якутск | Мирный | 8900.00 | 8100.00
Якутск | Санкт-Петербург | 145300.00 | 48400.00
(367 строк)
```

14. Предположим, что маркетологи нашей авиакомпании хотят знать, как часто встречаются различные имена среди пассажиров? Получить распределение частот имен пассажиров в таблице «Билеты» (tickets) поможет такой запрос:

```
SELECT left( passenger_name, strpos( passenger_name, ' ' ) - 1 )
AS firstname, count( * )
FROM tickets
GROUP BY 1
ORDER BY 2 DESC;
```

```
firstname | count
----------+-------
ALEKSANDR | 20328
SERGEY | 15133
VLADIMIR | 12806
TATYANA | 12058
ELENA | 11291
OLGA | 9998
...
MAGOMED | 14
ASKAR | 13
RASUL | 11
(363 строки)
```

Напишите запрос для ответа на аналогичный вопрос насчет распределения частот фамилий пассажиров.

Подробные сведения о других функциях для работы со строковыми данными приведены в документации в разделе 9.4 «Строковые функции и операторы».

15.* Самостоятельно прочитайте разделы документации, которые рекомендуется изучить для более детального ознакомления с этим классом функций.

Подумайте, в какой ситуации, связанной с базой данных «Авиаперевозки», было бы полезно применить оконные функции, и напишите запрос.

16.* Вместе с агрегатными функциями может использоваться предложение FILTER. Самостоятельно ознакомьтесь с этой темой, обратившись к разделу документации 4.2.7 «Агрегатные выражения». Напишите запрос с использованием предложения FILTER с агрегатной функцией.

17.* Изучите пример использования рекурсивного алгоритма в общем табличном выражении:

```
WITH RECURSIVE ranges ( min_sum, max_sum )
AS (
VALUES( 0, 100000 ),
( 100000, 200000 ),
( 200000, 300000 )
UNION ALL
SELECT min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
)
SELECT * FROM ranges;
```

```
min_sum | max_sum
---------+---------
0 | 100000 исходные строки
100000 | 200000
200000 | 300000
100000 | 200000 результат первой итерации
200000 | 300000
300000 | 400000
200000 | 300000 результат второй итерации
300000 | 400000
400000 | 500000
300000 | 400000
400000 | 500000
500000 | 600000
...
1000000 | 1100000 результат (n-3)-й итерации
1100000 | 1200000
1200000 | 1300000
1100000 | 1200000 результат (n-2)-й итерации
1200000 | 1300000
1200000 | 1300000 результат (n-1)-й итерации (предпоследней)
(36 строк)
```

Здесь мы с помощью предложения VALUES специально создали виртуальную таблицу из трех строк, хотя для получения требуемого результата достаточно только одной строки (0, 100000). Еще важно то, что предложение UNION ALL не удаляет строки-дубликаты, поэтому мы можем видеть весь рекурсивный процесс порождения новых строк.

При рекурсивном выполнении запроса

```
SELECT min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
```

каждый раз выполняется проверка в условии WHERE. И на (n−2)-й итерации это условие отсеивает одну строку, т. к. после (n − 3)-й итерации значение атрибута max_sum в третьей строке было равно 1 300 000.

Ведь запрос

```
SELECT max( total_amount ) FROM bookings;
```

выдаст значение

```
max
------------
1204500.00
(1 строка)
```

Таким образом, после (n − 2)-й итерации во временной области остается всего две строки, после (n−1)-й итерации во временной области остается только одна строка.

Заключительная итерация уже не добавляет строк в результирующую таблицу, поскольку единственная строка, поданная на вход команде SELECT, будет отклонена условием WHERE. Работа алгоритма завершается.

Задание 1. Модифицируйте запрос, добавив в него столбец level (можно назвать его и iteration). Этот столбец должен содержать номер текущей итерации, поэтому нужно увеличивать его значение на единицу на каждом шаге. Не забудьте задать начальное значение для добавленного столбца в предложении VALUES.

Задание 2. Для завершения экспериментов замените UNION ALL на UNION и выполните запрос. Сравните этот результат с предыдущим, когда мы использовали UNION ALL.

Тема «Общее понятие индексов, разновидности. Работа с транзакциями. Примеры использования»

Предположим, что для какой-то таблицы создан уникальный индекс по двум столбцам: column1 и column2. В таблице есть строка, у которой значение атрибута column1 равно ABC, а значение атрибута column2 — NULL. Мы решили добавить в таблицу еще одну строку с такими же значениями ключевых атрибутов, т. е. column1 — ABC, а column2 — NULL.

Как вы думаете, будет ли операция вставки новой строки успешной или завершится с ошибкой? Объясните ваше решение.

2. В тексте главы шла речь о выполнении одной и той же выборки из таблицы «Билеты» (tickets) при наличии индекса по столбцу passenger_name и при его отсутствии. Вы видели, что наличие индекса ускоряет выполнение запроса почти на порядок. Если секундомер в утилите psql выключен, то включите его с помощью команды \timing on

Проведите следующий эксперимент: выполните этот запрос несколько раз подряд при отсутствии индекса, а затем создайте индекс и опять выполните этот запрос несколько раз подряд.

```
SELECT count( * )
FROM tickets
WHERE passenger_name = 'IVAN IVANOV';
```

Вы увидите, что время выполнения повторных запросов к таблице сокращается, причем, когда создан индекс, оно сокращается на порядок. Как вы думаете, почему?

3. Известно, что индекс значительно ускоряет работу, если при выполнении запроса из таблицы отбирается лишь небольшая часть строк. Если же эта доля велика, скажем, половина строк или более, то большого положительного эффекта от наличия индекса уже не будет, а возможно даже, что не будет практически никакого эффекта. Наша задача — проверить это утверждение на практике.

Обратимся к таблице «Перелеты» (ticket_flights). В ней имеется столбец «Класс обслуживания» (fare_conditions), который отличается от остальных тем, что в нем могут присутствовать лишь три различных значения: Comfort, Business и Economy. Если секундомер в утилите psql выключен, то включите его.

Выполните запросы, подсчитывающие количество строк, в которых атрибут fare_conditions принимает одно из трех возможных значений. Каждый из запросов выполните три-четыре раза, поскольку время может немного изменяться, и подсчитайте среднее время. Обратите внимание на число строк, которые возвращает функция count для каждого значения атрибута. При этом среднее время выполнения запросов для трех различных значений атрибута fare_conditions будет различаться незначительно, поскольку в каждом случае СУБД просматривает все строки таблицы.

```
SELECT count( * )
FROM ticket_flights
WHERE fare_conditions = 'Comfort';
SELECT count( * )
FROM ticket_flights
WHERE fare_conditions = 'Business';


SELECT count( * )
FROM ticket_flights
```

```
WHERE fare_conditions = 'Economy';
```

Создайте индекс по столбцу fare_conditions. Конечно, в реальной ситуации такой индекс вряд ли целесообразен, но нам он нужен для экспериментов.

Проделайте те же эксперименты с таблицей ticket_flights. Будет ли различаться среднее время выполнения запросов для различных значений атрибута fare_conditions? Почему это имеет место?

В завершение этого упражнения отметим, что в случае ошибки планировщика при использовании индекса возможно не только отсутствие положительного эффекта, но и значительный отрицательный эффект.

4. Для одной из таблиц создайте индекс по двум столбцам, причем по одному из них укажите убывающий порядок значений столбца, а по другому — возрастающий. Значения NULL у первого столбца должны располагаться в начале, а у второго — в конце. Посмотрите полученный индекс с помощью команд psql

\d имя_таблицы

\di+ имя_индекса

Обратите внимание, что первая команда выведет не только имя индекса, но также и имена столбцов, по которым он создан, а вторая команда выведет размер индекса.

Подберите запросы, в которых созданный индекс предположительно должен использоваться, а также запросы, в которых он использоваться, по вашему мнению, не будет. Проверьте ваши гипотезы, выполнив запросы. Объясните полученные результаты.

5. В сложных базах данных целесообразно использование комбинаций индексов. Иногда бывают более полезны комбинированные индексы по нескольким столбцам, чем отдельные индексы по единичным столбцам. В реальных ситуациях часто приходится делать выбор, т. е. находить компромисс, между, например, созданием двух индексов по каждому из двух столбцов таблицы либо созданием одного индекса по двум столбцам этой таблицы, либо созданием всех трех индексов. Выбор зависит от того, запросы какого вида будут выполняться чаще всего. Предложите какую-нибудь таблицу в базе данных «Авиаперевозки» и смоделируйте ситуации, в которых вы приняли бы одно из этих трех возможных решений. Воспользуйтесь документацией на PostgreSQL.

6. Предложите какую-нибудь таблицу в базе данных «Авиаперевозки» и смоделируйте ситуацию, в которой было бы целесообразно использование индекса на основе функции или скалярного выражения от двух или более столбцов.

7.* В разделе документации 5.3.5 «Внешние ключи» говорится о том, что в некоторых ситуациях бывает целесообразно создавать индекс по столбцам внешнего ключа ссылающейся таблицы. Это позволит ускорить выполнение операций

DELETE и UPDATE над главной (ссылочной) таблицей.

Подумайте, есть ли такие таблицы в базе данных «Авиаперевозки», в отношении которых было бы целесообразно поступить так, как говорится в документации.

8.* В тексте главы был показан пример использования частичного индекса для таблицы «Бронирования». Для его создания мы выполняли команду

```
CREATE INDEX bookings_book_date_part_key
ON bookings ( book_date )
WHERE total_amount > 1000000;
```


Проведите эксперимент с целью сравнения эффекта от создания частичного индекса с эффектом от создания обычного индекса по столбцу total_amount. Для

этого удалите частичный индекс, а затем создайте обычный индекс.

```
DROP INDEX bookings_book_date_part_key;
CREATE INDEX bookings_total_amount_key
ON bookings ( total_amount );
```

Теперь выполните тот же запрос к таблице bookings:

```
SELECT *
FROM bookings
WHERE total_amount > 1000000
ORDER BY book_date DESC;
```

Сравните время выполнения с тем временем, которое было получено при использовании частичного индекса. Очень вероятно, что различия времени выполнения запроса будут незначительными.

Самостоятельно ознакомьтесь с разделом документации 11.8 «Частичные индексы» и попробуйте смоделировать ситуацию в предметной области «Авиаперевозки», когда частичный индекс дал бы больший эффект, чем обычный индекс.

9. Когда выполняются запросы с поиском по шаблону LIKE или регулярными выражениями POSIX, тогда для того, чтобы использовался индекс, нужно предусмотреть следующее. Если параметры локализации системы отличаются от стандартной настройки «C» (например, «ru_RU.UTF-8»), тогда при создании индекса необходимо указать так называемый класс операторов. Существуют различные классы операторов, например, для столбца типа text это будет text_pattern_ops.

```
CREATE INDEX tickets_pass_name
ON tickets ( passenger_name text_pattern_ops );
```

Индексы со специальными классами операторов пригодны не для всех типов запросов. Поэтому, возможно, потребуется создать еще и индекс с классом операторов по умолчанию. Самостоятельно изучите этот вопрос с помощью раздела документации 11.9 «Семейства и классы операторов».

2). Перечень вопросов, выносимых на промежуточную аттестацию (зачет)

7 семестр

1 Реляционная модель данных. Основные понятия. Область применения
2 Этапы проектирования базы данных. Концептуальное проектирование
3 Метод моделирования «Сущность - связь» (ER - диаграмма). Примеры
4 Логическое проектирование. Правила отображения ER - диаграммы на логическую схему. Примеры
5 Физическое проектирование
6 Какие группы операторов выделяются в составе языка SQL?

7 Дайте неформальное определение основных понятий реляционной модели данных: отношение, кортеж, атрибут.
8 Для чего нужны внешние ключи в реляционных таблицах?
9 Что такое потенциальный ключ?
10 Назовите основные типы данных, используемые при создании столбцов таблицы
11 Правила целостности данных. Внешний ключ
12 Арифметические выражения в выборке
13 Использование команды DESCRIBE для вывода структуры таблицы
14 Использование опции WHERE
15 Опция ORDER BY
16 Числовые функции
17 Строковые функции
18 Функции для работы с датой
19 Преобразования типов данных
20 Функция COALESCE и связанные с ней функции
21 Операция CASE
22 Основные агрегатные функции (GROUP BY, HAVING)
23 Структура перекрестного запроса
24 Опция INNER JOIN
25 Опция OUTER JOIN
26 Соединение таблицы с самой собой
27 Подзапросы
28 Объединение результирующих множеств
29 Добавление строк в таблицу при помощи команды INSERT
30 Использование команды UPDATE для изменения строк таблицы
31 Удаление данных из таблицы при помощи команды DELETE
32 Понятие транзакции

6.2. Описание показателей и критериев контроля успеваемости, описание шкал оценивания

Для оценки знаний, умений, навыков и формирования компетенций по дисциплине применяется балльно-рейтинговая система контроля и оценки успеваемости студентов.

В основу балльно-рейтинговой системы (БРС) положены принципы, в соответствии с которыми формирование рейтинга студента осуществляется в ходе текущего и промежуточного контроля знаний обучающихся.

Таблица 7

Система рейтинговой оценки успеваемости

Баллы	Балльная оценка текущей успеваемости			
За защиту практической работы	2	3	4	5
За ответы на вопросы промежуточной аттестации	0-8,5	9-11,5	12-13,5	14-15
Оценка	Неудовлетворительно	Удовлетворительно	Хорошо	Отлично

Студенты, получившие за контрольное мероприятие оценку «неудовлетворительно», обязаны пройти его повторно и получить минимально допустимое количество баллов.

Таблица 8

Итоговая сумма баллов

Виды контроля	Количество видов контроля	Количество баллов за единицу	Количество баллов
За защиту практической работы	5	5	25
Ответы на вопросы промежуточной аттестации	1	15	15
Всего	-	-	40

Баллы могут снижаться в случаях, представленных в таблице 9.

Таблица9

Критерии снижения баллов

Критерий	Вычитаемый балл
При оценке ответов на вопросы по выполненной практической работе:	
Отсутствие ответа на вопрос	-3
Неполный ответ (за шт.)	-1

Таблица 10

Балльно-рейтинговая система контроля успеваемости

Шкала оценивания	Итоговый балл
20-40	зачтено
0-19	не зачтено

Критерии оценки ответов на вопросы промежуточной аттестации (зачет): один вопрос оценивается в 1 балл, общая сумма – 15 баллов. Если студент дал неполный ответ на вопрос со множеством вариантов ответа или открытый вопрос, то ему ставится 0,5 балла.

Студенты, набравшие по итогам балльно-рейтинговой системы более 19 баллов, могут претендовать на получение оценки «зачтено», соответствующей набранным баллам рейтинга в таблице 10.

Студенты, не выполнившие ни одной практической работы (или же защитившие практические работы на неудовлетворительные оценки) до зачета не допускаются.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1 Основная литература

1. Осипов, Д. Л. Технологии проектирования баз данных / Д. Л. Осипов. — Москва : ДМК Пресс, 2019. — 498 с. — ISBN 978-5-97060-737-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131692. — Режим доступа: для авториз. пользователей.
2. Джонатан, Л. Ядро Oracle. Внутреннее устройство для администраторов и разработчиков баз данных / Л. Джонатан ; перевод с английского А. Н. Киселев. — Москва : ДМК Пресс, 2015. — 372 с. — ISBN 978-5-97060-169-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/7307. — Режим доступа: для авториз. пользователей.
3. Флегонтов, А. В. Моделирование информационных систем. Unified Modeling Language : учебное пособие / А. В. Флегонтов, И. Ю. Матюшичев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2019. — 112 с. — ISBN 978-5-8114-2907-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/112065 — Режим доступа: для авториз. пользователей.

7.2 Дополнительная литература

1. Кара-Ушанов, В. Ю. SQL — язык реляционных баз данных : учебное пособие / В. Ю. Кара-Ушанов. — Екатеринбург : УрФУ, 2016. — 156 с. — ISBN 978-5-7996-1622-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/98296 — Режим доступа: для авториз. пользователей.
2. Разработка и защита баз данных в Microsoft SQL Server 2005 : учебное пособие. — 2-е изд. — Москва : ИНТУИТ, 2016. — 147 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100448 — Режим доступа: для авториз. пользователей.
3. Чубукова, И. А. Data Mining : учебное пособие / И. А. Чубукова. — 2-е изд. — Москва : ИНТУИТ, 2016. — 470 с. — ISBN 978-5-94774-819-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100582 — Режим доступа: для авториз. пользователей.

7.3 Нормативные правовые акты

1. Федеральный закон «Об информации, информационных технологиях и о защите информации» от 06.04.2011 N 65-ФЗ.
2. ГОСТ 15971-90 Системы обработки информации. Термины и определения.
3. ГОСТ 34.321-96. Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

Для самостоятельного изучения разделов и подготовки к занятиям могут быть использованы следующие ресурсы:

1. https://habr.com/ru – русскоязычный веб-сайт в формате коллективного блога с элементами новостного сайта, созданный для публикации новостей, аналитических статей, мыслей, связанных с информационными технологиями, бизнесом и интернетом (открытый доступ).
2. https://www.internet-technologies.ru – русскоязычный веб-сайт в формате коллективного блога с элементами новостного сайта, созданный для публикации новостей, аналитических статей, мыслей, связанных с информационными технологиями, бизнесом и интернетом (открытый доступ).
3. https://postgrespro.ru/docs/postgresql/9.6/plpgsql - русскоязычный сайт содержащий опубликованную документацию по PostgreSQL.

9. Перечень программного обеспечения

Таблица 11

Перечень программного обеспечения

№ п/п	Наименование раздела учебной дисциплины	Наименование программы	Тип программы	Автор	Год разработки
Основное ПО					
1	Раздел 1 Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL	Windows 7	Контролирующее	Microsoft	2009
		Internet Explorer 9-11 версии	Обучающее	Microsoft	2011-12
2	Раздел 2 Проектирование и управление базами данных в СУБД PostgreSQL	PostgreSQL	Обучающее	The PostgreSQL Global Development Group	2017
Дополнительное ПО					
1	Раздел 1 Понятие реляционной модели данных. Разработка физической модели данных применительно к СУБД PostgreSQL	Microsoft Visio 2016	Обучающее	Microsoft	2016

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Лекции проводятся в специализированной аудитории, оборудованной мультимедийным проектором для демонстрации компьютерных презентаций.

Для проведения практических занятий по дисциплине «Разработка баз данных в СУБД PostgreSQL» необходим компьютерный класс с предустановленным на ПЭВМ программным обеспечением, указанным в п. 9 в качестве основного ПО.

Таблица 12

Сведения об обеспеченности специализированными аудиториями, кабинетами, лабораториями

Наименование специальных помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	Оснащенность специальных помещений и помещений для самостоятельной работы
Аудитория для проведения занятий лекционного типа, практических занятий, курсового проектирования, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации (№УИТ-7, уч. корпус №12)	Персональные компьютеры в количестве 26 штук
Аудитория для проведения занятий лекционного типа, практических занятий, курсового проектирования, групповых и индивидуальных консультаций, текущего контроля и	Персональные компьютеры в количестве 14 штук

Наименование специальных помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	Оснащенность специальных помещений и помещений для самостоятельной работы
промежуточной аттестации (№УИТ-101, уч. корпус №12)	
Аудитория для проведения занятий лекционного типа, практических занятий, курсового проектирования, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации (№УИТ-102, уч. корпус №12)	Персональные компьютеры в количестве 14 штук
Центральная научная библиотека имени Н.И. Железнова	Читальные залы библиотеки
Общежития РГАУ-МСХА имени К.А. Тимирязева	Комнаты для самоподготовки

11. Методические рекомендации студентам по освоению дисциплины

Освоение теоретических основ курса «Разработка баз данных в СУБД PostgreSQL» предусматривает прослушивание и проработку материалов лекций, работу с рекомендованными литературными источниками и интернет-ресурсами. Лекции читаются в аудиториях, оснащенных компьютерной техникой, на основе подготовленных лектором презентаций с применением активных и интерактивных образовательных технологий. Практические навыки по курсу «Разработка баз данных в СУБД PostgreSQL» приобретаются путем выполнения практических работ. Практические занятия проводятся в компьютерных классах, оснащенных соответствующими техническими и программными средствами.

Виды и формы отработки пропущенных занятий

Студент, пропустивший лекционное занятие, обязан представить конспект пропущенной лекции. При пропуске практического занятия студент обязан получить у преподавателя индивидуальный вариант, выполнить и защитить его. Прием и защита индивидуального задания проводится в часы и дни, устанавливаемые преподавателем.

12. Методические рекомендации преподавателям по организации обучения по дисциплине

Реализация компетентного подхода должна предусматривать широкое использование в учебном процессе активных и интерактивных форм проведения занятий (мастер-классов, выполнения заданий на ПК) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков обучающихся.

Программу разработали:

д.э.н., профессор Худякова Е.В. _____ (подпись)

ст. преподаватель Ханжиян К.И. _____ (подпись)