МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
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 №3 Подзапросы и общетабличные выражения (CTE)

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 по кнопке "...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	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.

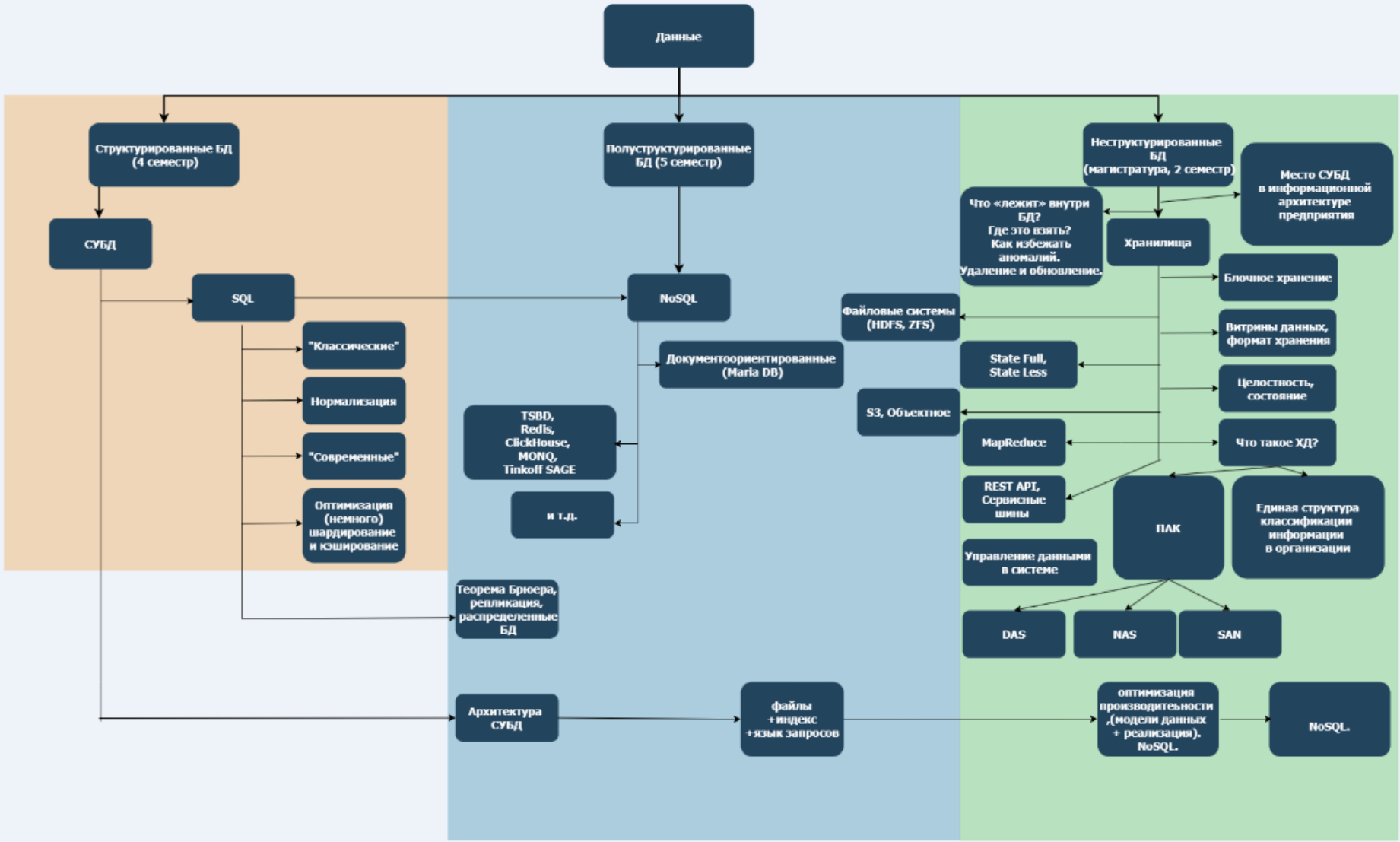
За тест студент может набрать 20 баллов, за которые можно получить:

- 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):

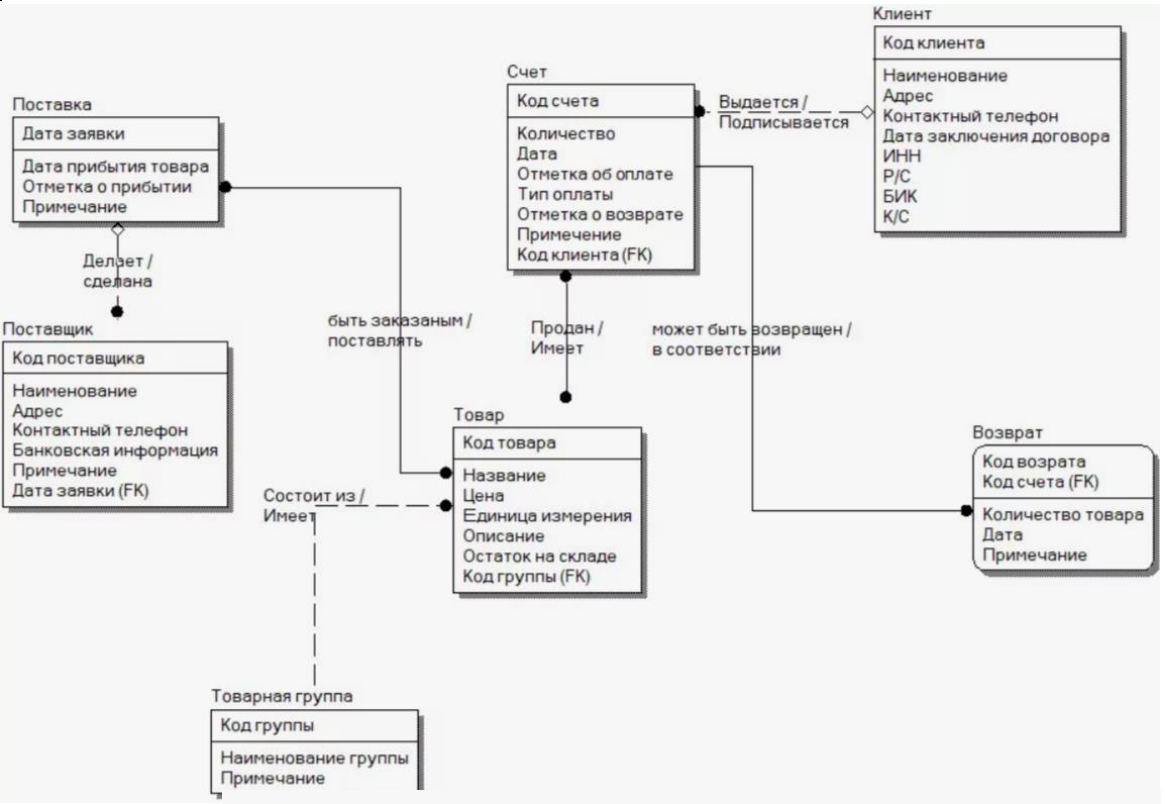
1) Место СУБД в информационной архитектуре **предприятия.**

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.



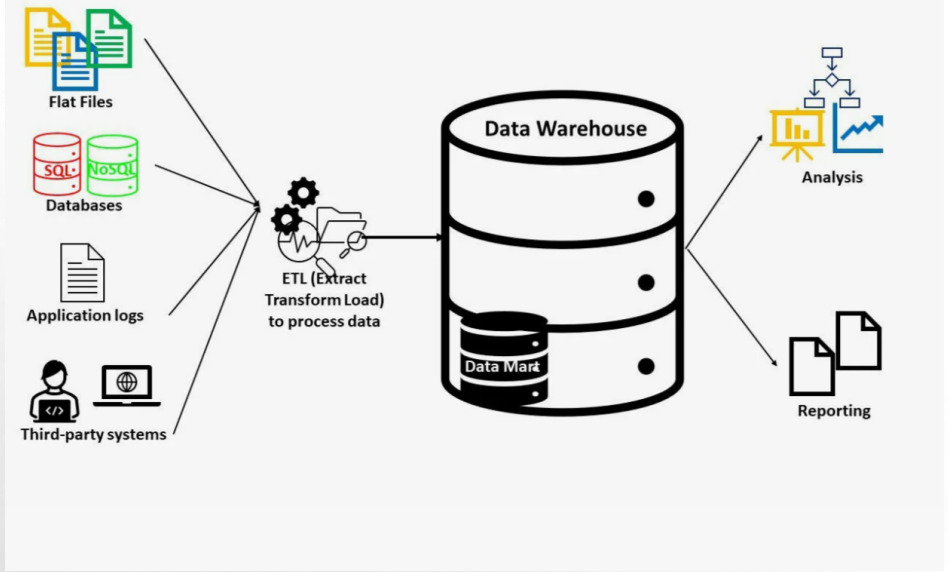
2 курс.

Проектирование баз данных

3 курс.

Разработка баз данных

PostgreSQL



Магистратура

Проектирование и разработка баз и хранилищ данных

Тема 3_ Подзапросы и общетабличные выражения Common Table Expressions

Функция CAST

CAST используется для явного преобразования значения из одного типа данных в другой:

```
SELECT CAST('123' AS INTEGER); ИЛИ SELECT '123' :: INTEGER;

SELECT CAST(123 AS VARCHAR); ИЛИ SELECT 123 :: VARCHAR;

SELECT CAST('2024-07-15' AS DATE);

SELECT
    'Средняя цена = ' || CAST(AVG(price) AS CHAR(15))
FROM products;
```

|| - операция конкатенации

Функции для работы с числовыми данными

Имя функции	Описание
POW(num, power)	Вычисляет число в указанной степени
SQRT(num)	Вычисляет квадратный корень числа
LOG(base, num)	Вычисляет логарифм числа по указанному основанию
EXP(num)	Вычисляет e^{num}
SIN(num)	Вычисляет синус числа
COS(num)	Вычисляет косинус числа
TAN(num)	Вычисляет тангенс числа

Функции округления

- Функции CEIL, FLOOR округляют число к ближайшему целому числу в большую и в меньшую сторону соответственно
- Функция ROUND – математическое округление

```
SELECT
        CEIL(69.69) AS ceiling,     FLOOR(69.69) AS floor,
        ROUND(69.499),              ROUND(69.501),
        ROUND(69.7171,1),           ROUND(69.7171,2),
        ROUND(69.7171,3),           ROUND(1691.7,-1),
        ROUND(1691.7,-2),           ROUND(1691.7,-3);
```

ceiling numeric	floor numeric	round numeric	round numeric	round numeric	round numeric	round numeric	round numeric	round numeric	round numeric
70	69	69	70	69.7	69.72	69.717	1690	1700	2000

Операция деления

Операция деления по умолчанию является целочисленным делением:

```
SELECT
        1/3 AS a,
        5/3 AS b;
```

Для того, чтобы эта операция выполнила обычное деление, необходимо сделать какой-либо из операндов вещественным:

```
SELECT
        cast(1 as DEC(12,4))/3 AS a,
        5./3 AS b;
```

Пример: Вычислить произведение элементов столбца. Гарантируется, что все значения в столбце положительные

```
SELECT  CAST(exp(sum(ln(quantity_in_stock))) AS numeric)
FROM products;
```

1. Вычисляет натуральный логарифм каждого значения в столбце quantity_in_stock:

ln(quantity_in_stock)

2. Логарифм произведения положительных чисел равен сумме логарифмов множителей:

sum(ln(quantity_in_stock)):

3. Вычисляем произведение:

exp(sum(ln(quantity_in_stock)))

Функции для работы со датой и временем

CURRENT_DATE – возвращает текущую дату,
CURRENT_TIME - возвращает текущее время вместе с локальным часовым поясом,
CURRENT_TIMESTAMP - возвращает текущую даты и время, а также локальный часовой пояс.

SELECT
CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, NOW();

current_date date	current_time time with time zone	current_timestamp timestamp with time zone	now timestamp with time zone
2025-07-15	13:51:58.522132+03:00	2025-07-15 13:51:58.522132+03	2025-07-15 13:51:58.522132+03

Функции для работы со датой и временем

TO_DATE - конвертирует строковый литерал в значение даты

Синтаксис:

TO_DATE(текст, формат_даты);

Функция TO_DATE() принимает два параметра.

```
SELECT
    TO_DATE('20210910','YYYYMMDD'),
    TO_DATE('10 Sep 2021', 'DD Mon YYYY');
```

to_date date	to_date date
2021-09-10	2021-09-10

TO_TIMESTAMP() - функция преобразует строковый литерал в метку времени на основе указанного формата

```
SELECT TO_TIMESTAMP(
   '2021-09-10 11:30:20',
   'YYYY-MM-DD HH:MI:SS');
```

	to_timestamp timestamp with time zone
1	2021-09-10 11:30:20+03

Функции для работы со датой и временем

EXTRACT() - извлекает определенную часть даты (например, год, месяц, день)

```
SELECT
  Current_Date
 ,EXTRACT(Year from Current_Date) as Yr
 ,EXTRACT(Month from Current_Date) as Mo
 ,EXTRACT(Day from Current_Date) as Da
 ,Current_Time
 ,EXTRACT(Hour from Current_Time) as Hr
 ,EXTRACT(Minute from Current_Timestamp) as Mn
 ,EXTRACT(Second from Current_Timestamp) as Sc ;
```

date	yr	mo	da	timetz	hr	mn	sc
2021-09-14	2021	9	14	14:18:03.966858+00	14	18	3.97

Функции для работы со датой и временем

Age() – вычисляет разницу двух дат:

```
SELECT
        AGE('2021-01-01','2018-10-24'),
        AGE(timestamp '1989-03-09');
```

!Во втором случае разницу с ТЕКУЩЕЙ датой

age interval	age interval
2 years 2 mons 8 days	36 years 4 mons 6 days

Строковые функции

Функция	Назначение
TRIM(), LTRIM() и RTRIM()	Удаление пробелов
SUBSTRING()	Извлечение подстрок
REPLACE()	Замена подстрок
CONCAT()	Объединение строк
LPAD() и RPAD()	Выравнивание строк
LOWER() и UPPER()	Преобразование регистра
LENGTH()	Определение длины строки

Строковые функции

```
SELECT
  LENGTH('hello'),
  LOWER('HELLO'),
  TRIM('  hello  '),
  SUBSTRING('hello world' FROM 7),
  SUBSTRING('hello world' FROM 1 FOR 5);
```

length integer	lower text	btrim text	substring text	substring text
5	hello	hello	world	hello

```
SELECT
  REPLACE('hello world', 'world', 'PostgreSQL'),
  CONCAT('hello', ' ', 'world'),
  RPAD('hello', 10, '*'),
  LPAD('5', 5, '0');
```

replace text	concat text	rpad text	lpad text
hello PostgreSQL	hello world	hello*****	00005

Условная логика CASE

Проверка условий и возвращение одного из нескольких возможных результатов.

Виды:

- **простое выражение** CASE для определения результата сравнивает выражение с набором значений (только проверка равенства);
- **поисковое выражение** CASE для определения результата вычисляет набор логических выражений.

Выражение CASE может использоваться в любой инструкции или предложении, где ожидается скалярный результат

Условная логика CASE

1. *Простое выражение CASE*

```
CASE input_expression
WHEN when_expression
THEN result_expression [ ...n ]
[ ELSE else_result_expression ]
END
```

2. Поисковое выражение CASE

```
CASE
WHEN Boolean_expression
THEN result_expression [ ...n ]
[ ELSE else_result_expression ]
END
```

Условная логика CASE

- **input_expression** - любое допустимое выражение.

• **when_expression** - простое выражение, с которым сравнивается input_expression при использовании простого формата функции CASE. Типы данных аргумента input_expression и каждого из выражений when_expression должны быть одинаковыми или неявно приводимыми друг к другу.

• **result_expression input_expression** - выражение, возвращаемое, когда равенство и when_expression имеет значение TRUE или Boolean_expression имеет значение TRUE

• **else_result_expression** - выражение, возвращаемое, если ни одна из операций сравнения не дает в результате TRUE. Если этот аргумент опущен и ни одна из операций сравнения не дает в результате TRUE, функция CASE возвращает NULL. Типы данных аргумента else_result_expression и любого из аргументов result_expression должны быть совместимыми.

• **boolean_expression** - логическое выражение, полученное при использовании поискового формата функции CASE. Любое допустимое логическое выражение.

Использование CASE в списке выборки

Получить список товаров, указав их дату производства и срок годности, если срок годности отсутствует, сообщить об этом

```
SELECT
    name AS Название,
    production_date AS Дата_изготовления,
    CASE
        WHEN expiration_date IS NULL
            THEN  'Отсутствует'
        ELSE
            CAST(expiration_date AS TEXT)
    END  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
6	Рис длинозерный	2025-07-01	Отсутствует
7	Рис круглозерный	2025-07-05	Отсутствует
8	Геркулес	2025-07-05	Отсутствует
9	Геркулес	2025-07-05	Отсутствует
10	Греча	2025-06-01	Отсутствует

Пример 1. Распределим продукты на группы: для товаров с ценой > 1000 выведем – Дорогой, между 100 и 1000 – Средняя цена, меньше 100 – Дешевый

```
SELECT name, price,
        CASE
        WHEN price > 1000
        THEN 'Дорогой'
        WHEN price between 100 and 1000
        THEN 'Средняя цена'
        WHEN price < 100 THEN 'Дешевый'
        END AS result
FROM products;
```

	name character varying (255)	price numeric (10,2)	result text
1	Молоко 3.2%	75.50	Дешевый
2	Хлеб ржаной	35.00	Дешевый
3	Хлеб ржаной	55.00	Дешевый
4	Яйца куриные	120.00	Средняя цена
5	Вода питьевая	25.00	Дешевый
6	Рис длинозерный	90.00	Дешевый
7	Рис круглозерный	105.00	Средняя цена
8	Греча	95.00	Дешевый
9	Геркулес	75.00	Дешевый
10	Греча	95.00	Дешевый
11	Геркулес	75.00	Дешевый

Пример 2 (вложенные CASE). Добавим в предыдущий пример возможность для каждой категории продавать товары со скидкой в зависимости от того, имеется у товара срок годности и сколько дней от текущей даты осталось до него. У Дорогих товаров скидка будет больше, чем у дешевых.

```
SELECT name, price,
CASE   WHEN price > 1000 THEN
            CASE WHEN expiration_date IS NOT NULL
            THEN     CASE     WHEN expiration_date - CURRENT_DATE < 2
                     THEN     CAST(price * 0.8 AS VARCHAR(20))
                     ELSE     CAST(price * 0.85 AS VARCHAR(20))
                     END
            ELSE     'Цена без скидки'
            END
END AS Новая_цена
……..
FROM products;
```

	name character varying (255)	price numeric (10,2)	Новая_цена character varying
1	Молоко 3.2%	75.50	71.73
2	Хлеб ржаной	35.00	Цена без скидки
3	Хлеб ржаной	55.00	Цена без скидки
4	Яйца куриные	120.00	108.000
5	Вода питьевая	25.00	Цена без скидки
6	Рис длинозерный	90.00	Цена без скидки
7	Рис круглозерный	105.00	Цена без скидки
8	Греча	95.00	Цена без скидки
9	Геркулес	75.00	Цена без скидки
10	Греча	95.00	Цена без скидки
11	Геркулес	75.00	Цена без скидки

Использование CASE в условиях

Пример. Отбираем те продукты, у которых цена больше определенного значения (50 или 100). При этом с первым значением сравниваем, если количество товара в магазине не больше 100, со вторым – больше 100

```
SELECT name, price, quantity_in_stock
FROM products
WHERE price >
        CASE
        WHEN quantity_in_stock <= 100
        THEN 50
        ELSE 100

        END
```

	name character varying (255)	price numeric (10,2)	quantity_in_stock integer
1	Молоко 3.2%	75.50	100
2	Хлеб ржаной	55.00	50
3	Яйца куриные	120.00	200
4	Рис длинозерный	90.00	100
5	Рис круглозерный	105.00	15
6	Греча	95.00	10
7	Геркулес	75.00	13
8	Греча	95.00	0

Использование CASE с агрегатными функциями

Получить по каждому виду товара общее его количество в магазине, кроме товаров с количество меньше 100. По ним подсчитать общую сумму

```
SELECT   CASE  WHEN quantity_in_stock < 100
                THEN 'Остатки в магазине'
                ELSE name
END AS Products,
SUM(quantity_in_stock) AS Remaining_goods
FROM products
GROUP BY CASE WHEN quantity_in_stock < 100
                THEN 'Остатки в магазине'
                ELSE    name
                END;
```

	products character varying	remaining_goods bigint
1	Геркулес	130
2	Молоко 3.2%	100
3	Вода питьевая	300
4	Яйца куриные	200
5	Рис длинозерный	100
6	Остатки в магазине	148

Подзапросы

Подзапросом называется законченный оператор SELECT, внедренный в тело другого оператора SELECT

• Применяются, когда при использовании условия поиска в предложении WHERE или HAVING значение, с которым надо сравнивать, заранее не определено и должно быть вычислено в момент выполнения оператора SELECT.

• Текст подзапроса должен быть заключен в скобки.

• Фраза ORDER BY не используется

• Допускается ссылка на столбцы таблицы, указанной во фразе FROM внешнего запроса (коррелированный подзапрос).

Два типа подзапросов:

• **Скалярный подзапрос** возвращает единственное значение.

• **Табличный подзапрос** возвращает значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке.

Применение подзапросов в основном запросе

Примеры использования скалярных подзапросов

Использование в SELECT внешнего запроса:

Вывести на экран все продукты и максимальную цену во всем магазине

```
SELECT *,  (SELECT MAX(price) FROM products)  FROM products;
```

Использование в FROM внешнего запроса:

Выведется на экран значение максимальной цены товара в магазине

```
SELECT *  FROM (SELECT MAX(price) FROM products);
```

Использование в WHERE внешнего запроса:

Получить список товаров, у которых цена равна максимальной цене в магазине

```
SELECT * FROM products
WHERE price = (SELECT MAX(price) FROM products)
```

Примеры использования скалярных подзапросов

Использование в HAVING внешнего запроса:

Получить те группы товаров в магазине, у которых средняя цена больше средней цены в магазине:

```
SELECT name, AVG(price)
FROM products
GROUP BY name
HAVING AVG(price) >
(SELECT AVG(price) FROM products);
```

Использование в ORDER BYвнешнего запроса:

Получить список товаров, отсортированный по отклонению цены товара от средней цены в магазине (по возрастанию отклонения)

```
SELECT name, price, price - (SELECT AVG(price) FROM products)
FROM products
ORDER BY price - (SELECT AVG(price) FROM products);
```

Табличные подзапросы

Вложенные табличные подзапросы генерируют промежуточное отношение - временную таблицу.

Общий формат:

• { WHERE | HAVING } выражение [NOT] IN (подзапрос);

• { WHERE | HAVING } выражение оператор_сравнения { ALL | SOME | ANY } (подзапрос);

• {WHERE | HAVING } [NOT] EXISTS (подзапрос);

Оператор IN

Оператор IN используется для сравнения некоторого выражения со множеством значений, формируемым вложенным запросом.

Получить названия компаний, у которых в магазине осталось менее 50 шт товаров:

```
SELECT company_name
FROM companies
WHERE company_id IN
        (SELECT DISTINCT company_id
        FROM products
        WHERE quantity_in_stock < 50);
```

	company_name character varying (255)
1	ИП "Хлебный рай"

JOIN?

Ключевые слова ANY и ALL

Ключевые слова ANY и ALL могут использоваться с подзапросами, возвращающими один столбец значений.

• Если подзапросу будет предшествовать ключевое слово ALL, условие сравнения считается выполненным, только когда оно выполняется для всех значений в результирующем столбце подзапроса.

• Если подзапросу предшествует ключевое слово ANY (SOME), то условие сравнения считается выполненным, когда оно выполняется хотя бы для одного из значений в результирующем столбце подзапроса.

• Если в результате выполнения подзапроса получено пустое множество, то для ключевого слова ALL условие сравнения будет считаться выполненным, а для ключевого слова ANY – невыполненным.

Получить товары, цена которых превышает цену хотя бы одного товара с количеством менее 50 шт в магазине

```
SELECT name, price, quantity_in_stock
FROM products
WHERE price > ANY
(SELECT price
FROM products WHERE quantity_in_stock < 50);
```

Получаем цены всех товаров с количеством < 50

	price numeric (10,2)
1	105.00
2	75.00
3	95.00
4	95.00

name character varying (255)	price numeric (10,2)	quantity_in_stock integer
Молоко 3.2%	75.50	100
Яйца куриные	120.00	200
Рис длинозерный	90.00	100
Рис круглозерный	105.00	15
Греча	95.00	10
Греча	95.00	10

Получить товары, цена которых превышает цену всех товаров с количеством менее 50 шт в магазине

```
SELECT name, price, quantity_in_stock
FROM products
WHERE price >
ALL (SELECT price
FROM products
WHERE quantity_in_stock < 50);
```

	name character varying (255)	price numeric (10,2)	quantity_in_stock integer
1	Яйца куриные	120.00	200

Предикаты EXISTS и NOT EXISTS

Синтаксис:

[NOT] EXISTS (<табличный подзапрос>)

Предикат EXISTS принимает значение TRUE, если подзапрос содержит любое количество строк, иначе его значение равно FALSE.

Для NOT EXISTS все наоборот.

Аргументом EXISTS является обычный оператор SELECT, т.е. подзапрос.

Так как результат этого выражения зависит только от того, возвращаются строки или нет, но не от их содержимого, список выходных значений подзапроса обычно не имеет значения:

EXISTS(SELECT 1 WHERE ...)

Полусоединения

Полусоединение (SEMI JOIN) – называют такую операцию соединения двух таблиц, при которой в результат включаются только строки из **первой** таблицы, удовлетворяющие условию наличия соответствий во второй, а данные второй таблицы при этом не добавляются к результату

Общий синтаксис:

```
SELECT ... FROM
TableA WHERE [EXISTS|IN] (
    SELECT ...
    FROM TableB
    WHERE условие);
```

где подзапрос проверяет наличие соответствующей записи в TableB.

```
SELECT * FROM companies WHERE
EXISTS
(SELECT 1 FROM products WHERE
companies.company_id =
products.company_id);
```

	company_id [PK] integer	company_name character varying (255)	country character varying (100)
1	1	ООО "Молочная река"	Россия
2	2	ИП "Хлебный рай"	Россия

Коррелированный подзапрос

Коррелированный подзапрос - это подзапрос, который обращается к данным из внешнего запроса. Главное его отличие от обычного подзапроса в том, что он выполняется для каждой строки основного запроса.

Получить для каждой компании – поставщика товаров – сумму прибыли:

```
SELECT companies.company_name,
(SELECT SUM(products.price*
products.quantity_in_stock)
    FROM products
    WHERE products.company_id =
companies.company_id ) AS summa
FROM companies;
```

	company_name character varying (255)	summa numeric
1	ООО "Молочная река"	31550.00
2	ИП "Хлебный рай"	35200.00

JOIN?

Коррелированный подзапрос

Получить список всех покупателей, в заказах которых имеются все продукты из магазина.

Способ, основанный на операции разности:
Если взять операцию **разности ВСЕХ имеющихся продуктов и продуктов, купленных конкретным покупателем**, то результирующая выборка не должна содержать строк.

Все товары

Товары текущего покупателя

```
SELECT c1.customer_id, c1.last_name
FROM customers AS c1
WHERE 0 = (SELECT COUNT(*)
    FROM
    (SELECT p.id FROM Products p
    EXCEPT
        SELECT o.product_id
        FROM  orders ord
        JOIN  order_items o
        ON o.order_id = ord.order_id
    WHERE c1.customer_id = ord.customer_id)
);
```

	customer_id [PK] integer	last_name character varying (255)
1	1	Иванов

Коррелированный подзапрос

Получить список всех покупателей, в заказах которых имеются все продукты из магазина.

Способ, основанный на существовании:
Не должно существовать такого типа продукции, которого бы не было у искомого покупателя.

Товары, отсутствующие у текущего покупателя

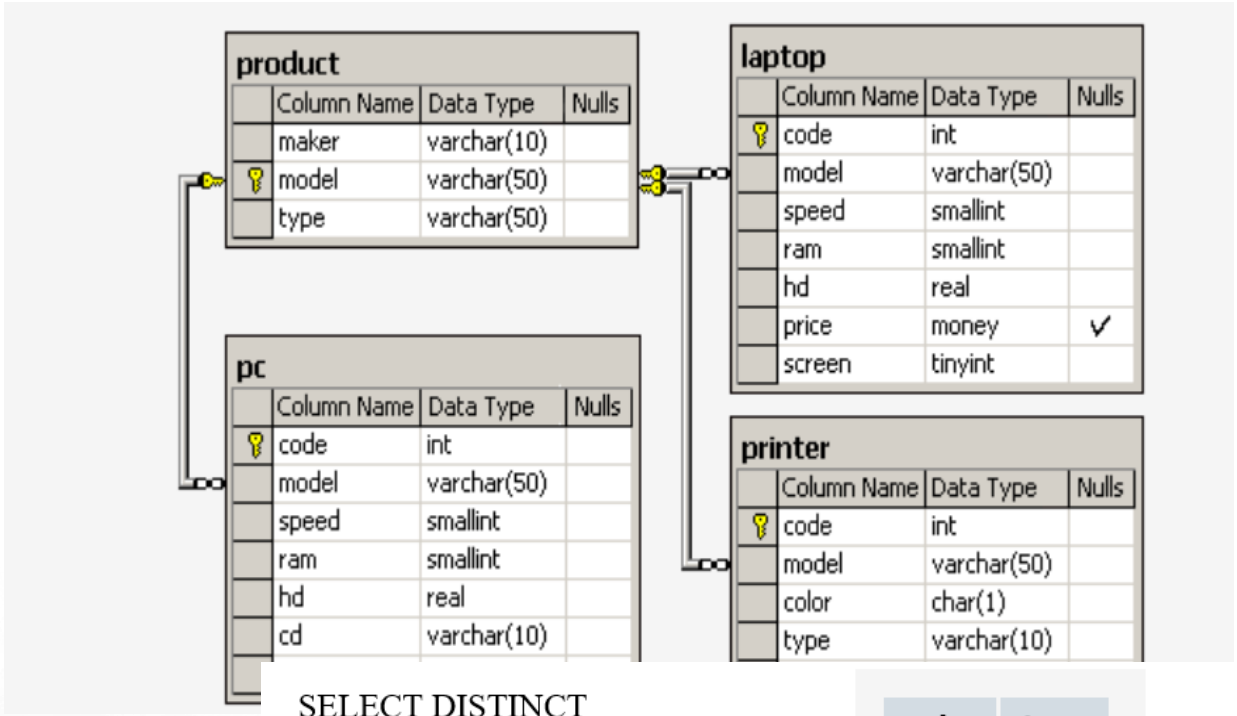
```
SELECT c1.customer_id, c1.last_name
FROM customers AS c1
WHERE NOT EXISTS (SELECT p1.id
        FROM products p1
        WHERE p1.id NOT IN
            (SELECT o.product_id
            FROM   orders ord
            JOIN  order_items o
            ON o.order_id = ord.order_id
    WHERE c1.customer_id = ord.customer_id));
```

Товары текущего покупателя

	customer_id [PK] integer	last_name character varying (255)
1	1	Иванов

Для каждого типа продукции и каждого производителя из таблицы Product с точностью до двух десятичных знаков найти процентное отношение числа моделей данного типа данного производителя к общему числу моделей этого производителя.

Вывод: maker, type, процентное отношение числа моделей данного типа к общему числу моделей производителя.




```
SELECT DISTINCT
    A.maker, B.type
FROM
    Product A, Product B
```

maker	type
A	Laptop
A	PC
A	Printer
B	Laptop
B	PC
B	Printer

```
SELECT DISTINCT A.maker, B.type,
(SELECT COUNT(*) FROM Product
WHERE Product.maker = A.maker AND
Product.type = B.type )
FROM Product A, Product B
```

maker	type	
A	Laptop	2
A	PC	2
A	Printer	3
B	Laptop	1
B	PC	1
B	Printer	0

```
SELECT DISTINCT
A.maker,
(SELECT COUNT(*) FROM Product
    WHERE Product.maker = A.maker)
FROM Product A, Product B
```

maker	
A	7
B	2
C	1
D	2
E	4

```
SELECT maker, type,

        CASE WHEN C IS NULL THEN 0     ELSE CAST(ROUND(C, 2) AS NUMERIC(6, 2)) END

FROM    (SELECT DISTINCT A.maker, B.type, 100.0 *

                        (SELECT COUNT(*) FROM Product

                        WHERE Product.maker = A.maker AND Product.type = B.type ) /

                        (SELECT COUNT(*) FROM Product

                        WHERE Product.maker = A.maker) AS C

        FROM    Product A, Product B

        ) D
```

Правильно.

Результат выполнения Вашего запроса:

maker	type	
A	Laptop	28.57
B	Laptop	50.00
C	Laptop	100.00
D	Laptop	.00
E	Laptop	.00
A	PC	28.57
B	PC	50.00
C	PC	.00
D	PC	.00

Общетабличные выражения CTE - COMMON TABLE EXPRESSIONS

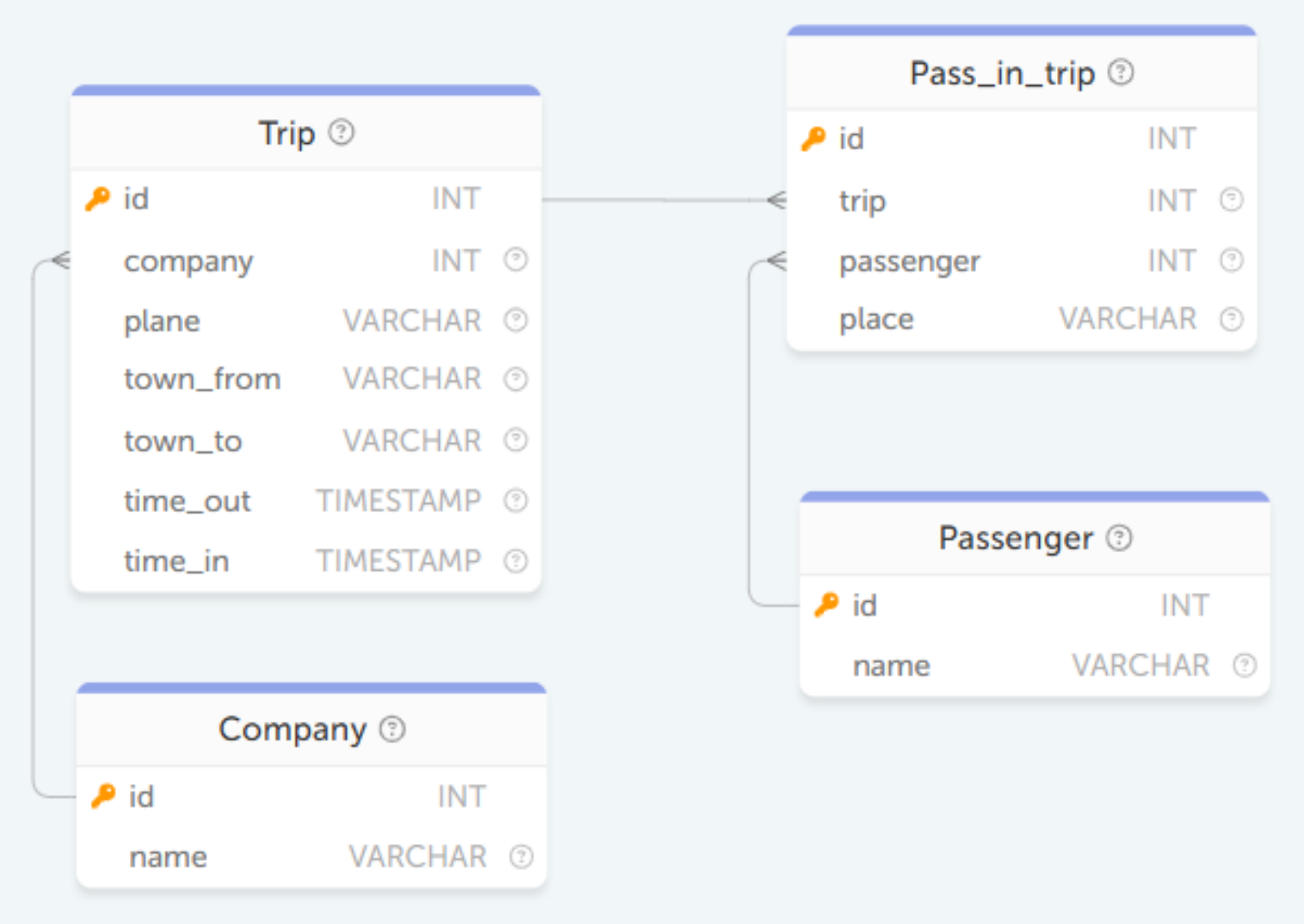
Обобщённое табличное выражение или CTE (Common Table Expressions) - это временный результирующий набор данных, к которому можно обращаться в последующих запросах:

- существует только в рамках текущего запроса и не сохраняется в базе данных
- позволяет разбить сложные запросы на более мелкие логические части, что улучшает читаемость и упрощает отладку.
- можно использовать в SELECT, INSERT, UPDATE, DELETE операторах.

Синтаксис:

```
WITH
        название_cte [(столбец_1 [, столбец_2 ] …)]
       AS (подзапрос)
       [, название_cte [(столбец_1 [, столбец_2 ] …)]
       AS (подзапрос)] …
```

Общетабличные выражения CTE



1) Выведите пассажиров с самым длинным ФИО. Пробелы, дефисы и точки считаются частью имени.
2) Выведите нагруженность (число пассажиров) каждого рейса (trip). Результат вывести в отсортированном виде по убыванию нагруженности.

Общетабличные выражения CTE

1) WITH cte AS(

SELECT name, LENGTH(name) as len

FROM Passenger)

SELECT name

FROM cte

WHERE len =

(SELECT MAX(len) FROM cte);

2) WITH cte AS

(SELECT Trip.id AS Trip FROM Trip

JOIN Pass_in_trip

ON Trip.id = Pass_in_trip.trip)

SELECT trip, COUNT(*) AS count FROM cte

GROUP BY trip

ORDER BY count DESC;

Рекурсивный CTE

Общий синтаксис:

WITH RECURSIVE <имя>[(<список столбцов>)]

AS(

<SELECT...> -- анкорная часть

UNION ALL -- рекурсивная часть

<SELECT...FROM <имя>…>

WHERE <условие продолжения итераций>

)

Рекурсивное CTE состоит из двух частей, разделенных оператором UNION ALL:

- Начальный набор данных, который не содержит рекурсивных ссылок.
- Рекурсивная часть: запрос, который ссылается на CTE, чтобы продолжить рекурсию.

Пример. Сумма чисел от 1 до 100:

```
WITH RECURSIVE t(n) AS
(
 SELECT 1
UNION ALL
SELECT n+1 FROM t
WHERE n < 100
)

SELECT sum(n) FROM t;
```

Рекурсивный CTE

- В рекурсивной части должна присутствовать ссылка на имя CTE (CTE ссылается само на себя).
- Анкорный и рекурсивный запросы должны иметь одинаковый набор столбцов.
- Для управления глубиной рекурсии в CTE (Common Table Expression) с помощью WITH RECURSIVE используется параметр max_recursive_steps (по умолчанию 100), но его можно изменить в конфигурации PostgreSQL или в рамках конкретного запроса. (SET LOCAL max_recursive_steps = 500;)
- Если глубина рекурсии не ограничена, рекурсивный запрос может продолжаться до тех пор, пока не будет достигнут предел, установленный параметром max_stack_depth. Это может привести к ошибке stack overflow.

Списки VALUES

VALUES позволяет создать «постоянную таблицу», которую можно использовать в запросе, не создавая и не наполняя таблицу в БД.

Синтаксис: **VALUES (выражение [, ...]) [, ...]**

Таблица из двух столбцов и трех строк:

```
VALUES (1, 'one'), (2, 'two'), (3, 'three');
```

Это равносильно такому запросу:

```
SELECT 1 AS column1, 'one' AS column2
UNION ALL
SELECT 2, 'two‘
UNION ALL
SELECT 3, 'three';
```

```
SELECT * FROM
(VALUES (1, 'one'), (2, 'two'), (3, 'three'))
AS t (num,letter);
```

	num integer	letter text
1	1	one
2	2	two
3	3	three

Рекурсивный CTE

Пример. Сумма чисел от 1 до 100:

```
WITH RECURSIVE t(n) AS
(
VALUES(1)
UNION ALL
SELECT n+1 FROM t
WHERE n < 100
)

SELECT sum(n) FROM t;
```

Пример. Использование рекурсивного CTE для генерации ряда чисел Фибоначчи:

```
WITH RECURSIVE fib(i, a, b) AS (
   VALUES(0, 0, 1)
  UNION ALL
   SELECT i + 1, b, a + b
   FROM fib
   WHERE i < 10
)

SELECT * FROM fib;
```

	i integer	a integer	b integer
1	0	0	1
2	1	1	1
3	2	1	2
4	3	2	3
5	4	3	5
6	5	5	8
7	6	8	13
8	7	13	21
9	8	21	34
10	9	34	55
11	10	55	89

Рекурсивный CTE

Рекурсивные запросы обычно применяются для работы с иерархическими или древовидными структурами данных.

Пример. Дана таблица Employees, содержащая имена сотрудников и идентификаторы их менеджеров. Требуется вывести всех подчиненных руководителя подразделения.

employee_id [PK] integer	first_name character varying (255)	last_name character varying (255)	phone_number character varying (20)	manager_id integer
1	Иван	Сорокин	+7(905)123-44-56	[null]
2	Алексей	Петров	+7(905)123-44-56	1
3	Марина	Иванова	+7(905)123-44-56	1
4	Евгений	Смирнов	+7(905)123-44-56	2
5	Александр	Орлов	+7(905)123-44-56	3

Рекурсивный CTE

```
WITH RECURSIVE Subordinates AS (
SELECT
      e.employee_id,       e.first_name,
      e.last_name,       e.manager_id,
      m.last_name
       AS manager_last_name
FROM       Employee e
LEFT JOIN       Employee m
ON e.manager_id = m.employee_id
WHERE       e.manager_id = 1
UNION ALL
SELECT
       em.employee_id,       em.first_name,
      em.last_name,       em.manager_id,
      s.last_name
 FROM       Employee AS em
INNER JOIN       Subordinates s
ON em.manager_id = s.employee_id)
```

employee_id integer	first_name character varying (255)	last_name character varying (255)	manager_id integer	manager_last_name character varying (255)
2	Алексей	Петров	1	Сорокин
3	Марина	Иванова	1	Сорокин
4	Евгений	Смирнов	2	Петров
5	Александр	Орлов	3	Иванова

Что вычисляет запрос?

```
SELECT
    department,
    COUNT(*) AS total_employees,
    SUM( CASE WHEN salary > 80000 THEN 1 ELSE 0 END ) AS salary_count
FROM Employees
GROUP BY department;
```

Что вычисляет запрос?

```
SELECT
    department,
    COUNT(*) AS employees,
    SUM( CASE WHEN salary > 80000 THEN 1 ELSE 0 END ) AS salary_count
FROM Employees
GROUP BY department;
```

Количество работников в каждом отделе и количество работников в каждом отделе, имеющих зарплату больше 80000

Что вернет указанный запрос?

В таблице Employees 50 сотрудников. Предположим, что у 10 из них salary IS NULL.

```
SELECT employee_name,
CASE        WHEN salary > 100000 THEN 'High'
            WHEN salary > 50000 THEN 'Medium'
            ELSE 'Low'
END as category FROM Employees
WHERE CASE        WHEN salary > 100000 THEN 'High'
                  WHEN salary > 50000 THEN 'Medium'
                  ELSE 'Low'
END = 'Low';
```

Что вернет указанный запрос?

В таблице Employees 50 сотрудников. Предположим, что у 10 из них salary IS NULL.

```
SELECT employee_name,
CASE        WHEN salary > 100000 THEN 'High'
            WHEN salary > 50000 THEN 'Medium'
            ELSE 'Low'
END as category FROM Employees
WHERE CASE        WHEN salary > 100000 THEN 'High'
                  WHEN salary > 50000 THEN 'Medium'
                  ELSE 'Low'
END = 'Low';
```

Вернет сотрудников, попавших в категорию Low: тех у кого salary IS NULL или <=50000

Что вернет запрос?

Таблица products:

product_id	category	price
1	Electronics	
2	Electronics	
3	Books	
4	Electronics	
5	Books	
6	Clothing	

```
SELECT product_id, price
FROM products
WHERE category IN (
    SELECT category
    FROM products
    GROUP BY category
    HAVING AVG(price) > 300
)
AND price < (
    SELECT MAX(price)
    FROM products
    WHERE category = 'Electronics'
);
```

Что вернет запрос?

Таблица products:

product_id	category	price
1	Electronics	1000
2	Electronics	500
3	Books	50
4	Electronics	800
5	Books	30
6	Clothing	100

product_id	price
2	500
4	800

```
SELECT product_id, price
FROM products
WHERE category IN (
    SELECT category
    FROM products
    GROUP BY category
    HAVING AVG(price) > 300
)
AND price < (
    SELECT MAX(price)
    FROM products
    WHERE category = 'Electronics'
);
```

Спасибо за внимание!