

Архитектура и реализация аналитической витрины ML- транскрибации данных на базе ClickHouse и PostgreSQL

ClickHouse для
инженеров и
архитекторов БД



**Меня хорошо видно
& слышно?**



Защита проекта

Тема: Архитектура и реализация аналитической витрины ML-транскрибации данных на базе ClickHouse и PostgreSQL



Даниил Чушенко

Главный аналитик компании НПФ
«Будущее»

Стаж работы 4 года

План защиты

Цель и задачи проекта

Какие технологии использовались

Что получилось

Выводы

Вопросы и рекомендации

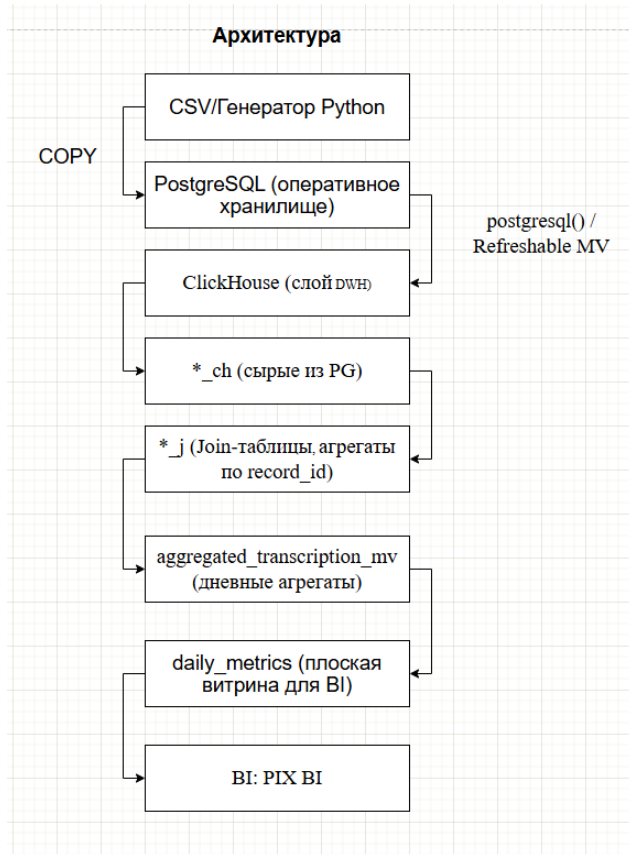
Цель и задачи проекта

Цель проекта: Создать архитектуру и хранилище данных для сбора и агрегации данных по ML-транскрибации с дальнейшим построением аналитических моделей

1. Собрать и хранить аудиометрики и результаты ML-транскрибации;
2. Построить витрину данных в ClickHouse;
3. Автоматизировать обновление данных из PostgreSQL;
4. Подготовить BI-дашборд для бизнес-аналитики.



Архитектура проекта



Блок-схема по ML-транскрибации



Какие технологии использовались

1. PostgreSQL — исходное хранилище данных;
2. ClickHouse — аналитическое хранилище;
3. Materialized Views (Refresh Every 1 Day) — ежедневный инкремент;
4. Join-таблицы (ENGINE = Join) — оптимизация агрегатов;
5. AggregatingMergeTree — основная витрина;
6. Pix BI — визуализация метрик;



Реализация пайплайна

Этап 1: Создание схемы в PostgreSQL

Этап 2: Репликация данных в ClickHouse через postgresql() и MVs

Этап 3: Создание join-таблиц для words, speakers, emotions

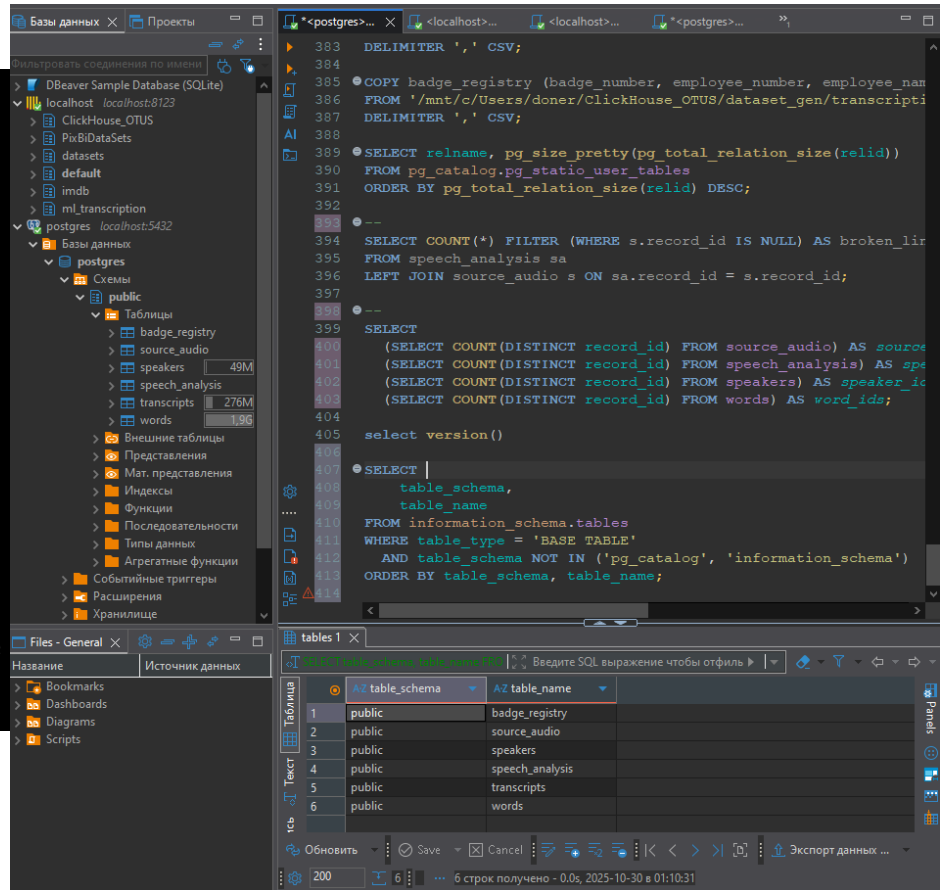
Этап 4: Формирование агрегатов → aggregated_transcription_mv

Этап 5: Финальная витрина → daily_metrics → Pix BI



Скрипт создания таблиц в PG

```
CREATE TABLE IF NOT EXISTS speakers (
id SERIAL PRIMARY KEY,
record_id UUID REFERENCES
source_audio(record_id) ON DELETE
CASCADE,
speaker TEXT,
speaker_type TEXT,
speech_speed NUMERIC(6,2),
speech_duration NUMERIC(10,2),
interruption_duration NUMERIC(10,2)
);
COPY speakers FROM
'/path/transcription_dataset_fixed/speakers.csv' CSV;
```



Сырые таблицы в СН

Модель данных

Дашборд

Модель данных приложения "Речевая аналитика (1)"

Новая связь

Добавить набор

Привязка данных

RLS

Сгенерировать календарь

words_ch ...

ml_transcription ...

speech_analysi... ...

speakers_ch ...

record_id	conflict_management	conversation_summary	created_at	updated_at	nda_compliance	repeat_prevention	conversation_end	survey_engagement	politeness	listening_skill	speakers_ch
cee602ef-fc39-4ccb-8000-0034a9dbdff3	76.44000244141	Клиент обратился по вопросу вопрос. Консультант ответил корректно.	29.09.2025 11:26:54	30.09.2025 11:26:54	0	0	82.83999633789	68.7799987793	84.12999725342	67.80999755859	92.121
72447075-ad43-4b3e-8000-823b909801d4	93.69000244141	Клиент обратился по вопросу продажа. Консультант ответил отлично.	28.09.2025 11:26:54	28.09.2025 11:26:54	0	0	71.19999694824	92.73999786377	75.05000305176	87.30000305176	85.61
6a2fb67f-3a04-476d-8001-79c946a3aed9	74.4700012207	Клиент обратился по вопросу жалоба. Консультант ответил отлично.	29.09.2025 11:26:54	29.09.2025 11:26:54	1	0	94.93000030518	64.98999786377	94.81999969482	81.5	89.44
4f50745f-df0f-		Клиент обратился по вопросу жалоба	23.09.2025	28.09.2025							

Название

ml_transcription

Описание

Источники

Datasets_ClickHouse

Соединения

words_ch

speech_analysis_ch

speakers_ch

100%

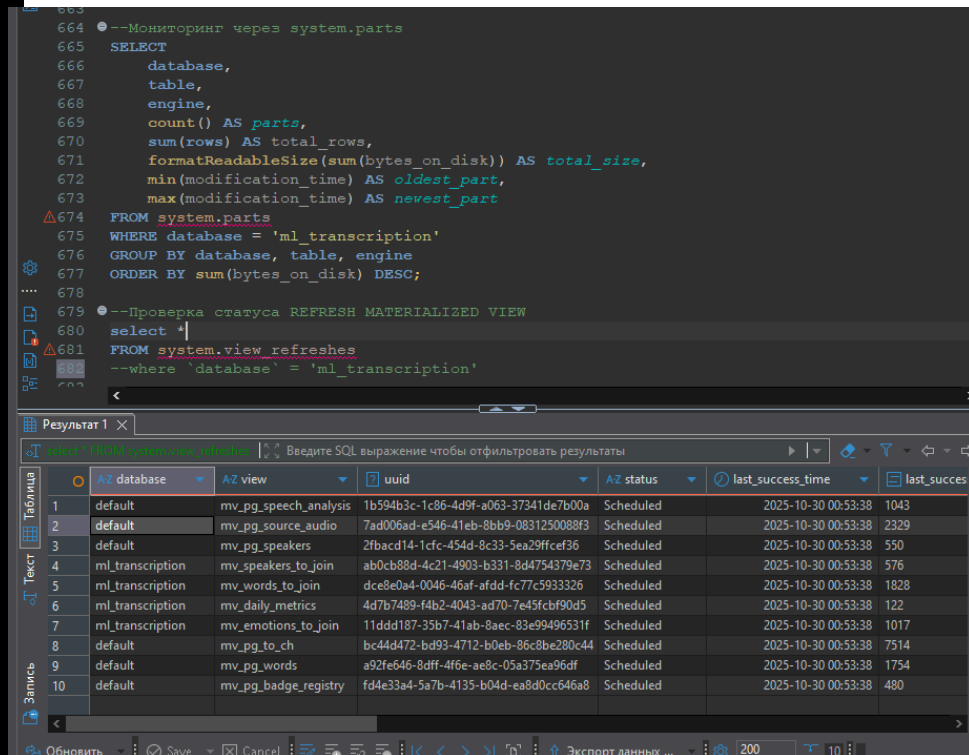
100%

100%



Обновление таблиц на СН через RMV

```
--speakers → speakers_ch
CREATE MATERIALIZED VIEW IF NOT EXISTS
mv_pg_speakers
REFRESH EVERY 1 DAY
APPEND
TO ml_transcription.speakers_ch AS
SELECT
id, record_id, speaker, speaker_type,
speech_speed, speech_duration,
interruption_duration
FROM postgresql(
'localhost:****', 'postgres', 'speakers',
'****', '*****')
WHERE id >
coalesce(
(SELECT max(id) FROM
ml_transcription.speakers_ch),
0
);
```



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains a SQL script with comments in Russian. The results pane shows a table with 10 rows of data.

```
663
664 --Мониторинг через system.parts
665 SELECT
666     database,
667     table,
668     engine,
669     count() AS parts,
670     sum(rows) AS total_rows,
671     formatReadableSize(sum(bytes_on_disk)) AS total_size,
672     min(modification_time) AS oldest_part,
673     max(modification_time) AS newest_part
674 FROM system.parts
675 WHERE database = 'ml_transcription'
676 GROUP BY database, table, engine
677 ORDER BY sum(bytes_on_disk) DESC;
678
679 --Проверка статуса REFRESH MATERIALIZED VIEW
680 select '
681 FROM system.view_refreshes
682 --where `database` = 'ml_transcription'
```

Результат 1

	AZ database	AZ view	uuid	AZ status	last_success_time	last_success
1	default	mv_pg_speech_analysis	1b594b3c-1c86-4d9f-a063-37341de7b00a	Scheduled	2025-10-30 00:53:38	1043
2	default	mv_pg_source_audio	7ad006ad-e546-41eb-8bb9-0831250088f3	Scheduled	2025-10-30 00:53:38	2329
3	default	mv_pg_speakers	2fbacd14-1cfc-454d-8c33-5ea29ffcef36	Scheduled	2025-10-30 00:53:38	550
4	ml_transcription	mv_speakers_to_join	ab0cb88d-4c21-4903-b331-8d4754379c73	Scheduled	2025-10-30 00:53:38	576
5	ml_transcription	mv_words_to_join	dce8e0a4-0046-46af-afdd-fc77c5933326	Scheduled	2025-10-30 00:53:38	1828
6	ml_transcription	mv_daily_metrics	4d7b7489-f4b2-4043-ad70-7e45fcbf90d5	Scheduled	2025-10-30 00:53:38	122
7	ml_transcription	mv_emotions_to_join	11ddd187-35b7-41ab-8aec-83e99496531f	Scheduled	2025-10-30 00:53:38	1017
8	default	mv_pg_to_ch	bc44d472-bd93-4712-b0eb-86c8be280c44	Scheduled	2025-10-30 00:53:38	7514
9	default	mv_pg_words	a92fe646-8dff-4f6e-ae8c-05a375ea96df	Scheduled	2025-10-30 00:53:38	1754
10	default	mv_pg_badge_registry	fd4e33a4-5a7b-4135-b04d-ea8d0cc646a8	Scheduled	2025-10-30 00:53:38	480

Подготовка таблиц с агрегатами

```
--Метрики по спикерам
CREATE TABLE ml_transcription.speakers_j
ENGINE = Join(ANY, LEFT, record_id)
AS
SELECT
record_id,
avg(speech_speed) AS avg_speech_speed,
avg(speech_duration) AS
avg_speech_duration,
avg(interruption_duration) AS
avg_interruption_duration
FROM ml_transcription.speakers_ch
GROUP BY record_id;
```

```
651 --Мониторинг всех таблиц и их объема
652 SELECT
653     database,
654     table,
655     engine,
656     total_rows AS rows,
657     formatReadableSize(total_bytes) AS size,
658     metadata_modification_time AS last_alter,
659     comment
660 FROM system.tables
661 WHERE database = 'ml_transcription'
662 and `engine` = 'Join'
663 ORDER BY total_bytes DESC;
664
665 --Мониторинг через system.parts
666 SELECT
667     database,
668     table,
```

Результат 1

Введите SQL выражение чтобы отфильтровать результаты

	AZ database	AZ table	AZ engine	rows	AZ size	last_alter
1	ml_transcription	emotions_j	Join	100000	11.82 MiB	2025-10-30 00:53:38
2	ml_transcription	speakers_j	Join	100000	10.29 MiB	2025-10-30 00:53:38
3	ml_transcription	words_j	Join	100000	10.29 MiB	2025-10-30 00:53:38

Фрагмент агрегирующей таблицы

```
CREATE MATERIALIZED VIEW
ml_transcription.aggregated_transcription_mv
ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(processed_at)
ORDER BY (channel_type, category, toDate(processed_at))
POPULATE AS
SELECT
-- Ключевые измерения
t.record_id,
t.channel_type,
t.category,
toDate(t.processed_at) AS date,
t.processed_at,
.....
FROM ml_transcription.transcripts_raw AS t
GLOBAL ANY LEFT JOIN ml_transcription.speakers_j AS sj USING
(record_id)
GLOBAL ANY LEFT JOIN ml_transcription.words_j AS wj USING
(record_id)
GLOBAL ANY LEFT JOIN ml_transcription.emotions_j AS ej USING
(record_id)
GROUP BY t.record_id, t.channel_type, t.category, t.processed_at;
```



Фрагмент RMV для таблица в BI

```
CREATE MATERIALIZED VIEW IF NOT EXISTS
ml_transcription.mv_daily_metrics
REFRESH EVERY 1 DAY
TO ml_transcription.daily_metrics
AS
SELECT
toDate(processed_at) AS date,
channel_type,
category,
avgMerge(avg_transcription_quality_state) AS avg_quality,
avgMerge(avg_overall_score_state) AS avg_score,
sumMerge(total_audio_duration_state) AS total_audio,
countMerge(calls_count_state) AS calls_count,
.....
FROM
ml_transcription.aggregated_transcription_mv
GROUP BY
date,
channel_type,
category;
```



Мониторинг

1) Активные процессы → `system.processes`

2) Место на диске → `system.disks`

3) Только завершённые запросы → `system.query_log`

`WHERE type = 'QueryFinish'`

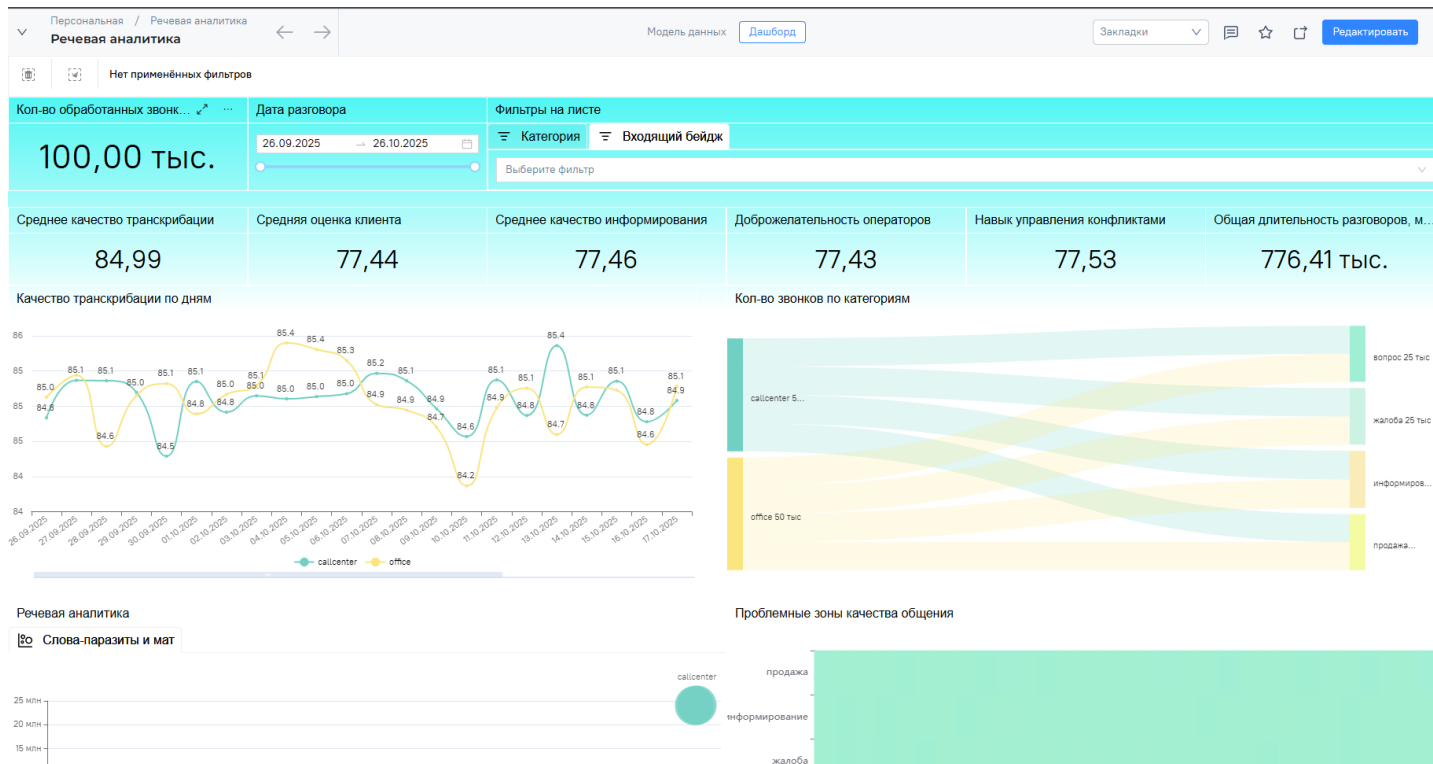
4) Ошибки в запросах → `system.query_log`

`WHERE exception != ''`

5) Мониторинг всех таблиц и их объёма → `system.tables`



Аналитический отчет в PIX BI



Модель данных [Дашборд](#)

[+ Новая связь](#)

Project_ML_CH ...

- Str channel_type
- 314 avg_conflict
- 123 total_words
- 123 filler_words
- 123 swear_words
- 123 positive_segme...
- 123 negative_segme...
- Str category
- date
- 314 avg_quality
- 314 avg_score
- 314 total_audio
- 314 avg_info_quality
- 314 avg_politeness
- 123 fragments_count
- 123 unique_speakers
- 123 neutral_segments
- 314 avg_volume_level
- 314 avg_speech_sp...
- 314 avg_speech_dur...
- 314 avg_interruptio...
- 314 avg_overlap_du...
- 314 avg_silence_dur...
- 314 avg_wait_time
- 314 avg_hold_time
- 314 avg_noise_level
- 123 calls_count

Выводы

- ✓ Построена полная архитектура DWH-уровня
- ✓ Реализована инкрементальная загрузка из PG
- ✓ Оптимизированы агрегаты через Join-таблицы
- ✓ Подготовлена BI-витрина и дашборд
- ✓ Создан тех-мониторинг ClickHouse

Вопросы и рекомендации



если есть вопросы



если вопросов нет



Спасибо за внимание!

OTUS

