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

Проект учебной курсовой OTUS «ClickHouse для инженеров и архитекторов БД».

Описание сквозного процесса ML-транскрибации

Два канала входа данных:

1. **Клиентская зона (бейджи)** — микрофоны/аудио-бейджи в офисах.
   → фиксируют диалог между клиентом и консультантом.
2. **Кол-центр** — телефонные звонки.
   → записываются через SIP-интеграцию, аудио хранится на сервере записи разговоров.

Проект реализует систему анализа клиентских коммуникаций, объединяющую ML-транскрибацию речи и ClickHouse-аналитику.

Система собирает аудиозаписи из клиентских зон и колл-центра, выполняет распознавание речи, анализ эмоций и тд.

Результаты транскрибации сохраняются в PostgreSQL и ежедневно инкрементально загружаются в ClickHouse через Refreshable Materialized Views.

На стороне ClickHouse формируются агрегаты и витрины данных, визуализируемые в Pix BI.

Цель: построить end-to-end конвейер от источника (PostgreSQL) до аналитической витрины в ClickHouse с автозагрузкой, агрегациями. готовой таблицей для BI и аналитическим отчетом PIX BI.

Содержание

Установка PIX BI, ClickHouse и PostgreSQL - https://docs.pixrobotics.com/articles/#!pix-bi-admin-1-31/install-linux

Состав и технологии

- **PostgreSQL**: исходные таблицы (оперативный слой)
- **ClickHouse**: хранение, инкрементальная загрузка из PG, агрегации, витрины
- **Python**: генератор синтетического датасета (CSV)
- **BI**: визуализация KPI (Pix BI)

Датасет и генерация CSV

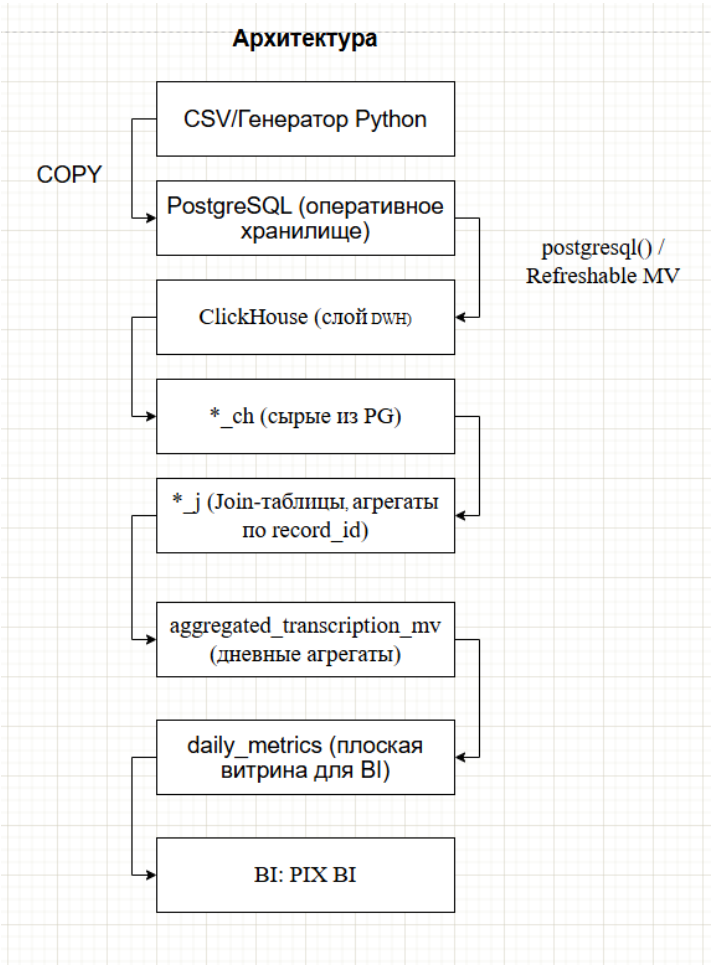
Скрипт генерирует ~ CSV для всех таблиц (целевой вес ≈ 4–5 ГБ).

cd ClickHouse_OTUS/dataset_gen

python3 generate_transcription_dataset_fixed.py

Выход: ./transcription_dataset_fixed/{source_audio.csv, speech_analysis.csv, speakers.csv, words.csv, badge_registry.csv}

Архитектура



Блок-схема по ML-транскрибации

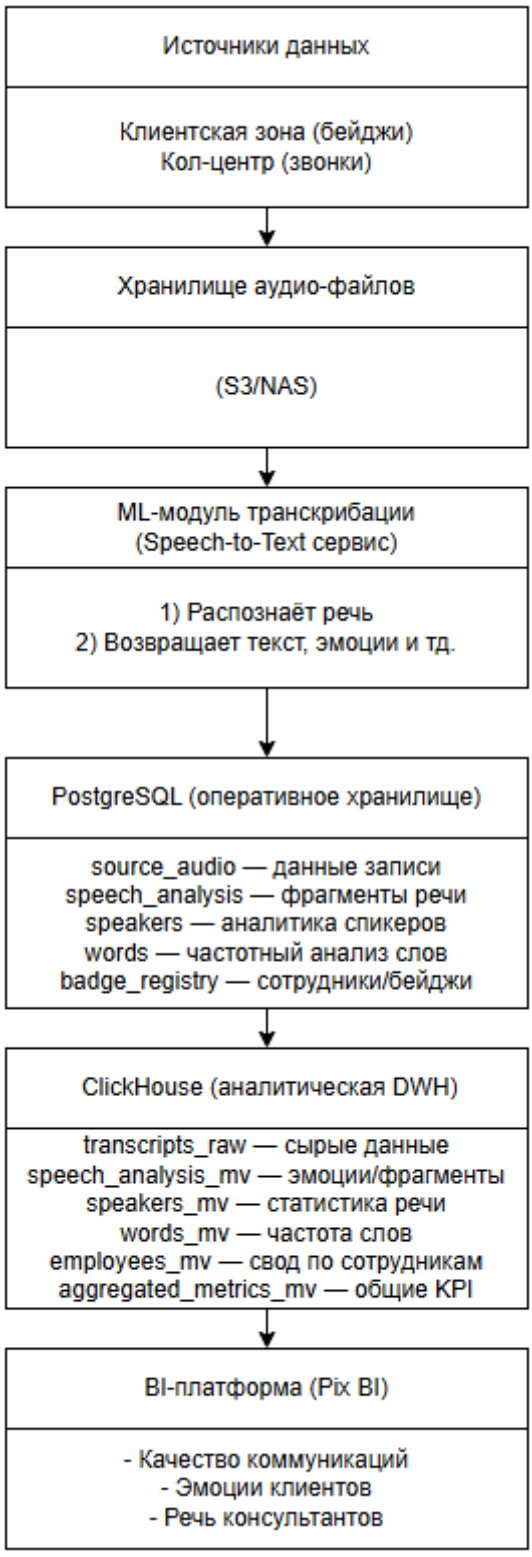


Схема PostgreSQL (DDL) + загрузка CSV

DDL (создание схемы)

```
CREATE TABLE IF NOT EXISTS source_audio (
    record_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    audio_name TEXT NOT NULL,
    category TEXT,
    channel_type TEXT CHECK (channel_type IN ('office','callcenter')),
    overlap_duration NUMERIC(10,2),
    silence_duration NUMERIC(10,2),
    transcription_quality NUMERIC(5,2),
    audio_duration NUMERIC(10,2),
    processed_at TIMESTAMP DEFAULT NOW(),
    overall_score NUMERIC(5,2),
    wait_time NUMERIC(10,2),
    client_end_time NUMERIC(10,2),
    hold_time NUMERIC(10,2),
    interruption BOOLEAN DEFAULT FALSE,
    background_noise NUMERIC(5,2),
    volume_level NUMERIC(5,2),
    init_quality NUMERIC(5,2),
    client_identification NUMERIC(5,2),
    request_understanding NUMERIC(5,2),
    info_quality NUMERIC(5,2),
    product_presentation NUMERIC(5,2),
    objection_handling NUMERIC(5,2),
    nda_compliance BOOLEAN DEFAULT TRUE,
    repeat_prevention BOOLEAN DEFAULT TRUE,
    conversation_end NUMERIC(5,2),
    survey_engagement NUMERIC(5,2),
```

```
    politeness NUMERIC(5,2),
    listening_skill NUMERIC(5,2),
    speech_clarity NUMERIC(5,2),
    conflict_management NUMERIC(5,2),
    conversation_summary TEXT,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
CREATE INDEX idx_source_audio_processed_at ON
source_audio(processed_at);
CREATE INDEX idx_source_audio_channel_type ON
source_audio(channel_type);

CREATE TABLE IF NOT EXISTS speech_analysis (
    id SERIAL PRIMARY KEY,
    record_id UUID REFERENCES source_audio(record_id) ON DELETE
CASCADE,
    start_time NUMERIC(10,2),
    end_time NUMERIC(10,2),
    text TEXT,
    emotion TEXT,
    speaker TEXT,
    speaker_type TEXT
);
CREATE INDEX idx_speech_analysis_record ON speech_analysis(record_id);
CREATE INDEX idx_speech_analysis_speaker ON speech_analysis(speaker);

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)
);
CREATE INDEX idx_speakers_record ON speakers(record_id);

CREATE TABLE IF NOT EXISTS words (
    id SERIAL PRIMARY KEY,
    record_id UUID REFERENCES source_audio(record_id) ON DELETE
CASCADE,
    word TEXT,
    frequency INT,
    speaker TEXT,
    is_filler BOOLEAN DEFAULT FALSE,
    is_swear BOOLEAN DEFAULT FALSE
);
CREATE INDEX idx_words_record ON words(record_id);
CREATE INDEX idx_words_word ON words(word);

CREATE TABLE IF NOT EXISTS badge_registry (
    id SERIAL PRIMARY KEY,
    badge_number TEXT NOT NULL,
    employee_number TEXT,
    employee_name TEXT,
    attached_at TIMESTAMP,
```

```
    detached_at TIMESTAMP
);
CREATE INDEX idx_badge_registry_number ON
badge_registry(badge_number);
```

Загрузка CSV в PostgreSQL

```
COPY source_audio FROM '/path/transcription_dataset_fixed/source_audio.csv'
CSV;
COPY speech_analysis FROM
'/path/transcription_dataset_fixed/speech_analysis.csv’ CSV;
COPY speakers FROM '/path/transcription_dataset_fixed/speakers.csv' CSV;
COPY words FROM '/path/transcription_dataset_fixed/words.csv' CSV;
COPY badge_registry FROM
'/path/transcription_dataset_fixed/badge_registry.csv'   CSV;
```

ClickHouse: слой хранения и загрузки из PG

Подключение к PostgreSQL

```
CREATE NAMED COLLECTION pg_conn AS
host='localhost', port=5432, database='postgres', user='postgres', password='***';


--Проверка
SELECT count()
FROM postgresql('localhost:5432', 'postgres', 'source_audio', 'postgres', '***');
```

База и сырые таблицы (под структуру PG)

```
CREATE DATABASE IF NOT EXISTS ml_transcription;


CREATE TABLE IF NOT EXISTS ml_transcription.transcripts_raw
(
    record_id UUID,
    audio_name String,
    category String,
```

channel_type LowCardinality(String),
overlap_duration Float32,
silence_duration Float32,
transcription_quality Float32,
audio_duration Float32,
processed_at DateTime,
overall_score Float32,
wait_time Float32,
client_end_time Float32,
hold_time Float32,
interruption UInt8,
background_noise Float32,
volume_level Float32,
init_quality Float32,
client_identification Float32,
request_understanding Float32,
info_quality Float32,
product_presentation Float32,
objection_handling Float32,
nda_compliance UInt8,
repeat_prevention UInt8,
conversation_end Float32,
survey_engagement Float32,
politeness Float32,
listening_skill Float32,
speech_clarity Float32,
conflict_management Float32,
conversation_summary String,
created_at DateTime,

```
    updated_at DateTime
)
ENGINE = ReplacingMergeTree(updated_at)
PARTITION BY toYYYYMM(processed_at)
ORDER BY (record_id, processed_at);

CREATE TABLE IF NOT EXISTS ml_transcription.speech_analysis_ch
(
    id UInt32, record_id UUID, start_time Float32, end_time Float32,
    text String, emotion LowCardinality(String),
    speaker String, speaker_type LowCardinality(String)
)
ENGINE = MergeTree ORDER BY (record_id, start_time);

CREATE TABLE IF NOT EXISTS ml_transcription.speakers_ch
(
    id UInt32, record_id UUID, speaker String, speaker_type
LowCardinality(String),
    speech_speed Float32, speech_duration Float32, interruption_duration Float32
)
ENGINE = MergeTree ORDER BY (record_id, speaker);

CREATE TABLE IF NOT EXISTS ml_transcription.words_ch
(
    id UInt32, record_id UUID, word LowCardinality(String),
    frequency UInt32, speaker String, is_filler UInt8, is_swear UInt8
)
ENGINE = MergeTree ORDER BY (record_id, word);
```

```
CREATE TABLE IF NOT EXISTS ml_transcription.badge_registry_ch
(
    id UInt32, badge_number String, employee_number String,
    employee_name String, attached_at DateTime, detached_at Nullable(DateTime)
)
ENGINE = MergeTree ORDER BY (badge_number);
```

Refreshable Materialized Views: инкремент из PG

```
-- source_audio → transcripts_raw
DROP MATERIALIZED VIEW IF EXISTS mv_pg_source_audio;
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_pg_source_audio
REFRESH EVERY 1 DAY
APPEND
TO ml_transcription.transcripts_raw AS
SELECT
    record_id, audio_name, category, channel_type,
    overlap_duration, silence_duration, transcription_quality, audio_duration,
    toDateTime(processed_at) AS processed_at, overall_score,
    wait_time, client_end_time, hold_time, CAST(interruption AS UInt8) AS interruption,
    background_noise, volume_level, init_quality, client_identification,
    request_understanding, info_quality, product_presentation, objection_handling,
    CAST(nda_compliance AS UInt8) AS nda_compliance,
    CAST(repeat_prevention AS UInt8) AS repeat_prevention,
    conversation_end, survey_engagement, politeness, listening_skill,
    speech_clarity, conflict_management, conversation_summary,
    toDateTime(created_at) AS created_at, toDateTime(updated_at) AS updated_at
FROM postgresql('localhost:5432','postgres','source_audio','postgres','***')
WHERE updated_at > coalesce(
    (SELECT max(updated_at) FROM ml_transcription.transcripts_raw),
```

```
    toDateTime('1970-01-01 00:00:00')
);

-- аналогично для остальных
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_pg_speech_analysis
REFRESH EVERY 1 DAY APPEND TO ml_transcription.speech_analysis_ch AS
SELECT id, record_id, start_time, end_time, text, emotion, speaker, speaker_type
FROM postgresql('localhost:5432','postgres','speech_analysis','postgres','***')
WHERE id > coalesce((SELECT max(id) FROM
ml_transcription.speech_analysis_ch),0);

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:5432','postgres','speakers','postgres','***')
WHERE id > coalesce((SELECT max(id) FROM
ml_transcription.speakers_ch),0);

CREATE MATERIALIZED VIEW IF NOT EXISTS mv_pg_words
REFRESH EVERY 1 DAY APPEND TO ml_transcription.words_ch AS
SELECT id, record_id, word, frequency, speaker, CAST(is_filler AS UInt8),
CAST(is_swear AS UInt8)
FROM postgresql('localhost:5432','postgres','words','postgres','***')
WHERE id > coalesce((SELECT max(id) FROM ml_transcription.words_ch),0);

CREATE MATERIALIZED VIEW IF NOT EXISTS mv_pg_badge_registry
REFRESH EVERY 1 DAY APPEND TO ml_transcription.badge_registry_ch AS
SELECT id, badge_number, employee_number, employee_name,
    toDateTime(attached_at) AS attached_at,
```

```
    CAST(NULLIF(detached_at,'') AS Nullable(DateTime)) AS detached_at
FROM postgresql('localhost:5432','postgres','badge_registry','postgres','***')
WHERE id > coalesce((SELECT max(id) FROM
ml_transcription.badge_registry_ch),0);
```

Join-таблицы для быстрых агрегаций

```
DROP TABLE IF EXISTS ml_transcription.words_j;
CREATE TABLE ml_transcription.words_j
ENGINE = Join(ANY, LEFT, record_id)
AS SELECT
   record_id,
   toUInt64(sum(frequency))                    AS total_words,
   toUInt64(sumIf(frequency, is_filler = 1))       AS filler_words,
   toUInt64(sumIf(frequency, is_swear  = 1))       AS swear_words
FROM ml_transcription.words_ch GROUP BY record_id;

DROP TABLE IF EXISTS ml_transcription.emotions_j;
CREATE TABLE ml_transcription.emotions_j
ENGINE = Join(ANY, LEFT, record_id)
AS SELECT
   record_id,
   toUInt64(count())                           AS fragments_count,
   toUInt64(uniqExact(speaker))                AS unique_speakers,
   toUInt64(countIf(emotion='positive'))       AS positive_segments,
   toUInt64(countIf(emotion='negative'))       AS negative_segments,
   toUInt64(countIf(emotion='neutral'))        AS neutral_segments
FROM ml_transcription.speech_analysis_ch GROUP BY record_id;

DROP TABLE IF EXISTS ml_transcription.speakers_j;
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;
```

MV для обновления Join-таблиц

```
DROP MATERIALIZED VIEW IF EXISTS ml_transcription.mv_words_to_join;
CREATE MATERIALIZED VIEW ml_transcription.mv_words_to_join
REFRESH EVERY 1 DAY TO ml_transcription.words_j AS
SELECT record_id,
      toUInt64(sum(frequency))              AS total_words,
      toUInt64(sumIf(frequency, is_filler=1)) AS filler_words,
      toUInt64(sumIf(frequency, is_swear=1))  AS swear_words
FROM ml_transcription.words_ch GROUP BY record_id;

DROP MATERIALIZED VIEW IF EXISTS
ml_transcription.mv_emotions_to_join;
CREATE MATERIALIZED VIEW ml_transcription.mv_emotions_to_join
REFRESH EVERY 1 DAY TO ml_transcription.emotions_j AS
SELECT record_id,
      toUInt64(count())                   AS fragments_count,
      toUInt64(uniqExact(speaker))            AS unique_speakers,
      toUInt64(countIf(emotion='positive'))    AS positive_segments,
      toUInt64(countIf(emotion='negative'))    AS negative_segments,
      toUInt64(countIf(emotion='neutral'))     AS neutral_segments
FROM ml_transcription.speech_analysis_ch GROUP BY record_id;
```

```
DROP MATERIALIZED VIEW IF EXISTS
ml_transcription.mv_speakers_to_join;

CREATE MATERIALIZED VIEW ml_transcription.mv_speakers_to_join

REFRESH EVERY 1 DAY TO ml_transcription.speakers_j 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;
```

Агрегированная витрина

Храним уже дневные агрегаты (гранулярность: date, channel_type, category). Это резко снижает объем и ускоряет BI.

```
DROP MATERIALIZED VIEW IF EXISTS
ml_transcription.aggregated_transcription_mv;


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

  t.record_id, t.channel_type, t.category,

  toDate(t.processed_at) AS date, t.processed_at,


  avgState(t.transcription_quality)  AS avg_transcription_quality_state,

  avgState(t.overall_score)          AS avg_overall_score_state,

  sumState(t.audio_duration)         AS total_audio_duration_state,

  countState(t.record_id)            AS calls_count_state,
```

```
    avgState(t.info_quality)          AS avg_info_quality_state,
    avgState(t.politeness)            AS avg_politeness_state,
    avgState(t.conflict_management)   AS avg_conflict_state,

    avgState(t.overlap_duration)      AS avg_overlap_duration_state,
    avgState(t.silence_duration)      AS avg_silence_duration_state,
    avgState(t.wait_time)             AS avg_wait_time_state,
    avgState(t.hold_time)             AS avg_hold_time_state,
    avgState(t.background_noise)      AS avg_noise_level_state,
    avgState(t.volume_level)          AS avg_volume_level_state,

    avgState(sj.avg_speech_speed)     AS avg_speech_speed_state,
    avgState(sj.avg_speech_duration)  AS avg_speech_duration_state,
    avgState(sj.avg_interruption_duration) AS avg_interruption_duration_state,

    sumState(wj.total_words)          AS total_words_state,
    sumState(wj.filler_words)         AS filler_words_state,
    sumState(wj.swear_words)          AS swear_words_state,

    sumState(ej.fragments_count)      AS fragments_count_state,
    sumState(ej.unique_speakers)      AS unique_speakers_state,
    sumState(ej.positive_segments)    AS positive_fragments_state,
    sumState(ej.negative_segments)    AS negative_fragments_state,
    sumState(ej.neutral_segments)     AS neutral_fragments_state

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;
```

BI-таблица с готовыми KPI

```
DROP TABLE IF EXISTS ml_transcription.daily_metrics;
CREATE TABLE ml_transcription.daily_metrics
(
    date Date,
    channel_type LowCardinality(String),
    category String,
    avg_quality Float32,
    avg_score Float32,
    total_audio Float32,
    calls_count UInt64,
    avg_info_quality Float32,
    avg_politeness Float32,
    avg_conflict Float32,
    avg_overlap_duration Float32,
    avg_silence_duration Float32,
    avg_wait_time Float32,
    avg_hold_time Float32,
    avg_noise_level Float32,
    avg_volume_level Float32,
    avg_speech_speed Float32,
    avg_speech_duration Float32,
    avg_interruption_duration Float32,
    total_words UInt64,
```

```
    filler_words UInt64,
    swear_words UInt64,
    fragments_count UInt64,
    unique_speakers UInt64,
    positive_segments UInt64,
    negative_segments UInt64,
    neutral_segments UInt64
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(date)
ORDER BY (channel_type, category, date);

DROP MATERIALIZED VIEW IF EXISTS ml_transcription.mv_daily_metrics;
CREATE MATERIALIZED VIEW ml_transcription.mv_daily_metrics
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,
    avgMerge(avg_info_quality_state)           AS avg_info_quality,
    avgMerge(avg_politeness_state)             AS avg_politeness,
    avgMerge(avg_conflict_state)               AS avg_conflict,
    avgMerge(avg_overlap_duration_state)       AS avg_overlap_duration,
    avgMerge(avg_silence_duration_state)       AS avg_silence_duration,
    avgMerge(avg_wait_time_state)              AS avg_wait_time,
    avgMerge(avg_hold_time_state)              AS avg_hold_time,
```

```
    avgMerge(avg_noise_level_state)           AS avg_noise_level,
    avgMerge(avg_volume_level_state)          AS avg_volume_level,
    avgMerge(avg_speech_speed_state)          AS avg_speech_speed,
    avgMerge(avg_speech_duration_state)       AS avg_speech_duration,
    avgMerge(avg_interruption_duration_state) AS avg_interruption_duration,
    sumMerge(total_words_state)               AS total_words,
    sumMerge(filler_words_state)              AS filler_words,
    sumMerge(swear_words_state)               AS swear_words,
    sumMerge(fragments_count_state)           AS fragments_count,
    sumMerge(unique_speakers_state)           AS unique_speakers,
    sumMerge(positive_fragments_state)        AS positive_segments,
    sumMerge(negative_fragments_state)        AS negative_segments,
    sumMerge(neutral_fragments_state)         AS neutral_segments
FROM ml_transcription.aggregated_transcription_mv
GROUP BY date, channel_type, category;
```

Порядок первичного запуска

```
-- 1) Подтянуть данные из PG
REFRESH MATERIALIZED VIEW mv_pg_source_audio;
REFRESH MATERIALIZED VIEW mv_pg_speech_analysis;
REFRESH MATERIALIZED VIEW mv_pg_speakers;
REFRESH MATERIALIZED VIEW mv_pg_words;
REFRESH MATERIALIZED VIEW mv_pg_badge_registry;

-- 2) Построить join-таблицы (через их MV)
REFRESH MATERIALIZED VIEW ml_transcription.mv_words_to_join;
REFRESH MATERIALIZED VIEW ml_transcription.mv_emotions_to_join;
REFRESH MATERIALIZED VIEW ml_transcription.mv_speakers_to_join;
```

-- 3) aggregated_transcription_mv заполняется (POPULATE) и дальше будет обновляться

-- 4) daily_metrics наполняется автоматически через mv_daily_metrics

Проверки:

SELECT count(), min(processed_at), max(processed_at) FROM ml_transcription.transcripts_raw;

SELECT count() FROM ml_transcription.aggregated_transcription_mv;

SELECT count() FROM ml_transcription.daily_metrics;

Мониторинг ETL и хранения

Мониторинг включает 4 уровня контроля: размер таблиц, партиции, статус MV, свободное место на дисках:

- Размеры таблиц:

SELECT database, table, engine, total_rows AS rows, formatReadableSize(total_bytes) AS size,

metadata_modification_time AS last_alter, comment

FROM system.tables

WHERE database='ml_transcription'

ORDER BY total_bytes DESC;

- По партициям:

SELECT database, table, engine, count() AS parts, sum(rows) AS total_rows,

formatReadableSize(sum(bytes_on_disk)) AS total_size,

min(modification_time) AS oldest_part, max(modification_time) AS newest_part

FROM system.parts

WHERE database='ml_transcription'

GROUP BY database, table, engine

ORDER BY sum(bytes_on_disk) DESC;

- Статусы Refresh-MV:

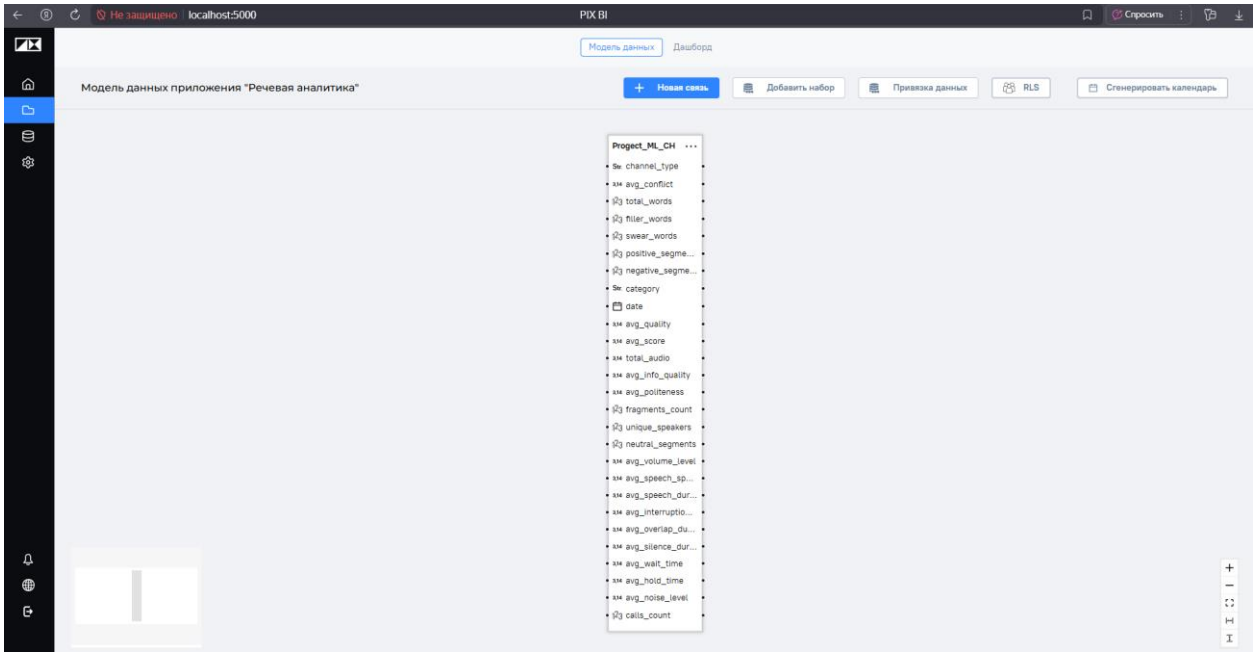
```
SELECT view_name, last_successful_refresh, last_refresh_duration_ms,
    last_refresh_status, next_scheduled_refresh, last_refresh_error
FROM system.view_refreshes
WHERE database='ml_transcription'
ORDER BY last_successful_refresh DESC;
```

- Диски:

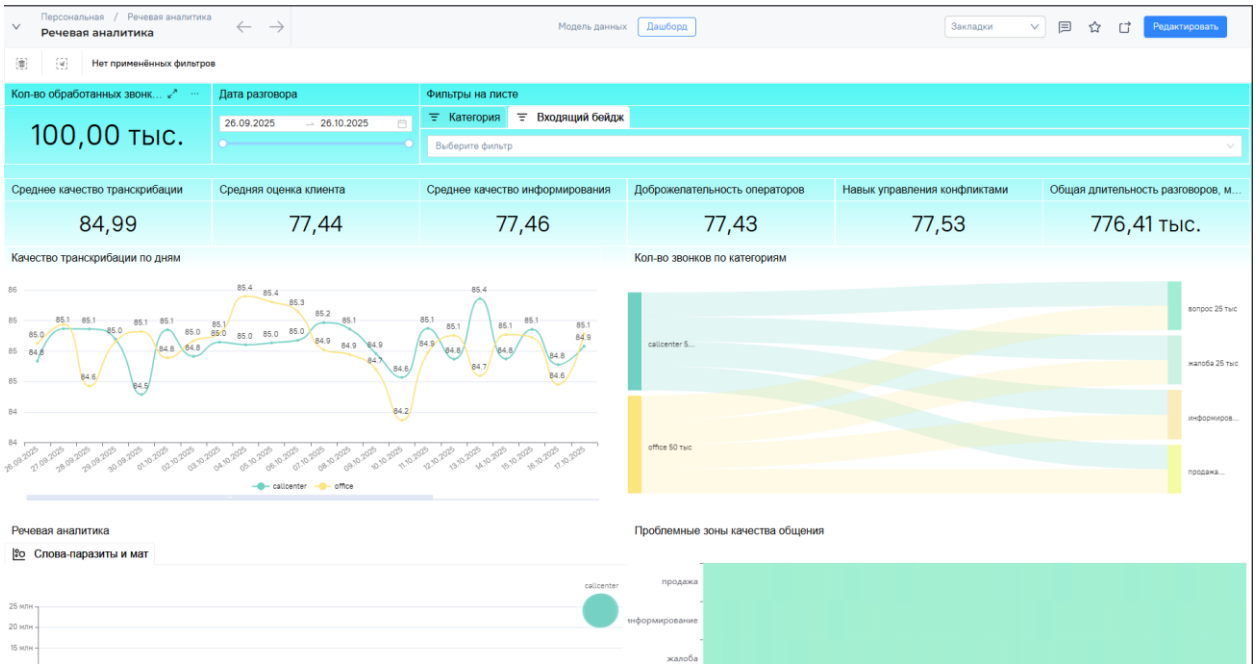
```
SELECT name, formatReadableSize(free_space) AS free_space,
    formatReadableSize(total_space) AS total_space,
    round(free_space/total_space*100,2) AS free_percent
FROM system.disks;
```

Дашборд:

Модель данных:



Дашборд:


Персональная / Речевая аналитика
Речевая аналитика
Модель данных
Дашборд
Закладки
Редактировать
Нет применённых фильтров
Кол-во обработанных звонк...
100,00 тыс.
Дата разговора
26.09.2025
26.10.2025
Фильтры на листе
Категория
Входящий бейдж
Выберите фильтр
Среднее качество транскрибации
84,99
Средняя оценка клиента
77,44
Среднее качество информирования
77,46
Доброжелательность операторов
77,43
Навык управления конфликтами
77,53
Общая длительность разговоров, м...
776,41 тыс.
Качество транскрибации по дням
callcenter
office
Кол-во звонков по категориям
callcenter 5...
office 50 тыс
вопрос 25 тыс
жалоба 25 тыс
информиров...
продажа...
Речевая аналитика
Слова-паразиты и мат
25 млн
20 млн
15 млн
callcenter
Проблемные зоны качества общения
продажа
информирование
жалоба

Сравнение по таблицам

Таблица	PostgreSQL	ClickHouse	Сжатие	Пример строк
words / words_ch	1.917 GB	**179 MB**	**~10.7× меньше**	~17.5 млн
speech_analysis / speech_analysis_ch	583 MB	**50.6 MB**	**~11.5× меньше**	~2.5 млн
source_audio / transcripts_raw	45 MB	**10.7 MB**	**~4.2× меньше**	100 000
speakers / speakers_ch	49 MB	**8.4 MB**	**~5.8× меньше**	~400 000
badge_registry / badge_registry_ch	1.36 MB	**0.21 MB**	**~6.4× меньше**	10 000

CHANGELOG

Версия PIX BI: 1.31.13

CH: 25.6.2.5

PG: PostgreSQL 17.5