Текст доклада для защиты

Слайд 1: Титульный (30 секунд)

Добрый день, уважаемые коллеги!
Представляю вашему вниманию курсовой проект по разработке системы хранения и аналитики почтовых данных на PostgreSQL.

Проект демонстрирует применение современных подходов к работе с большими объемами структурированных и неструктурированных данных.

Слайд 2: Проблема и Цель (1.5 минуты)

В современной корпоративной среде почтовая система генерирует огромные объемы данных - терабайты писем, вложений, метаданных.
Традиционно эти данные хранятся разрозненно: метаданные в одной системе, файлы в другой, что приводит к серьезным проблемам.

Во-1, медленный поиск - найти нужное письмо по содержанию или метаданным становится настоящей задачей.
Во-вторых, отсутствие единой платформы для аналитики - невозможно быстро получить статистику по переписке,
анализировать коммуникации между сотрудниками, строить отчеты.

Целью моего проекта стало создание единого реляционного хранилища, которое решает эти проблемы.
Я ставил задачу не просто хранить данные, а обеспечить быстрый поиск, глубокую аналитику и при этом продемонстрировать
ключевые приемы оптимизации в PostgreSQL.

Слайд 3: Архитектура решения (1.5 минуты)

Я разделил хранение метаданных и самих файлов.

Метаданные писем - информация об отправителе, получателе, теме, дате отправки - хранятся в PostgreSQL.
Это позволяет использовать всю мощь реляционной модели: связи, транзакции, сложные запросы.

А вот сами MIME-файлы - то есть оригинальные письма со всей структурой - хранятся в S3-совместимом хранилище.
Это экономически эффективно и надежно.

Связь между ними осуществляется через object_id - каждая запись в таблице сообщений содержит ссылку на соответствующий объект в S3.
Такое разделение ответственности дает лучшее из двух миров: скорость реляционной БД для запросов и
масштабируемость объектного хранилища для больших файлов.

Слайд 4: Схема данных - Ядро системы (2 минуты)

Подчеркнуть продуманность схемы

Я разработал нормализованную схему из 15 взаимосвязанных таблиц.

Ключевые сущности включают:

- Таблицу пользователей с JSONB-профилями, что позволяет гибко хранить дополнительную информацию о сотрудниках
- Иерархическую систему папок через таблицу tags с поддержкой родительских связей
- Таблицу сообщений, которая связывает письма с папками и файлами в S3
- Отдельную таблицу для MIME-частей, что позволяет работать со структурой писем

Особенностью схемы является тщательная нормализация.
Например, email-адреса вынесены в отдельную таблицу, что исключает дублирование и обеспечивает целостность данных.

Также я реализовал полное журналирование: каждая смена статуса письма, каждое действие пользователя записывается в соответствующие таблицы. Это не только для аудита, но и для последующей аналитики.

Слайд 5: Оптимизация - Система индексов (2 минуты)

Для операций аутентификации и навигации я использовал B-tree индексы - например, индекс по логину пользователя
для быстрой аутентификации и индекс по связке пользователь-родительская папка для эффективной навигации по иерархии.

Для специализированных задач я выбрал соответствующие типы индексов:
-GIN индекс для JSONB-поля с профилями пользователей
-BRIN индекс для временных меток в таблице статусов - это особенно эффективно, так как данные вставляются последовательно во времени
-Уникальные индексы с использованием COALESCE для обеспечения уникальности имен папок, даже для корневых папок с NULL parent_id

Такой осознанный выбор типов индексов позволил мне значительно ускорить выполнение запросов при минимальном overhead на обслуживание.

Слайд 6: Масштабирование - Партиционирование (1.5 минуты)

Показать преимущества партиционирования

Одной из ключевых задач было обеспечение масштабируемости системы.
Для этого я применил hash-партиционирование таблицы snippets на 10 партиций.

Почему именно snippets? Потому что эта таблица содержит текстовое содержимое писем и полнотекстовые индексы -
именно она будет расти быстрее всего и испытывать наибольшую нагрузку при поиске.

Партиционирование по msg_id позволяет равномерно распределить данные и запросы между партициями.
В каждой партиции создается свой GIN-индекс для полнотекстового поиска, что позволяет выполнять поиск параллельно.

Результаты впечатляют: на текущих 104 тысячах сообщений индекс занимает всего 140 мегабайт, а поиск выполняется за 39 миллисекунд.
Мой прогноз показывает, что даже при росте до 10 миллионов сообщений система сохранит отличную производительность.

Слайд 7: Полнотекстовый поиск (1.5 минуты)

Полнотекстовый поиск - одна из ключевых возможностей моей системы.
Я реализовал его через generated column с типом tsvector, который автоматически строится из текстового содержимого писем.

Система поддерживает одновременно русский и английский языки, что критически важно для международных компаний.
Для ускорения поиска я создал GIN-индекс по этому полю.

Особенность моей реализации - использование функции ts_headline, которая не только находит relevant documents,
но и выделяет в них совпадения с запросом. Это значительно улучшает пользовательский опыт.

На практике поиск по сложным запросам, таким как «Зеленский AND leggings»,
выполняется за 367 миллисекунд даже при 2000 совпадениях.
При этом система поддерживает все возможности tsquery: булеву логику, веса, фразовый поиск.

Слайд 8: Бизнес-логика - PL/pgSQL (1.5 минуты)

Значительная часть бизнес-логики моей системы реализована на стороне БД через PL/pgSQL функции.
Это обеспечивает атомарность операций и снижает нагрузку на прикладной уровень.

Например, функция init_user_folders идемпотентно создает системные папки для нового пользователя, используя ON CONFLICT DO NOTHING для избежания дублей.

Функция move_message не просто перемещает письмо, но и проверяет права доступа - чтобы пользователь не мог переместить чужое письмо.

Особенно интересна функция send_draft, которая реализует всю логику отправки письма в рамках одной транзакции: валидация, создание копий для получателей, обновление MIME-заголовков, логирование.

Все эти операции обернуты в триггеры, которые гарантируют целостность данных и автоматическое выполнение необходимых действий.

Слайд 9: Аналитика - Рекурсивные запросы (1.5 минуты)

Для аналитики я реализовал сложную функцию get_tags_hierarchy_live, которая строит полную иерархию папок пользователя с расчетом размеров.

Эта функция использует рекурсивные CTE, состоящие из четырех этапов:
Сначала строится дерево папок, затем рассчитываются размеры писем в каждой папке, устанавливаются связи предок-потомок,
и наконец агрегируются размеры по всему поддереву.

Результат - я получаю полную картину использования дискового пространства: вижу не только размер
писем непосредственно в папке, но и суммарный размер всех вложенных подпапок.

При этом функция выполняется всего за 9 миллисекунд для конкретного пользователя,
что делает ее пригодной для использования в реальном времени.

Слайд 10: Аналитические представления (1 минута)

Для упрощения работы с данными я создал набор аналитических представлений.

Операционные вьюхи, такие как user_mailboxes, материализуют информацию о системных папках пользователя,
предоставляя единую точку доступа.

Аналитические вьюхи, like tag_total_size, предоставляют агрегированные данные для построения отчетов и дашбордов.

Эти представления не только упрощают написание запросов, но и могут быть легко материализованы для дальнейшего
повышения производительности при необходимости.

Слайд 11: Производительность и результаты (1 минута)

Давайте посмотрю на конкретные результаты производительности.
На текущем объеме данных в 104 тысячи сообщений GIN-индекс для полнотекстового поиска занимает 140 мегабайт,
что составляет примерно 1.3 килобайта на сообщение.

Поисковые запросы выполняются за десятки миллисекунд, даже при работе с тысячами совпадений.
Рекурсивные запросы для построения иерархии выполняются за 9 миллисекунд.

Мой анализ масштабируемости показывает, что система будет эффективно работать и при значительном росте объема
данных благодаря партиционированию и оптимизированной индексации.

Слайд 12: Итоги и выводы (1 минута)

В заключение хочу отметить, что все поставленные цели проекта были мной достигнуты. Я создал единое хранилище,
обеспечил быстрый поиск, реализовал глубокую аналитику и продемонстрировал ключевые приемы оптимизации PostgreSQL.

Система использует современные возможности PostgreSQL: например расширение pg_cron и партиционирование сниппетов.
Она готова к интеграции с реальной почтовой инфраструктурой и может служить основой для построения корпоративной
системы аналитики почтовой переписки.

Спасибо за внимание! Готов ответить на ваши вопросы.