МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

РТУ МИРЭА

Кафедра цифровой трансформации

Институт информационных технологий

Презентации по лекционным материалам по дисциплине «Разработка баз данных»

Направления подготовки:

01.03.04 Прикладная математика
09.03.03 Прикладная информатика
09.03.04 Программная инженерия
09.03.01 Информатика и вычислительная техника

Уровень: бакалавриат

Форма обучения: очная

Лекция №1. Реляционная модель данных и базовые операции SQL

Лектор
Исаева Мария Владимировна
Кандидат технических наук,
доцент кафедры
цифровой трансформации

2025/2026 учебный год

СВЕДЕНИЯ О ДИСЦИПЛИНЕ

Лекции ведут:

- Исаева Мария Владимировна, к.т.н., доцент каф. Цифровой трансформации
- Дзгоев Алан Эдуардович, к.т.н., доцент каф. Цифровой трансформации (Dzgoev@mirea.ru / При обращении в теме письма указывайте номер группы)
- Миронов Антон Николаевич, Старший преподаватель каф. Цифровой трансформации
- Семыкина Наталья Александровна, д.т.н., профессор каф. Цифровой трансформации
- Резеньков Роман Николаевич, к.т.н., доцент каф. Цифровой трансформации

Объём **аудиторной** работы по дисциплине в 5 семестре:

- Лекции – 16 часов;
- Практические занятия – 48 часа;
- Аттестация – экзамен.

Допуск к экзамену:

- посещение лекций и практических работ;
- выполнение в установленный срок всех практических работ.

подробности в памятке по дисциплине (следующие слайды)

Выполнение практических заданий

1. Все материалы курса на https://online-edu.mirea.ru в разделе **Разработка баз данных**: *доступ к материалам курса открывается по расписанию занятий.*

2. Доступ с аккаунта МИРЭА.

3. Все создаваемые файлы прикладываются к заданию.

4. Учёт выполненных работ.

5. Общение с преподавателем через отзывы в заданиях.

...ков ...ке ...ль ...вия"	Л4_% кликов по кнопке "контроль присутствия"	Л5_% кликов по кнопке "контроль присутствия"	Л6_04-05_% присутствия	Л7_18-05_% присутствия	Л8_01-06_% присутствия	%СРЗНАЧ участия в лекциях	Задание:Практика 1_Законодательная и нормативно-техническая база стандартизации в РФ	Задание:Практика 2_Нормативные документы по стандартизации ИТ	Задание:Практика 3_ч1_Классификаторы в области ИТ	Задание:Практика 3_ч2_Классификаторы в области ИТ	Задание:Практика 3_ч3_Классификаторы в области ИТ	Задание:Практика 4_ч1_Создание технического задания в соответствии с ГОСТ 34.602–2020
	0	50				6,3	-	-	-	-	-	-
	0	50		100	100	31,3	-	1	1	1	1	1
)	0	0				0	-	-	-	-	-	-
00	0	50	100		100	56,3	1	1	1	1	1	1
100	100	100	100	100	100	87,5	1	1	1	1	1	1
0	0	0				0	-	-	-	-	-	-
0	50	0				6,3	1	-	-	-	-	-
100	0	100		100		56,3	1	1	1	1	1	1
0	0	0				12,5	1	1	-	-	-	-
0	0	0				0	-	-	-	-	-	-
0	0	0				0	-	-	-	-	-	-
100	0	0				18,8	-	-	-	-	-	-
0	100	100	100	100	100	68,8	1	1	1	1	1	1
0	0	50				18,8	-	-	-	-	-	-
0	0	0	100	100	100	50	1	1	1	1	1	1
0	0	0				0	-	-	-	-	-	-
0	100	100	100	100	100	100	1	1	1	1	1	1
0	0	0	0			0	-	-	-	-	-	-
100	0	0	0			18,8	1	1	1	1	-	-
50	0	50	0			18,8	-	-	-	-	-	-
0	0	0	0			0	-	-	-	-	-	-
0	0	0	0			0	-	-	-	-	-	-
0	0	0	0		100	12,5	1	1	1	1	1	1
100	100	0	0			25	1	1	-	-	-	-
0	0	0	0			0	-	-	-	-	-	-
100	100	100	100	100	100	100	1	1	1	1	1	1

Рекомендуемая литература

1. Новиков Б.А. Основы технологий баз данных: учебное пособие / Б.А. Новиков, Е.А. Горшкова, Н.Г. Графеева; под.ред. Е.В. Рогова. – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с. Режим доступа: https://postgrespro.ru//education/books/dbtech
2. Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. Пособие / Е.П. Моргунов; под. ред. Е.В. Рогова, П.В. Лузанова. – СПб.: БХВ-Петербург, 2018. – 336 с.: ил. Режим доступа: https://postgrespro.ru/education/books/sqlprimer
3. Комаров В.И. Путеводитель по базам данных. – М.: ДМК-Пресс, 2024. – 520 с. Режим доступа: https://edu.postgrespro.ru/dbguide.pdf
4. Смирнов М.В. Проектирование баз данных [Электронный ресурс]: Конспект лекций / Смирнов М.В. - М., МИРЭА – Российский технологический университет, 2020 – 1 электрон. Опт. диск (CD-ROM). Режим доступа: https://e.lanbook.com/book/163892/
5. Чистякова М.А. Проектирование и эксплуатация баз данных [Электронный ресурс]: Учебно-методическое пособие / Чистякова М.А., Иванова И.А., Котилевец И.Д. – М.: МИРЭА – Российский технологический университет, 2021. – 1 электрон. Опт. диск (CD-ROM). Режим доступа: https://e.lanbook.com/book/176572/
6. Братусь Н.В. Базы данных. Часть 1 [Электронный ресурс]: Практикум / Братусь Н.В.,Маличенко С.В., Матчин В.Т. — М.: МИРЭА – Российский технологический университет, 2024. — 1 электрон. опт. диск (CD-ROM) – Режим доступа: https://ibc.mirea.ru/books/SHARE/5859/
7. Моргунов Е. П. PostgreSQL. Профессиональный SQL : учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова. — М.: ДМК Пресс, 2025. — 444 с. Режим доступа: https://postgrespro.ru/education/books/advancedsql
8. Рогов Е. В. PostgreSQL 17 изнутри. — М.: ДМК Пресс, 2025. — 668 с. Режим доступа: https://postgrespro.ru/education/books/internals
9. Введение в системы баз данных. : Пер. с англ. / К. Дж. Дейт .— М. : Изд. дом "Вильямс" , 2001 .— 1072 с. https://ibc.mirea.ru/books/search/?search_field=Дейт&page=3

4

Дисциплина «Разработка баз данных»

План-график занятий по неделям

Необходимо предусмотреть Три текущих теста и Две контрольные работы

Неделя	Лекции	Практические работы	Дедлайны, текущие и контрольные мероприятия
1	Лекция 1. Тема: Реляционная модель данных и базовые операции SQL. (2 часа)	Занятие 1. ПРАКТИЧЕСКАЯ РАБОТА №1. Создание БД и запросы на выборку данных (2 часа)	
2		Занятие 2. ПРАКТИЧЕСКАЯ РАБОТА №1. (2 часа)	Дедлайн по защите отчета по практике №1.
		Занятие 3. ПРАКТИЧЕСКАЯ РАБОТА №2. Многотабличные запросы, использование операций объединения, пересечения, разности (2 часа)	
3	Лекция 2. Тема: Многотабличные запросы и теоретико-множественные операции. (2 часа)	Занятие 4. ПРАКТИЧЕСКАЯ РАБОТА №2.(2 часа)	
4		Занятие 5. ПРАКТИЧЕСКАЯ РАБОТА №2. (2 часа)	Дедлайн по защите отчета по практике №2.
		Занятие 6 ПРАКТИЧЕСКАЯ РАБОТА №3 Использование подзапросов и CTE (2 часа)	
5	Лекция 3. Использование подзапросов и обобщетабличные выражения (2 часа)	Занятие 7. ПРАКТИЧЕСКАЯ РАБОТА №3 (2 часа)	Тест №1. (по лекциям №1-2).
6		Занятие 8. ПРАКТИЧЕСКАЯ РАБОТА №3 (2 часа).	Дедлайн по защите отчета по практике №3.
		Занятие 9. ПРАКТИЧЕСКАЯ РАБОТА №4 Использование оконных функций и функций ранжирования (2 часа)	
7	Лекция 4. Тема: SQL. Оконные функции и аналитические запросы (2 часа)	Занятие 10. ПРАКТИЧЕСКАЯ РАБОТА №4 (2 часа)	Дедлайн по защите отчета по практике №4.
8		Занятие 11. ПРАКТИЧЕСКАЯ РАБОТА №4 (2 часа)	Дедлайн по защите отчета по практике №4.
		Занятие 12. ПРАКТИЧЕСКАЯ РАБОТА №5 (2 часа)	
9	Лекция 5. Операторы модификации данных, объекты базы данных (2 часа)	Занятие 13. ПРАКТИЧЕСКАЯ РАБОТА №5 (2 часа)	Дедлайн по защите отчета по практике №5.

План – график лекций и практических занятий

10		Занятие 14. ПРАКТИЧЕСКАЯ РАБОТА №5 (2 часа)	Дедлайн по защите отчета по практике №5.
		Занятие 15. ПРАКТИЧЕСКАЯ РАБОТА №6 (2 часа)	
11	Лекция 6. Управление данными: триггеры, курсоры (2 часа)	**Занятие 16.** 2 часа	**Контрольная работа №1.**
12		Занятие 17. ПРАКТИЧЕСКАЯ РАБОТА №6 (2 часа)	Дедлайн по защите отчета по практике №6.
		Занятие 18. ПРАКТИЧЕСКАЯ РАБОТА №7 Оптимизация запросов (2 часа)	**Тест №2. (по лекциям №3-5).**
13	Лекция 7. Оптимизация запросов (2 часа)	Занятие 19. ПРАКТИЧЕСКАЯ РАБОТА №7 (2 часа)	
14		Занятие 20. ПРАКТИЧЕСКАЯ РАБОТА №7 (2 часа)	Дедлайн по защите отчета по практике №7.
		Занятие 21. ПРАКТИЧЕСКАЯ РАБОТА №8 Хранение неструктурированных данных (2 часа)	
15	Лекция 8. Администрирование баз данных и хранение неструктурированных данных (2 часа) Тест №3. (по лекциям №6-7).	Занятие 22. ПРАКТИЧЕСКАЯ РАБОТА №8 (2 часа)	
16		Занятие 23. ПРАКТИЧЕСКАЯ РАБОТА №8 (2 часа)	Дедлайн Практики №8.
		Занятие 24. 2 часа	**Контрольная работа №2.**
Итого	**16 часов**	**48 часов**	

ПАМЯТКА

о правилах обучения дисциплины

«Разработка баз данных» (2025-2026, I семестр)

№	Наименование	Формат	Период проведения	Баллы (макс.)
1	Защита практических работ (**8 практик**)	Очно	Сентябрь 2025 – Декабрь 2025	48 (6 баллов за 1 практику)
2	Контрольный тест №1 (по лекциям №1 и №2)	Очно, СДО	Октябрь 2025	5
3	Контрольный тест №2 (по лекциям №3 - №5)	Очно, СДО	Ноябрь 2025	5
4	Контрольная работа №1	Очно, СДО	Ноябрь 2025	10
5	Контрольный тест №3 (по лекциям №6 - №8)	Очно, СДО	Декабрь 2025	6
6	Контрольная работа №2	Очно, СДО	Декабрь 2025	10
7	Посещение (лекции 8 занятий, практики 24 занятия)	Очно	Сентябрь– Декабрь 2025	16 б. 0,5 баллов за посещения 1 лекции и практики (0,5·32 пар)
Максимальная сумма баллов за активность по дисциплине в течение учебного семестра				**100**

ДИСЦИПЛИНА	**Разработка баз данных**
	(укажите полное наименование дисциплины без сокращений)
ИНСТИТУТ	информационных технологий
	(укажите название учебного института)
КАФЕДРА	Цифровой трансформации
	(укажите полное наименование кафедры)
Уровень обучения	Бакалавриат
Аудиторные часы	16/0/48
	(укажите количество часов лекционных, лабораторных и практ

Лекции 16 часов.

Обязательное посещение всех лекций.

В тестах в рамках мероприятий текущего контроля вопросы из лекций

ПАМЯТКА

о правилах обучения дисциплины

«Разработка баз данных» (2025-2026, I семестр)

№	Наименование	Формат	Период проведения	Баллы (макс.)
1	Защита практических работ (**8 практик**)	Очно	Сентябрь 2025 – Декабрь 2025	48 (6 баллов за 1 практику)
2	Контрольный тест №1 (по лекциям №1 и №2)	Очно, СДО	Октябрь 2025	5
3	Контрольный тест №2 (по лекциям №3 - №5)	Очно, СДО	Ноябрь 2025	5
4	Контрольная работа №1	Очно, СДО	Ноябрь 2025	10
5	Контрольный тест №3 (по лекциям №6 - №8)	Очно, СДО	Декабрь 2025	6
6	Контрольная работа №2	Очно, СДО	Декабрь 2025	10
7	Посещение (лекции 8 занятий, практики 24 занятия)	Очно	Сентябрь– Декабрь 2025	16 б. 0,5 баллов за посещения 1 лекции и практики (0,5·32 пар)
Максимальная сумма баллов за активность по дисциплине в течение учебного семестра				**100**

Практические работы (8 работ)

Обязательное посещение всех практических занятий.

Допуск к экзамену – очная защита всех практических работ в течение семестра. Защита отчёта до дедлайна

Защита практической работы после дедлайна – 0 баллов, но работа засчитывается.

Если есть справка по перенесённой болезни, заверенная в учебном отделе, то эту справку необходимо принести и показать преподавателю по практике. В таком случае назначается пересдача пропущенной практической работы в день консультаций (важно: студент защищает именно ту практическую работу, которую он пропустил по болезни, согласно датам в справке). Если студент пропустил практические работы без уважительной причины и, соответственно, работу не защитил, то на экзамен он не допускается.

Если студент загрузил отчет до дедлайна, но не защитил, то такой отчет не засчитывается студенту, т.е. не сдал.

Дополнительного времени на защиту практических работ не выделяется.

Выполненные работы студент загружает в СДО в формате pdf.

Фон области построения модели – белый (в отчёте).

Если студент не загрузил отчет до дедлайна, то загрузка после дедлайна станет недоступна.

Контрольные тесты

Тесты студенты выполняют в СДО на паре.

Баллы минусуются, если по базовым вопросам студент отвечает не правильно.

Проходного балла нет, сколько студент набрал столько и остается.

Контрольный тест №1. (Лекции 1, 2) проводится на 5-м практическом занятии в начале пары (20 минут).

Контрольный тест №2. (Лекции 3, 4, 5) проводится на 16-м практическом занятии в начале пары (20 минут)..

Контрольный тест №3 (Лекции №6-7) проводится на 8-й лекции в начале пары (20 минут).

Контрольные работы

Выполняются на 15 и 24 практических занятиях в аудитории.

Студент получает задание на паре.

Время выполнения контрольной работы – 1 час.

Экзамен

Допуск к экзамену – защита всех практических работ в течение семестра очно. В случае, если студент не сдал какие-либо практики, у него есть возможность сдать их во время экзамена, если успеет. В противном случае, отправляется на пересдачу (необходимость сдачи работ сохраняется).

Если студент набрал меньше 50 % от общей суммы баллов, то экзамен будет проходить по билету (2теоретических вопроса + 1 задача).

Если студент набрал больше 50% от суммы максимальной суммы баллов, то сдаёт тестирование на экзамене. + добавляются баллы за активность.

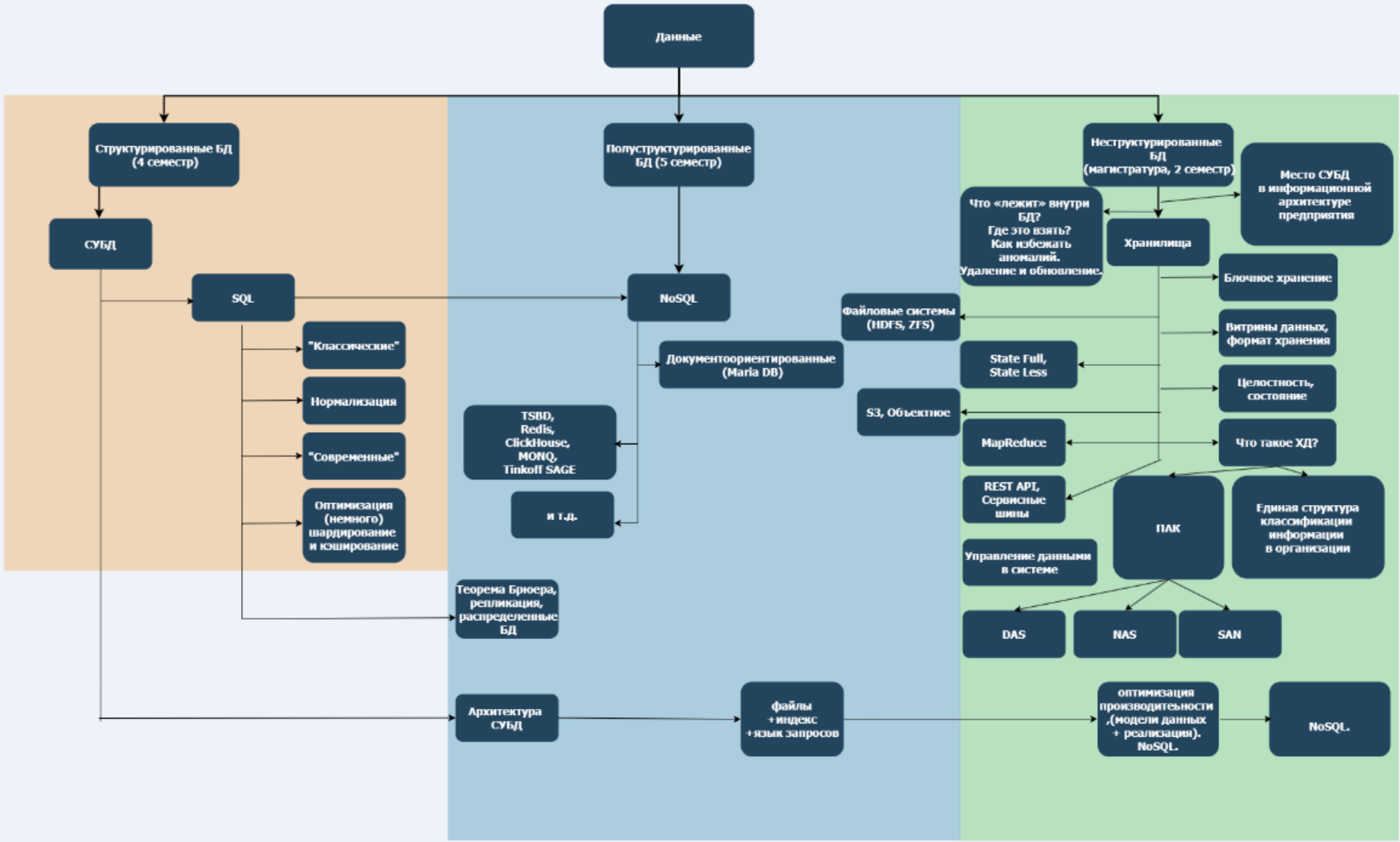
За тест студент может набрать 20 баллов, за которые можно получить:

- От 10 до 14 баллов — оценка «3»;
- От 15 до 19 баллов — оценка «4»;
- 20 баллов — оценка «5».

К баллам за тест прибавляются доп. баллы, которые студент получает в течение семестра по данным критериям:

- Набрано 55-64 балла за семестр — 1 доп.;
- Набрано 65-74 балла за семестр — 2 доп.;
- Набрано 75-84 балла за семестр — 3 доп.;
- Набрано 85-94 балла за семестр — 4 доп.;
- Набрано 95-100 балла за семестр — 5 доп.

В случае, если студент сдал тест на «2», то отправляется на пересдачу без учета доп. баллов.



Заметки:
Бакалавриат:

Проектирование БД (2 курс): Классические структурированные БД.

СУБД = приложение / БП (фактические информационные сущности).

Бакалавриат:

Разработка БД (3 курс) Упор на стройку СУБД в конечное приложение:

1) Место СУБД в архитектуре современных **приложений**.
2) Нестандартные СУБД (NoSQL, S3 Хранилища…).

3) Распределенные БД (шардирование, кэширование, проблема удаленного узла).

Управление данными в приложении:

1) Физический. Блочное ХД.

2) Файловая система.

3) Объект.
4) СУБД (или набор СУБД). 5) Приложение.

Магистратура:

Проектирование
и разработка баз и хранилищ данных (магистратура 1 курс):

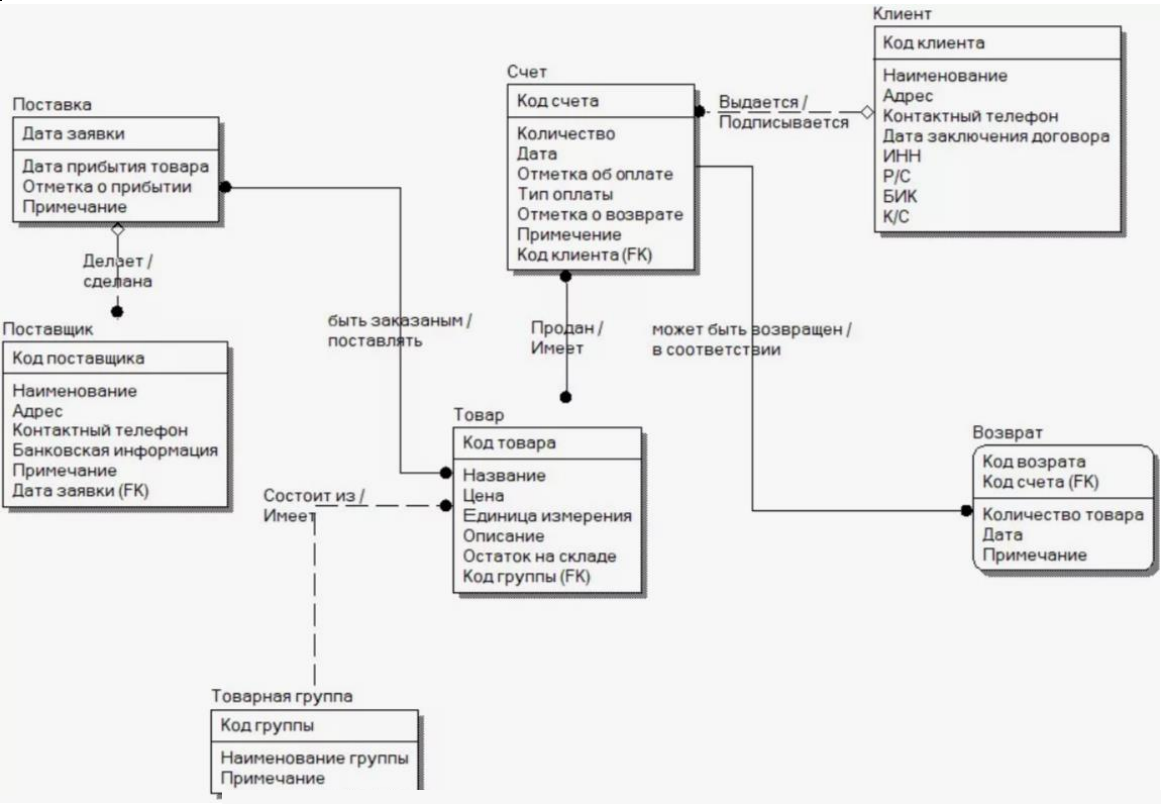
1) Место СУБД в информационной архитектуре **предприятия.**

2) Онтологическая модель. Что «лежит» внутри БД? Где это взять?

Как избежать аномалий. Удаление и обновление.

3) Единая структура классификации.

MapReduce. NoSQL, State Less, State Full, состояние , целостность данных.



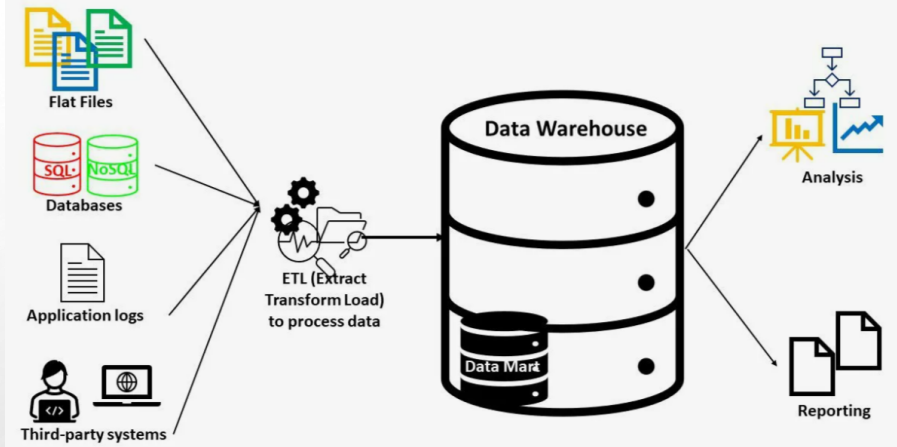
2 курс.

Проектирование баз данных

3 курс.

Разработка баз данных

PostgreSQL



Магистратура

Проектирование и разработка баз и хранилищ данных

Тема 1_ Реляционная модель данных и базовые операции SQL

1.1_Реляционная модель данных 12

DDL 21

Типы данных 41

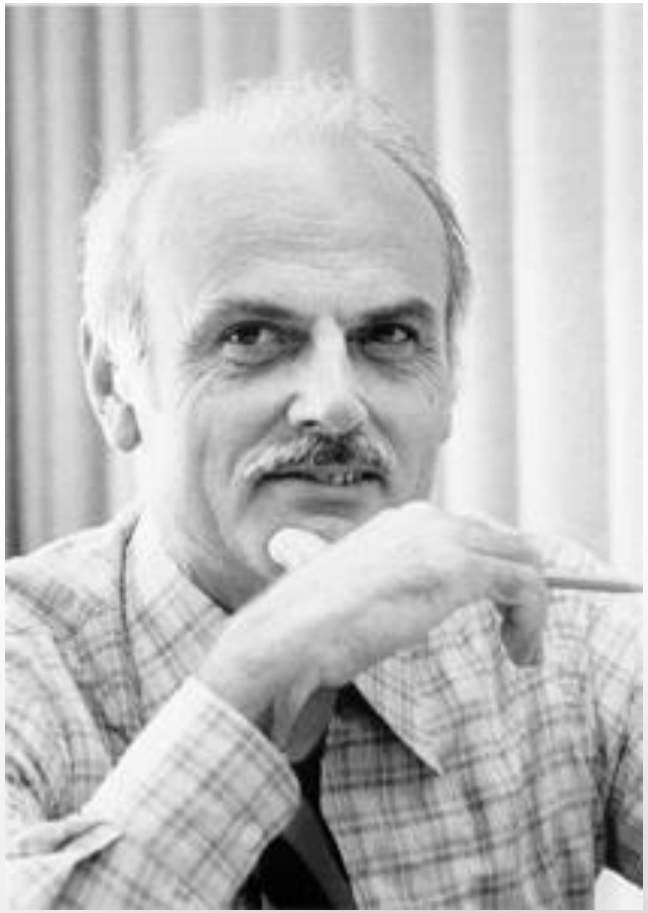
1.2_Основы SQL 45

Трехзначная логика и NULL-значения 56

Агрегатные функции. Группировка 60

Определение реляционной модели данных

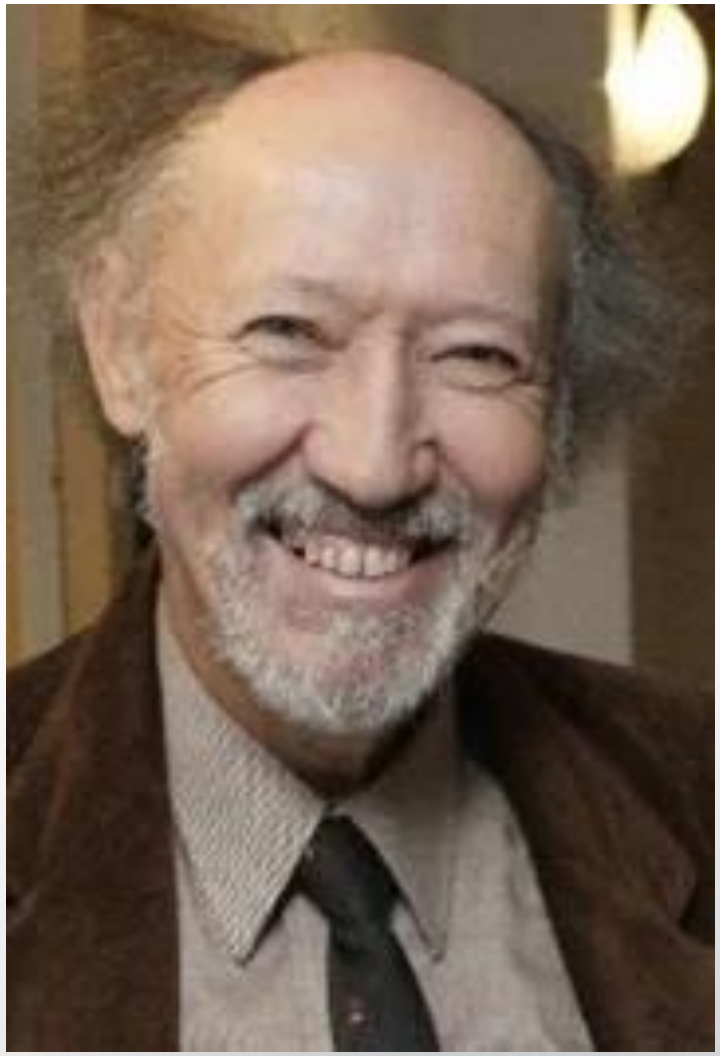
Эдгар Кодд в 1970 издал работу «A Relational Model of Data for Large Shared Data Banks», которая считается первой работой по реляционной модели данных. модель данных



Модель данных - это некоторый набор родовых понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими базы данных, если они основываются на этой модели

Определение реляционной модели данных

К. Дж. Дейт. Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. — М.: «Вильямс», 2006.



Реляционная модель данных состоит из:

- Структурная часть.
- Манипуляционная часть.
- Целостная часть.

Определение реляционной модели данных

- **Структурная часть модели данных** определяет основные логические структуры данных, которые могут быть использованы на уровне пользователя при организации базы данных, соответствующие рассматриваемой модели.

- **Манипуляционная часть модели данных** содержит спецификацию одного или нескольких языков, предназначенных для формирования запросов к базе данных.

- **Целостная часть модели данных -** это набор правил и ограничений, которые гарантируют точность, надежность и консистентность данных, хранящихся и обрабатываемых в соответствии с этой моделью.

Структурная часть реляционной модели данных

- **Реляционная модель данных —** это концептуальная модель организации баз данных, основанная на представлении данных в виде отношений (таблиц).
- **Основные понятия, лежащие в основе этой модели:**

Структурная часть реляционной модели данных

- **Отношение — это таблица, состоящая из строк и столбцов**
- **Кортеж — это строка в таблице (отношении)**

- **Атрибут - это столбец в таблице (отношении)**
- **Домен - это набор допустимых значений для атрибута.**
- **Ключ - атрибут или набор атрибутов, однозначно идентифицирующих кортеж в отношении**

Элемент реляционной модели	Форма представления
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Любой различимый объект
Атрибут	Заголовок столбца таблицы
Домен	Множество атомарных значений атрибута
Тело отношения	Множество строк таблицы

Элементы реляционной модели (1)

Целостная часть реляционной модели данных

- *Требованием целостности сущности (entity integrity)*: у любой переменной отношения должен существовать первичный ключ, и никакое значение первичного ключа в кортежах значения-отношения переменной отношения не должно содержать неопределенных значений.
- *Требованием целостности по ссылкам (referential integrity)*: для каждого значения внешнего ключа, появляющегося в кортеже ссылающегося отношения, должен найтись кортеж в главном отношении с таким же значением первичного ключа, либо значение внешнего ключа должно быть полностью неопределенным

Стандарты SQL

- Реляционными базами данных управляют с помощью языка **SQL - Structured Query Language**. Формально стандарт SQL называется ISO/IEC 9075 «Database Language SQL» (Язык баз данных SQL)
- Американский национальный институт стандартов (ANSI)
- Описание стандарта разделяется на несколько частей, каждая из которых также имеет короткое имя и номер:
- ISO/IEC 9075-1 Структура (SQL/Framework)
- ISO/IEC 9075-2 Основа (SQL/Foundation)
- ISO/IEC 9075-3 Интерфейс уровня вызовов (SQL/CLI)
- ISO/IEC 9075-4 Модули постоянного хранения (SQL/PSM)
- ISO/IEC 9075-9 Управление внешними данными (SQL/MED)
- ISO/IEC 9075-10 Привязки объектных языков (SQL/OLB)
- ISO/IEC 9075-11 Схемы информации и определений (SQL/Schemata)
- ISO/IEC 9075-13 Программы и типы, использующие язык Java (SQL/JRT)
- ISO/IEC 9075-14 Спецификации, связанные с XML (SQL/XML)
- ISO/IEC 9075-15 Многомерные массивы (SQL/MDA)
- ISO/IEC 9075-16 Запросы графов свойств (SQL/PGQ)

Группы операторов SQL

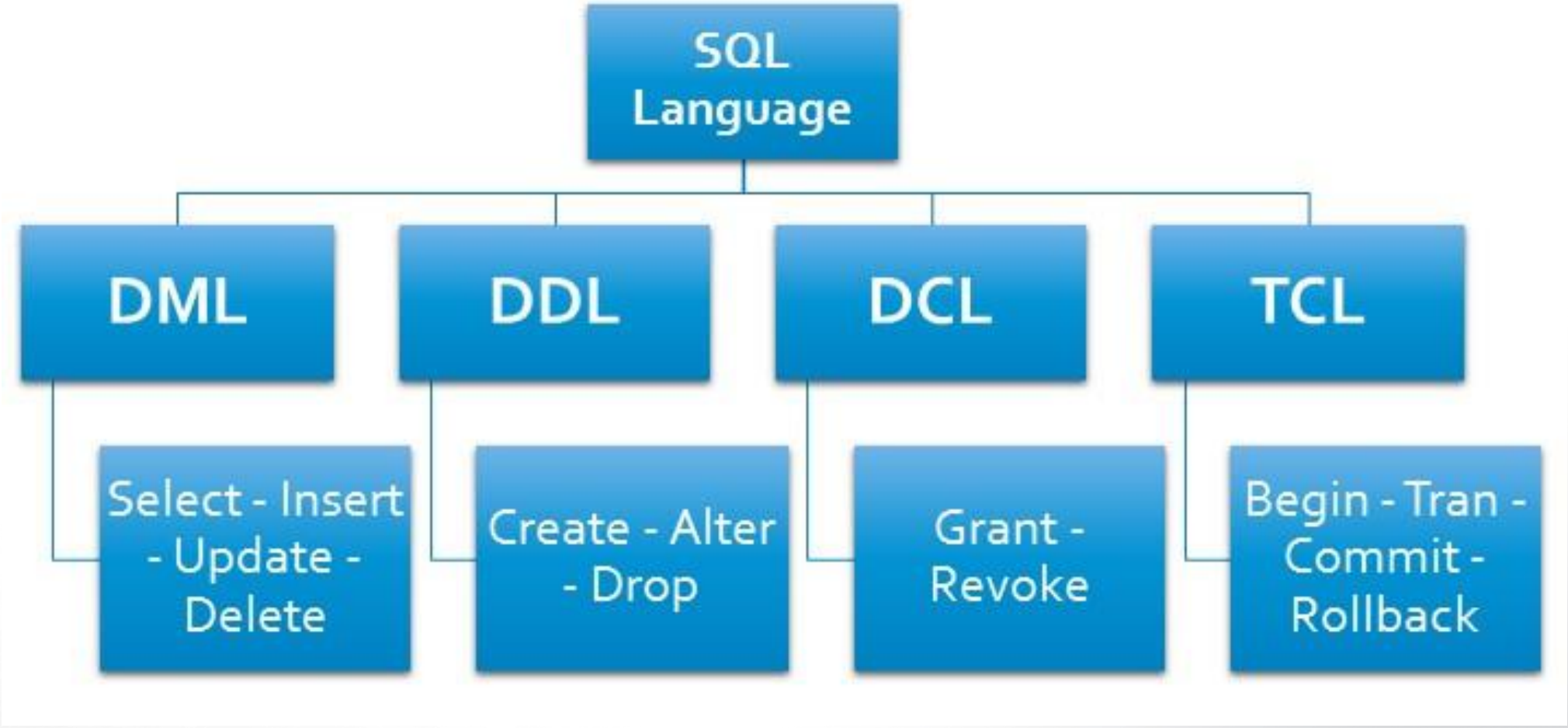
Операторы определения данных (*Data Definition Language, DDL*)

Операторы манипуляции данными (*Data Manipulation Language, DML*)

Операторы определения доступа к данным (*Data Control Language, DCL*)

Операторы управления транзакциями (*Transaction Control Language, TCL*)

Группы операторов SQL



Data Definition Language

Язык определения данных (Data Definition Language) предназначен для создания, изменения и удаления объектов базы данных.

CREATE используется для создания новых объектов в базе данных, таких как таблицы, индексы, представления, функции, триггеры и другие:

- CREATE TABLE: создание новой таблицы.
- CREATE INDEX: создание индекса для ускорения поиска по столбцу или набору столбцов.
- CREATE VIEW: создание представления для удобного доступа к данным.

Data Definition Language

ALTER используется для изменения структуры или свойств существующих объектов в базе данных:

- ALTER TABLE: изменение структуры таблицы, добавление или удаление столбцов, изменение типов данных и другие операции.
- ALTER INDEX: изменение индекса, добавление или удаление столбцов из индекса.
- ALTER VIEW: изменение определения представления.

Data Definition Language

DROP используется для удаления объектов из базы данных. Например:

- DROP TABLE: удаление таблицы и связанных с ней данных и индексов.
- DROP INDEX: удаление индекса.
- DROP VIEW: удаление представления.
- DROP FUNCTION: удаление функции.

TRUNCATE используется для удаления всех записей из таблицы, сохраняя структуру таблицы.

Виды ограничений

- Declarative Constraints, или декларативные ограничения. Определяются при создании таблиц и объявляются вместе со схемой базы данных с помощью языка определения данных. К ним относятся ограничение UNIQUE, CHECK, DEFAULT, NOT NULL, PRIMARY KEY и FOREIGN KEY.

- Procedural Constraints, или процедурные ограничения. Реализуются с использованием хранимых процедур, триггеров и других программных объектов в базе данных. (бизнес-правила)

Ограничение - проверка

Ограничение-проверка — наиболее общий тип ограничений. В его определении можно указать, что значение данного столбца должно удовлетворять логическому выражению (проверке истинности).

Ограничение столбца

```
CREATE TABLE products (
    product_no INT,
    name text,
    price numeric(10, 2) CHECK (price > 0));
```

```
CREATE TABLE products (
    product_no INT,
    name text,
    price numeric(10, 2)
CONSTRAINT positive_price CHECK
(price > 0) );
```

Ограничение таблицы

Ограничение - проверка

```
CREATE TABLE products (
    product_no INT,
    name text,
    price numeric(10, 2) CHECK (price > 0),
    discounted_price numeric(10, 2) CHECK
(discounted_price > 0),
    CHECK (price > discounted_price)
);
```

```
CREATE TABLE products (
    product_no INT,
    name text,
    price numeric(10,2) ,    CHECK (price > 0),
    discounted_price numeric(10,2),
    CHECK (discounted_price > 0),
    CONSTRAINT valid_discount
    CHECK (price > discounted_price)
);
```

Ограничение NOT NULL

Ограничение NOT NULL просто указывает, что столбцу нельзя присваивать значение NULL.

```
CREATE TABLE products (
    product_no integer NOT NULL,
    name text NOT NULL,
    price numeric
);
```

```
CREATE TABLE products (
    product_no integer NOT NULL,
    name text NOT NULL,
    price numeric NOT NULL CHECK (price > 0)
);
```

Несколько ограничений столбца

Ограничение уникальности

Ограничения уникальности гарантируют, что данные в определённом столбце или группе столбцов уникальны среди всех строк таблицы.

```
CREATE TABLE products (
    product_no integer UNIQUE,
    name text,
    price numeric
);
```

Только в одной строке может быть NULL

```
CREATE TABLE products (
    product_no integer UNIQUE NULLS
                        NOT DISTINCT,
    name text,
    price numeric );
```

```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    UNIQUE (a, c) );
```

Ограничение первичного ключа

Ограничение первичного ключа означает, что образующий его столбец или группа столбцов может быть уникальным идентификатором строк в таблице.

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric(10, 2)
);
```

```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    PRIMARY KEY (a, c)
);
```

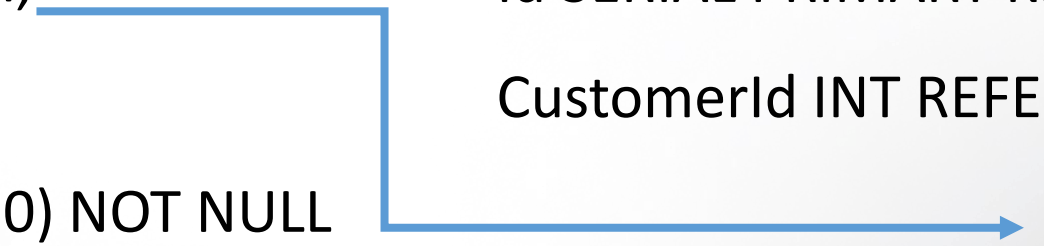
Составной первичный ключ

Ограничение внешнего ключа

Ограничение внешнего ключа указывает, что значения столбца (или группы столбцов) должны соответствовать значениям в некоторой строке другой таблицы.

```
CREATE TABLE Customers
(
    Id SERIAL PRIMARY KEY,
    Age INT,
    FirstName VARCHAR(20) NOT NULL
);
```

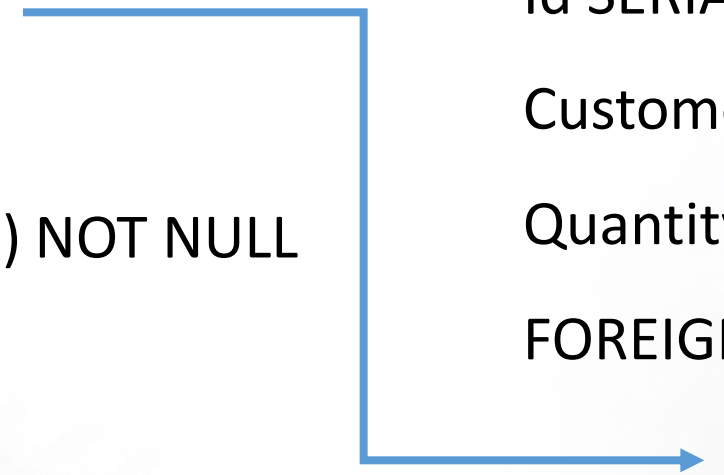
```
CREATE TABLE Orders
(
    Id SERIAL PRIMARY KEY,
    CustomerId INT REFERENCES
                    Customers (Id),
    Quantity INT
);
```



Ограничение внешнего ключа

```
CREATE TABLE Customers
(
    Id SERIAL PRIMARY KEY,
    Age INT,
    FirstName VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE Orders
(
    Id SERIAL PRIMARY KEY,
    CustomerId INT,
    Quantity INT,
    FOREIGN KEY (CustomerId)
        REFERENCES Customers (Id)
);
```



Ограничение внешнего ключа

С помощью выражений **ON DELETE** и **ON UPDATE** можно установить действия, которые выполняются соответственно при удалении и изменении связанной строки из главной таблицы. Опции:

- **CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
- **RESTRICT**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице.
- **NO ACTION**: действие по умолчанию, предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. И генерирует ошибку. В отличие от RESTRICT выполняет отложенную проверку на связанность между таблицами.
- **SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL.
- **SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение NULL.

Удаление и изменение таблиц

```
DROP TABLE [ IF EXISTS ] имя [, ...] [ CASCADE | RESTRICT ]

DROP TABLE IF EXISTS films, distributors;
```

```
ALTER TABLE products ADD COLUMN description text;

 ALTER TABLE products DROP COLUMN description CASCADE;

ALTER TABLE products ADD CHECK (name <> '');

ALTER TABLE products ADD CONSTRAINT some_name UNIQUE (product_no);

ALTER TABLE products ADD FOREIGN KEY (product_group_id)

REFERENCES product_groups;
```

Удаление и изменение таблиц

```
ALTER TABLE products DROP CONSTRAINT some_name;

ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL;

ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77;

ALTER TABLE products ALTER COLUMN price DROP DEFAULT;

ALTER TABLE products ALTER COLUMN price TYPE numeric(10,2);

ALTER TABLE products RENAME COLUMN product_no TO product_number;

ALTER TABLE products RENAME TO items;
```

Очистка таблиц

TRUNCATE [TABLE] [ONLY] имя [*] [, ...]
[RESTART IDENTITY | CONTINUE IDENTITY] [CASCADE | RESTRICT]

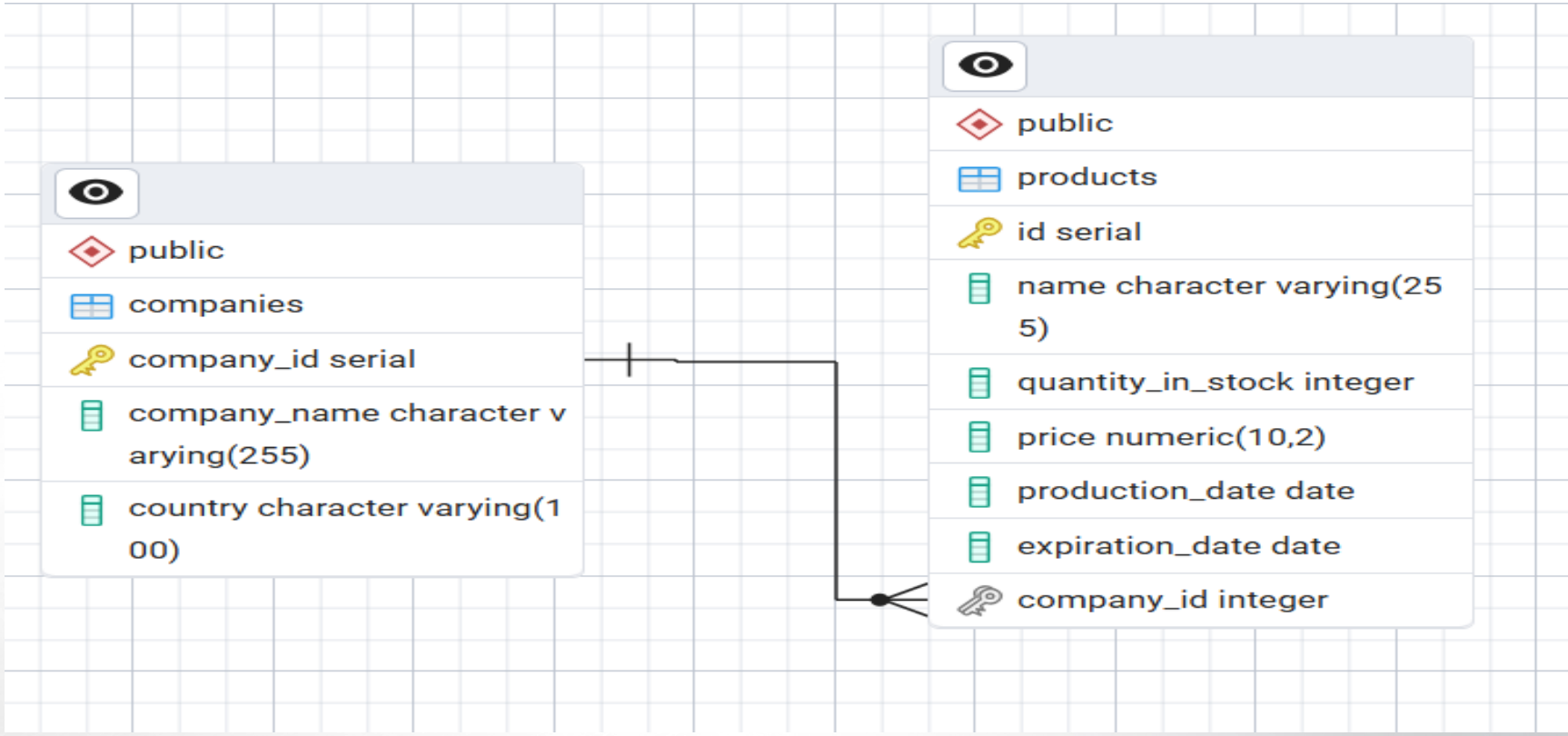
TRUNCATE TABLE Employee;

Преимущества TRUNCATE по сравнению с DELETE:

- TRUNCATE значительно быстрее, особенно для больших таблиц, так как не сканирует каждую строку.
- TRUNCATE немедленно освобождает дисковое пространство
- TRUNCATE не вызывает триггеры, которые могут быть настроены для операций удаления строк.
- После выполнения TRUNCATE данные не могут быть восстановлены с помощью команды ROLLBACK.
- TRUNCATE не может быть использована для таблиц, на которые есть внешние ссылки, если только эти таблицы также не будут очищены этой же командой.

CREATE DATABASE

- **Создаём новую базу данных:** CREATE DATABASE shop;



CREATE TABLE

Название столбца/атрибута	Тип данных	Ограничения
company_id	Целочисленный с автоувеличением	Первичный ключ
Название производителя (company_name)	Строковый, длина 255	Обязательность
Страна производителя(country)	Строковый, длина 100	-

- **Создаем таблицу companies:**

```
CREATE TABLE companies (
    company_id SERIAL PRIMARY KEY,
    company_name VARCHAR(255) NOT NULL,
    country VARCHAR(100)
);
```

CREATE TABLE

- **Создаём таблицу products**

Название столбца/атрибута	Тип данных	Ограничения
id	Целочисленный с автоувеличением	Первичный ключ
Название (name)	Строковый, длина 255	Обязательность
Количество в магазине (quantity_in_stock)	Целочисленный	Обязательность Условие на значение (>=0) По умолчанию = 0
Цена (price)	Дробный(2 знака после запятой)	Обязательность Условие на значение (> 0)
Дата производства (production_date)	Дата/время	Обязательность
Срок годности (expiration_date)	Дата/время	может быть NULL, если продукт не имеет срока годности
Идентификатор производителя		Внешний ключ

CREATE TABLE

- **Создаём таблицу products**

```
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    quantity_in_stock INT NOT NULL DEFAULT 0,
    price DECIMAL(10, 2) NOT NULL,
    production_date DATE NOT NULL,
    expiration_date DATE,
    CHECK (quantity_in_stock >= 0),
    CHECK (price > 0),
    company_id INTEGER,
    FOREIGN KEY(company_id) REFERENCES companies(company_id) ON
DELETE SET NULL
);
```

Добавление ограничений

```
ALTER TABLE products

ADD CONSTRAINT check_production_date CHECK (production_date <=

CURRENT_DATE);
```

```
ALTER TABLE products

ADD CONSTRAINT check_pexpiration_date CHECK (expiration_date >

CURRENT_DATE);
```

Типы данных

- Числовые типы :

Имя	Размер	Описание	Диапазон
smallint	2 байта	целое в небольшом диапазоне	-32768 .. +32767
integer	4 байта	типичный выбор для целых чисел	-2147483648 .. +2147483647
bigint	8 байт	целое в большом диапазоне	-9223372036854775808 .. 9223372036854775807
decimal	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
numeric	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
real	4 байта	вещественное число с переменной точностью	точность в пределах 6 десятичных цифр
double precision	8 байт	вещественное число с переменной точностью	точность в пределах 15 десятичных цифр
smallserial	2 байта	небольшое целое с автоувеличением	1 .. 32767
serial	4 байта	целое с автоувеличением	1 .. 2147483647
bigserial	8 байт	большое целое с автоувеличением	1 .. 9223372036854775807

Типы данных

- Символьные типы :

Имя	Описание
character varying(*n*), varchar(*n*)	строка ограниченной переменной длины
character(*n*), char(*n*)	строка фиксированной длины, дополненная пробелами
text	строка неограниченной переменной длины

- Логический тип:

Имя	Размер	Описание
boolean	1 байт	состояние: истина или ложь

Типы данных

- Дата и время :

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
`timestamp [ (p) ] [ without time zone ]`	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`timestamp [ (p) ] with time zone`	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`date`	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
`time [ (p) ] [ without time zone ]`	8 байт	время суток (без даты)	00:00:00	24:00:00	1 микросекунда / 14 цифр
`time [ (p) ] with time zone`	12 байт	только время суток (с часовым поясом)	00:00:00+1459	24:00:00-1459	1 микросекунда / 14 цифр
`interval [ поля ] [ (p) ]`	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда / 14 цифр

Типы данных

- Дата и время :

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
`timestamp [ (p) ] [ without time zone ]`	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`timestamp [ (p) ] with time zone`	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`date`	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
`time [ (p) ] [ without time zone ]`	8 байт	время суток (без даты)	00:00:00	24:00:00	1 микросекунда / 14 цифр
`time [ (p) ] with time zone`	12 байт	только время суток (с часовым поясом)	00:00:00+1459	24:00:00-1459	1 микросекунда / 14 цифр
`interval [ поля ] [ (p) ]`	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда / 14 цифр

Общий формат команды SELECT

SELECT [ALL | DISTINCT] *список_выборки*

FROM *источник_данных_запроса*

[WH ERE *условия_отбора_записей*]

[GROUP BY *список_полей_для_группировки*]

[HAVING *условия_отбора_групп*]

[ORDER BY список_*полей_для_сортировки* [ASC | DESC]]

Последовательность обработки фраз SELECT

1. FROM – определяются имена используемых таблиц;
2. WHERE – выполняется *фильтрация строк* объекта в соответствии с заданными условиями;
3. GROUP BY – образуются *группы строк* , имеющих одно и то же значение в указанном столбце;
4. HAVING – фильтруются группы строк объекта в соответствии с указанным условием;
5. SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
6. ORDER BY – определяется упорядоченность результатов выполнения операторов.

Элементы списка выборки

- * – все поля всех таблиц, указанных во фразе FROM
- <имя таблицы>.* – все поля указанной таблицы
- <имя таблицы>.<имя поля> – заданное поле таблицы
- <вычисляемое поле> – формулы, константы
- <поле> AS <псевдоним(alias)>

Элементы списка выборки

- SELECT name, quantity_in_stock FROM products;
- SELECT * FROM products;
- SELECT name, quantity_in_stock * price AS total_amount

FROM products;

- SELECT name, 1 AS one FROM products;
- SELECT company_name || ': ' || country

FROM companies;

DISTINCT

- Команда SELECT DISTINCT используется для выборки уникальных значений из одного или нескольких столбцов, автоматически удаляя повторяющиеся записи.
- Уникальные названия:
- SELECT DISTINCT name FROM products;
- Уникальные пары в столбцах:
- SELECT DISTINCT name, price FROM products;

Сортировка результирующего набора

- SELECT name, price FROM products

ORDER BY name, price DESC;

	name character varying (255)	price numeric (10,2)
1	Вода питьевая	25.00
2	Молоко 3.2%	75.50
3	Хлеб ржаной	55.00
4	Хлеб ржаной	35.00
5	Яйца куриные	120.00

- SELECT name, expiration_date

FROM products

ORDER BY expiration_date DESC;

-

	name character varying (255)	expiration_date date
1	Хлеб ржаной	[null]
2	Хлеб ржаной	[null]
3	Вода питьевая	[null]
4	Молоко 3.2%	2025-07-20
5	Яйца куриные	2025-07-20

Предложение WHERE

Формирует ***условие отбора*** ***записей в результирующий набор***

Типы ***условий*** :

- ***Сравнение***: сравниваются результаты вычисления одного выражения с результатами вычисления другого.
- ***Диапазон***: проверяется, попадает ли результат вычисления выражения в заданный *диапазон* значений.
- ***Принадлежность множеству***: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.
- ***Соответствие шаблону***: проверяется, отвечает ли некоторое строковое значение заданному шаблону.
- ***Значение*** **NULL**: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

Сравнение

- **Используются простые операторы сравнения (=; <; >; <=;**
- **>=; <>) и логические операторы AND, OR , NOT**

- **Получить информацию о продуктах, произведенных более трех дней назад и количество, которых в магазине больше 10**
- SELECT name, quantity_in_stock, production_date

FROM products

WHERE (CURRENT_DATE - production_date) > 3 AND quantity_in_stock > 10;

Диапазон

- **Оператор BETWEEN** используется для поиска значения внутри некоторого интервала.

- **Получить все товары, цене которых варьируется в диапазоне от 10 до 100 рублей**
- SELECT name, price

FROM products

WHERE price BETWEEN 10 AND 100;

Принадлежность

- **Оператор IN** используется для **сравнения** некоторого значения со списком заданных значений.
- **Получить товары из определенного перечня**
- SELECT name

FROM products

WHERE name IN ('Хлеб ржаной', 'Вода питьевая', 'Греча');

Соответствие шаблону

- **Оператор LIKE** выполняет *сравнение строкового* выражения с заданным шаблоном.
- **Стандарт SQL** определяет следующие символы шаблона:
  - Символ % –любое количество произвольных символов.
  - Символ _ заменяет один символ строки.
- Примеры:
  - 'abc' LIKE 'abc' true
  - 'abc' LIKE 'a%' true
  - 'abc' LIKE '_b_' true
  - 'abc' LIKE 'c' false
  - Получить названия продуктов, содержащих и не содержащих букву ‘o’:

```
SELECT name
FROM products
WHERE name LIKE '%o%';
```

```
SELECT name
FROM products
WHERE name NOT LIKE '%o%';
```

Трехзначная логика

- Трехзначная логика означает, что логические выражения могут иметь три возможных результата: TRUE (Истина), FALSE (Ложь) и UNKNOWN (Неизвестно). "Неизвестно", возникает из-за поддержки NULL-значений в SQL, которые представляют отсутствующие или неопределенные данные.

- Для работы с NULL в SQL используются специальные операторы:
  - IS NULL и IS NOT NULL для проверки, является ли значение NULL
  - Функция COALESCE возвращает первый попавшийся аргумент, отличный от NULL. Если же все аргументы равны NULL, результатом тоже будет NULL.
  - Функция NULLIF в PostgreSQL возвращает NULL, если два заданных выражения равны, и первое выражение, если они не равны

IS NULL, IS NOT NULL

- **Получить все данные о продуктах с ограниченным сроком годности**
- SELECT * FROM Products

WHERE expiration_date IS NOT NULL;

COALESCE

- SELECT COALESCE(NULL, NULL, 1, 2, NULL, 3)
- -- Результат 1
- Подстановки некоторого значения по умолчанию вместо значений NULL:

```
SELECT  name AS Название,
        production_date AS Дата_изготовления,
COALESCE(CAST(expiration_date AS TEXT), 'Отсутствует') AS Срок_годности
FROM products;
```

	Название character varying (255)	Дата_изготовления date	Срок_годности text
1	Молоко 3.2%	2025-07-10	2025-07-20
2	Хлеб ржаной	2025-07-12	Отсутствует
3	Хлеб ржаной	2025-07-11	Отсутствует
4	Яйца куриные	2025-07-10	2025-07-20
5	Вода питьевая	2025-07-10	Отсутствует

NULLIF

- **Получить названия продуктов, необходимо, чтобы цена, равная 0, отображалась как NULL**

```
SELECT
    name,
    NULLIF(price, 0) AS corrected_price
FROM  products;
```

Агрегатные функции

Название	Описание
COUNT(*)	Возвращает количество строк источника записей
COUNT	Возвращает количество значений в указанном столбце
SUM	Возвращает сумму значений в указанном столбце
AVG	Возвращает среднее значение в указанном столбце
MIN	Возвращает минимальное значение в указанном столбце
MAX	Возвращает максимальное значение в указанном столбце

Агрегатные функции

Поиск минимальной и максимальной цены товара:

```
SELECT  MIN(price) AS MinPrice, MAX(price) AS MaxPrice FROM Products;
```

Подсчет общего количества заказов:

```
SELECT COUNT(name) FROM Products;
```

Расчет общего количества товаров в магазине:

```
SELECT SUM(quantity_in_stock) AS TotalCount FROM Products;
```

Вычисление средней цены товаров:

```
SELECT AVG(Salary) AS AverageSalary FROM Products;
```

Группирование GROUP BY

```
SELECT   name,   COUNT(name)   AS order_count,

SUM(quantity_in_stock) AS count_product,   AVG(price)      AS avg_price,

MIN(price)  AS min_price,   MAX(price)  AS max_price

FROM products GROUP BY name;
```

	name character varying (255)	order_count bigint	count_product bigint	avg_price numeric	min_price numeric	max_price numeric
1	Геркулес	2	143	75.0000000000000000	75.00	75.00
2	Молоко 3.2%	1	100	75.5000000000000000	75.50	75.50
3	Вода питьевая	1	300	25.0000000000000000	25.00	25.00
4	Рис круглозерный	1	15	105.0000000000000000	105.00	105.00
5	Хлеб ржаной	2	100	45.0000000000000000	35.00	55.00
6	Греча	2	10	95.0000000000000000	95.00	95.00
7	Яйца куриные	1	200	120.0000000000000000	120.00	120.00
8	Рис длинозерный	1	100	90.0000000000000000	90.00	90.00

GROUP BY – правила использования

1. COUNT(*) подсчитывает все строки в таблице, включая строки с NULL значениями в любом столбце, тогда как COUNT(column_name) считает только строки, где указанный столбец не содержит NULL.

2. NULL значения игнорируются функциями SUM и AVG. Таким образом, они не влияют на суммирование или вычисление среднего значения.

3. Если совместно с GROUP BY используется предложение WHERE, то оно обрабатывается первым, а группировке подвергаются только те строки, которые удовлетворяют условию поиска.

4. При проведении группировки все значения NULL рассматриваются как равные

5.Непосредственно в WHERE использовать агрегатные функции нельзя. Для фильтрации агрегированных результатов используйте предложение HAVING.

GROUP BY – правила использования

6. Все имена полей, приведенные в списке предложения SELECT, должны присутствовать и во фразе GROUP BY - за исключением случаев, когда имя столбца используется в итоговой функции

Неправильно	Правильно
SELECT name, SUM(quantity_in_stock) FROM products;	SELECT name, SUM(quantity_in_stock) FROM products GROUP BY name;
SELECT name, price, SUM(quantity_in_stock) FROM products GROUP BY name;	SELECT name, price, SUM(quantity_in_stock) FROM products GROUP BY name, price;

HAVING

- При помощи HAVING фильтруются все предварительно сгруппированные посредством GROUP BY блоки данных, удовлетворяющие заданным в HAVING условиям.
- В условии могут использоваться поля и выражения, указанные в списке GROUP BY, а также выражения с агрегатными функциями.

HAVING

- **Получить список товаров, суммарное количество которых в магазине больше 100.**

```
SELECT name, SUM(quantity_in_stock)

FROM products

GROUP BY name

HAVING SUM(quantity_in_stock) > 100;
```

	name character varying (255)	sum bigint
1	Геркулес	143
2	Вода питьевая	300
3	Яйца куриные	200

HAVING

- В данном запросе HAVING применяется к неагрегированному полю quantity_in_stock, что некорректно, поскольку quantity_in_stock не является результатом агрегатной функции в данном контексте.

Неправильно	Правильно
SELECT name, SUM(quantity_in_stock) FROM products GROUP BY name HAVING quantity_in_stock > 100;	SELECT name, SUM(quantity_in_stock) FROM products GROUP BY name HAVING SUM(quantity_in_stock) > 100;

HAVING

- Здесь фильтрация по названию должна быть выполнена в WHERE, чтобы эффективно уменьшить объем обрабатываемых данных до группировки.

Неправильно	Правильно
SELECT name, SUM(quantity_in_stock) FROM products GROUP BY name HAVING name = 'Греча' AND SUM(quantity_in_stock) < 100;	SELECT name, SUM(quantity_in_stock) FROM products WHERE name = 'Греча‘ GROUP BY name HAVING SUM(quantity_in_stock) < 100;

Ограничение результирующего набора

- LIMIT задаёт максимальное число возвращаемых строк, а OFFSET — количество строк, которые нужно пропустить перед началом вывода.
- **Получить 5 записей из таблицы, начиная со второй.**

```
SELECT *
FROM products
LIMIT 5
OFFSET 1;
```

Что делает запрос?

- Таблица: Employees, Столбцы: EmployeeID, FirstName, LastName, Department, Salary, HireDate (дата приема на работу)

1. **SELECT * FROM Employees WHERE Department = 'Sales';**
2. **SELECT AVG(Salary) FROM Employees;**
3. **SELECT FirstName, LastName FROM Employees ORDER BY HireDate DESC;**
4. **SELECT Department, COUNT(*) FROM Employees GROUP BY Department;**

Что делает запрос?

- Таблица: Employees, Столбцы: EmployeeID, FirstName, LastName, Department, Salary, HireDate (дата приема на работу)

1. **SELECT * FROM Employees WHERE Department = 'Sales'; -** Выбирает всю информацию о сотрудниках, которые работают в отделе продаж.
2. **SELECT AVG(Salary) FROM Employees; -** Выводит среднюю заработную плату всех сотрудников в компании.
3. **SELECT FirstName, LastName FROM Employees ORDER BY HireDate DESC; -** Формирует список имен и фамилий сотрудников, отсортированный от самых новых к самым старым по дате приема на работу.
4. **SELECT Department, COUNT(*) FROM Employees GROUP BY Department; -** Показывает список всех отделов и количество сотрудников в каждом из них.

Что вернет запрос к таблице Books?

BookID	Title	Author	Price	InStock
101	Великий Гэтсби	Ф. Фицджеральд	10.99	Да
102	Солярис	С. Лем	12.50	Нет
103	Убийство в Восточном экспрессе	А. Кристи	8.75	Да
104	Дюна	Ф. Герберт	14.99	Да
105	Сияние	С. Кинг	9.99	Нет

```
SELECT COUNT(*) FROM Books WHERE InStock = 'Да';
```

Что вернет запрос к таблице Books?

BookID	Title	Author	Price	InStock
101	Великий Гэтсби	Ф. Фицджеральд	10.99	Да
102	Солярис	С. Лем	12.50	Нет
103	Убийство в Восточном экспрессе	А. Кристи	8.75	Да
104	Дюна	Ф. Герберт	14.99	Да
105	Сияние	С. Кинг	9.99	Нет

```
SELECT COUNT(*) FROM Books WHERE InStock = 'Да';
```

Ответ: 3

Спасибо за внимание!