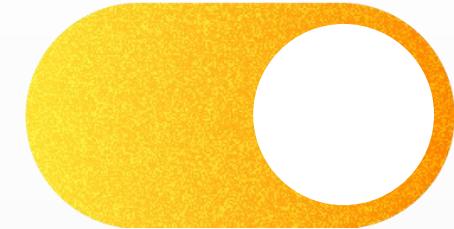


Т

PostgreSQL Partitioning

Управление данными и производительность с .NET и EF Core

Илья Мжачев



Ведущий разработчик

Email

ilya@mzhachev.ru

Telegram

[@mzhach](https://t.me/mzhach)



О проекте



О проекте



Обрабатываем **> 10 млн.** звонков в день



О проекте

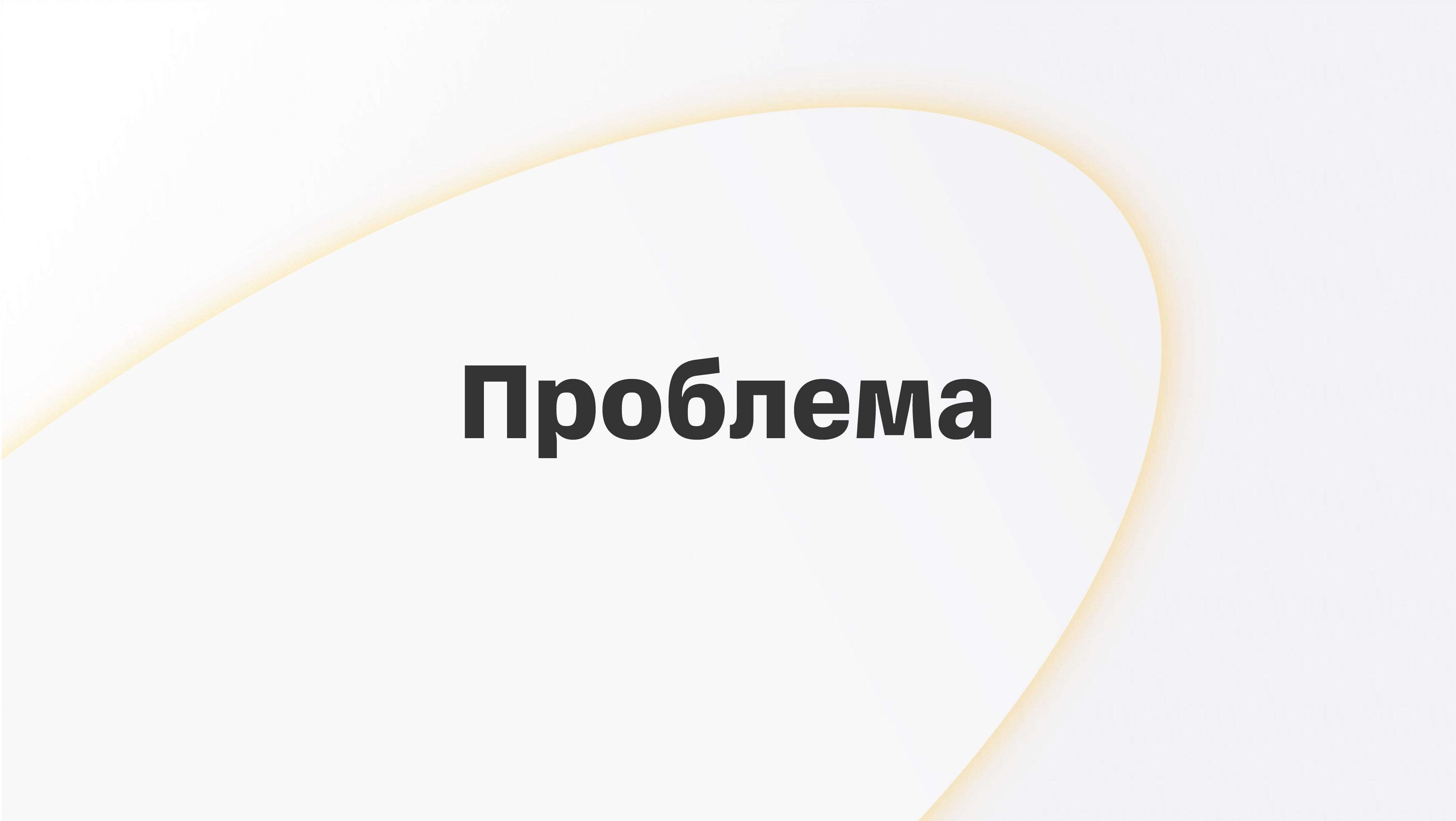
- ✓ Обрабатываем **> 10 млн.** звонков в день
- ✓ Между звонками в среднем **5-10** секунд



О проекте

- ✓ Обрабатываем **> 10 млн.** звонков в день
- ✓ Между звонками в среднем **5-10** секунд
- ✓ **> 50%** звонков обслуживается роботами



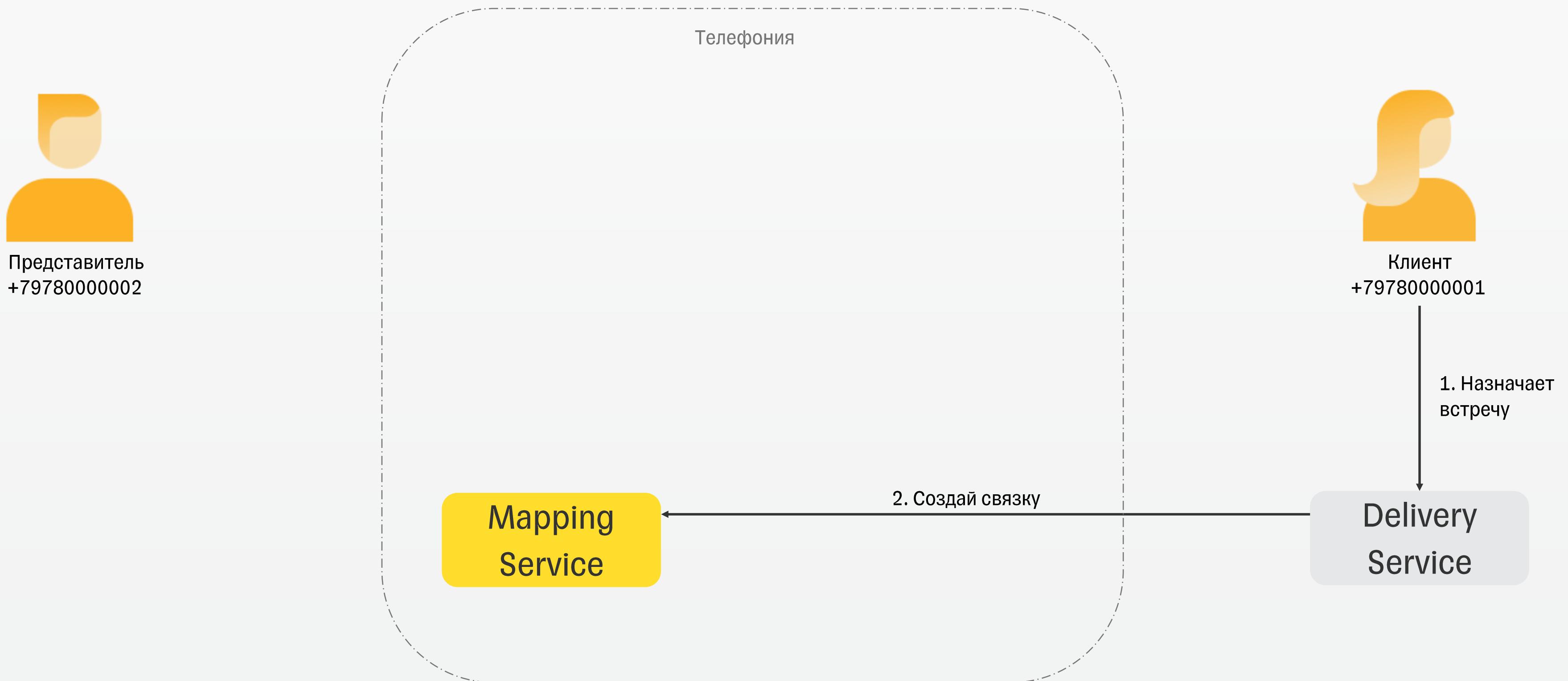


проблема

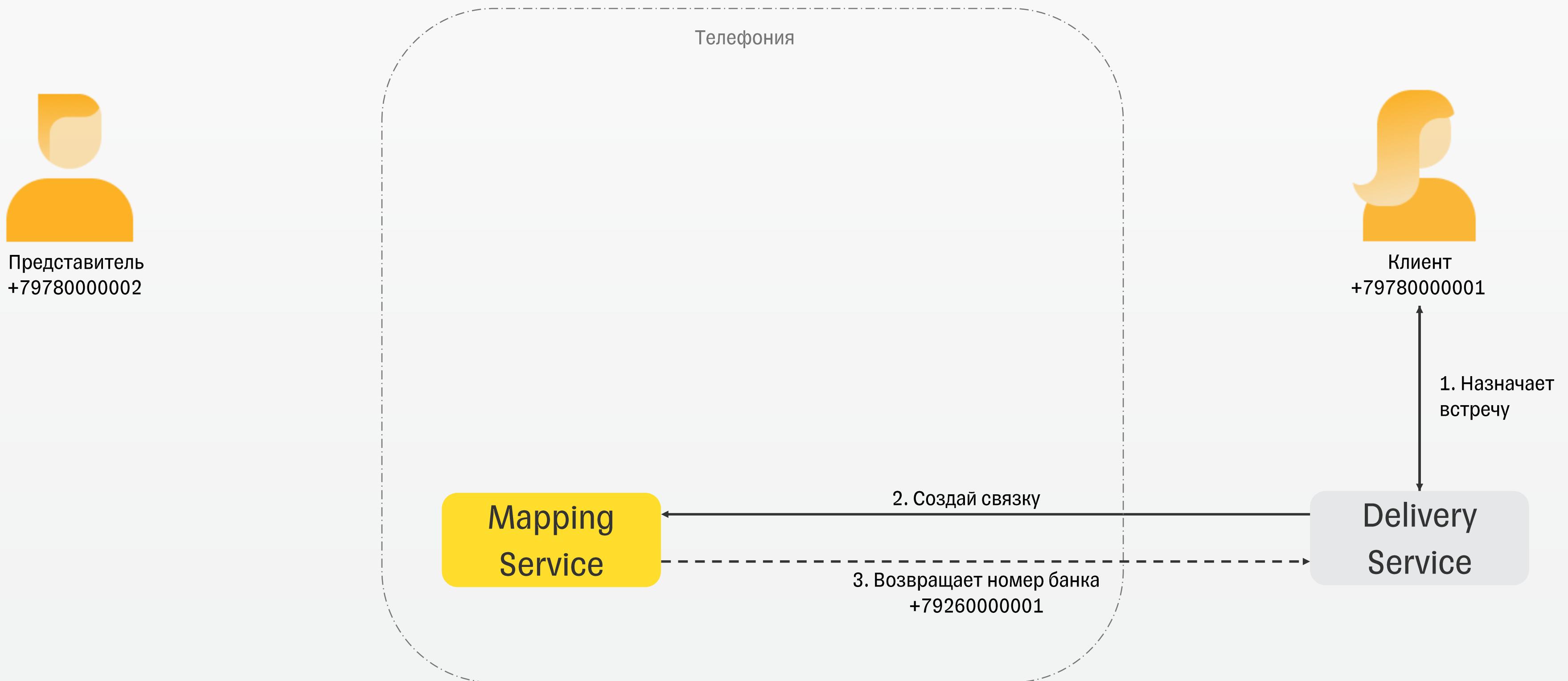
Создание связи



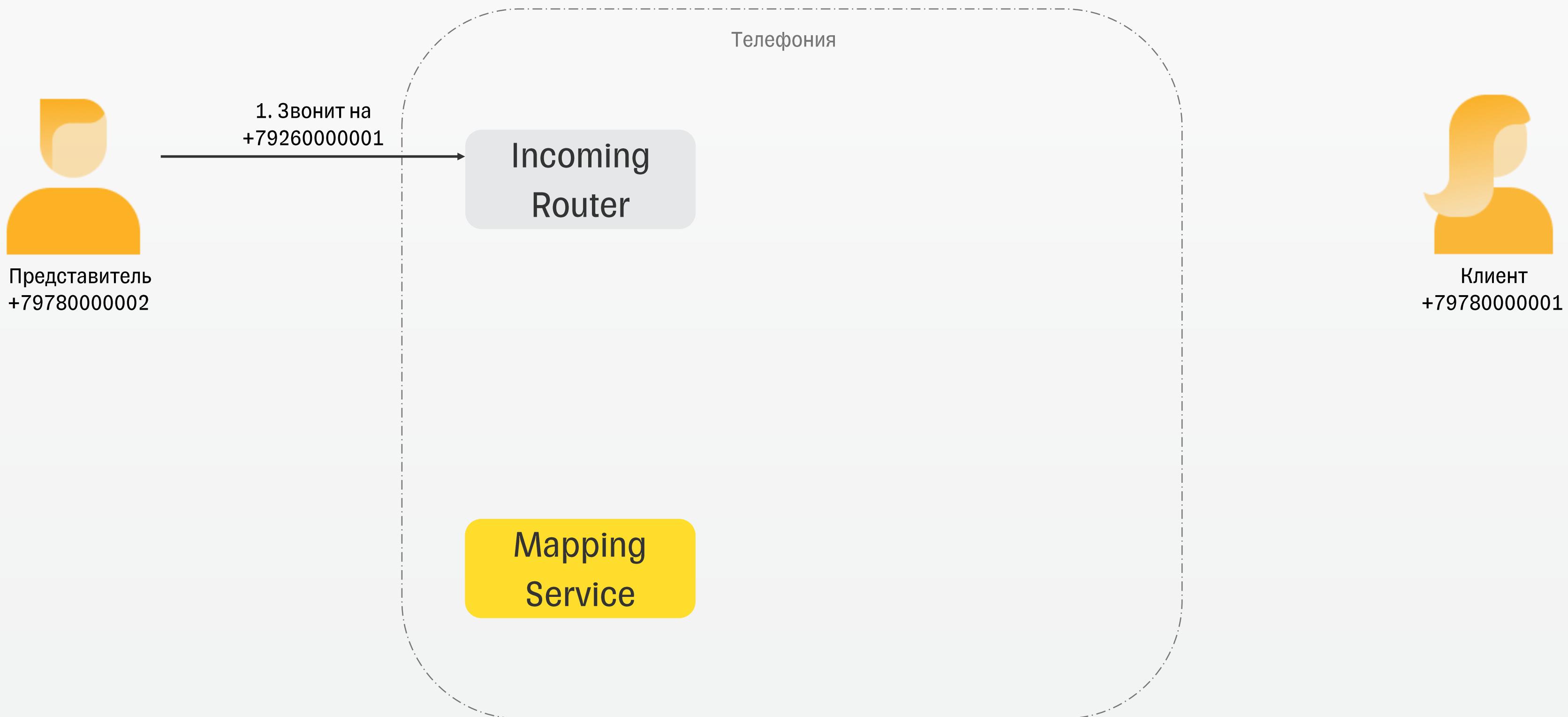
Создание связи



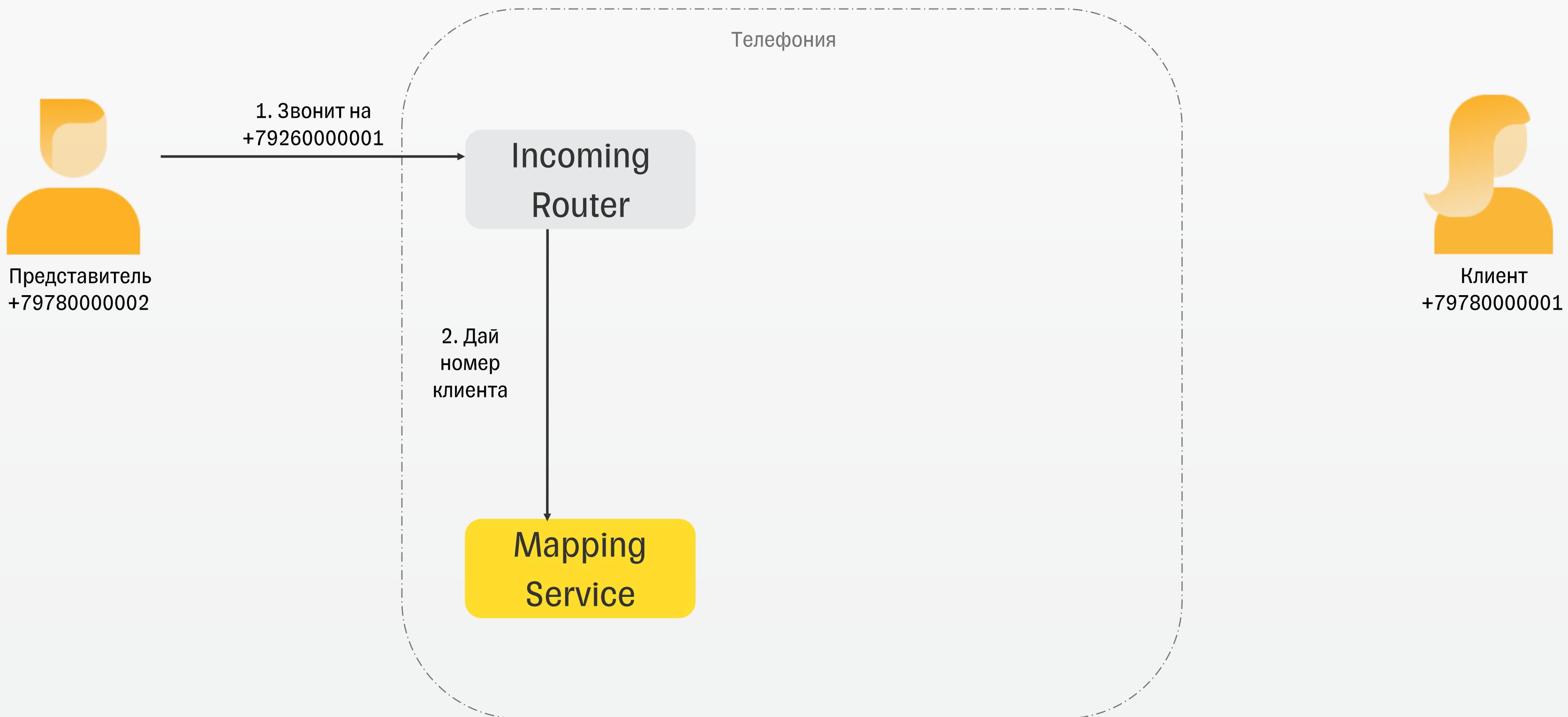
Создание связи



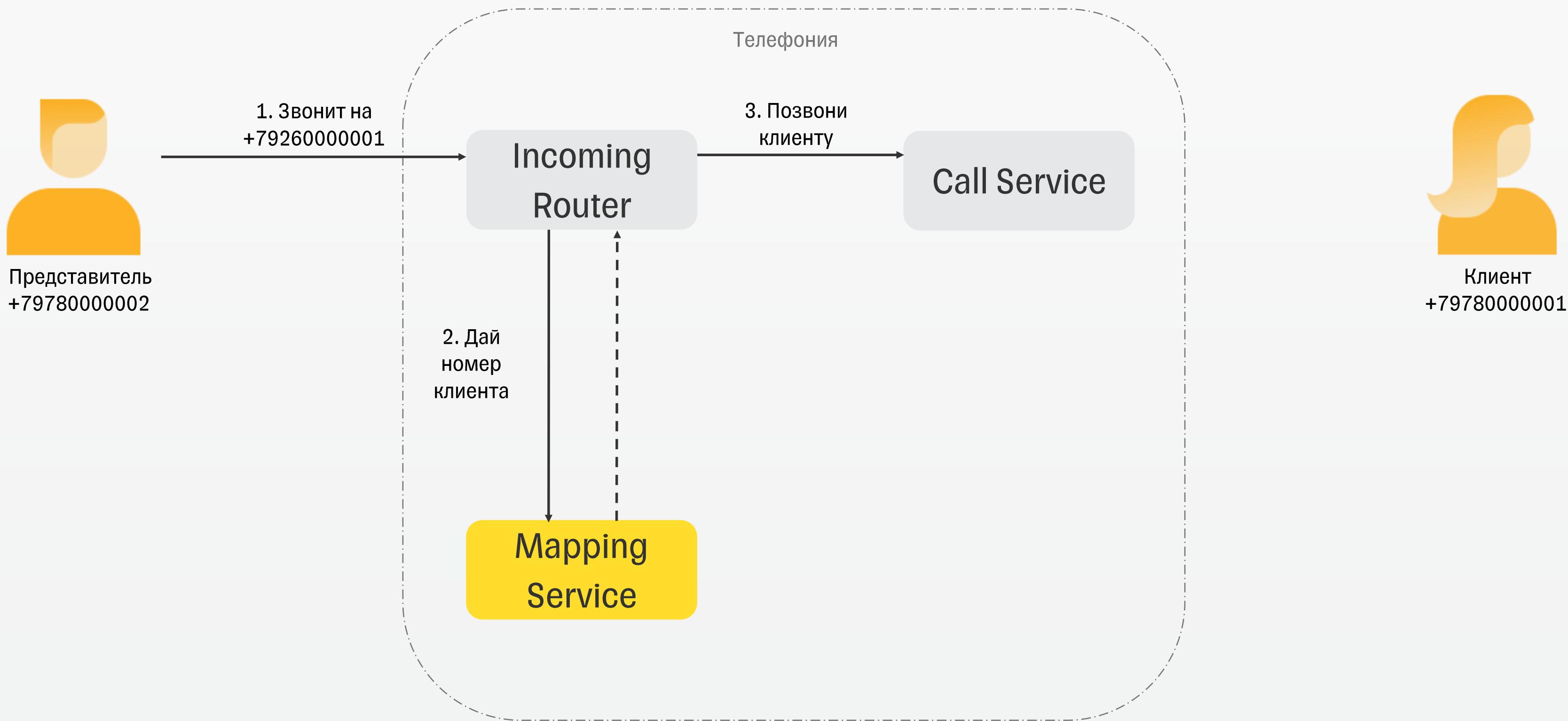
Сокрытие номера клиента



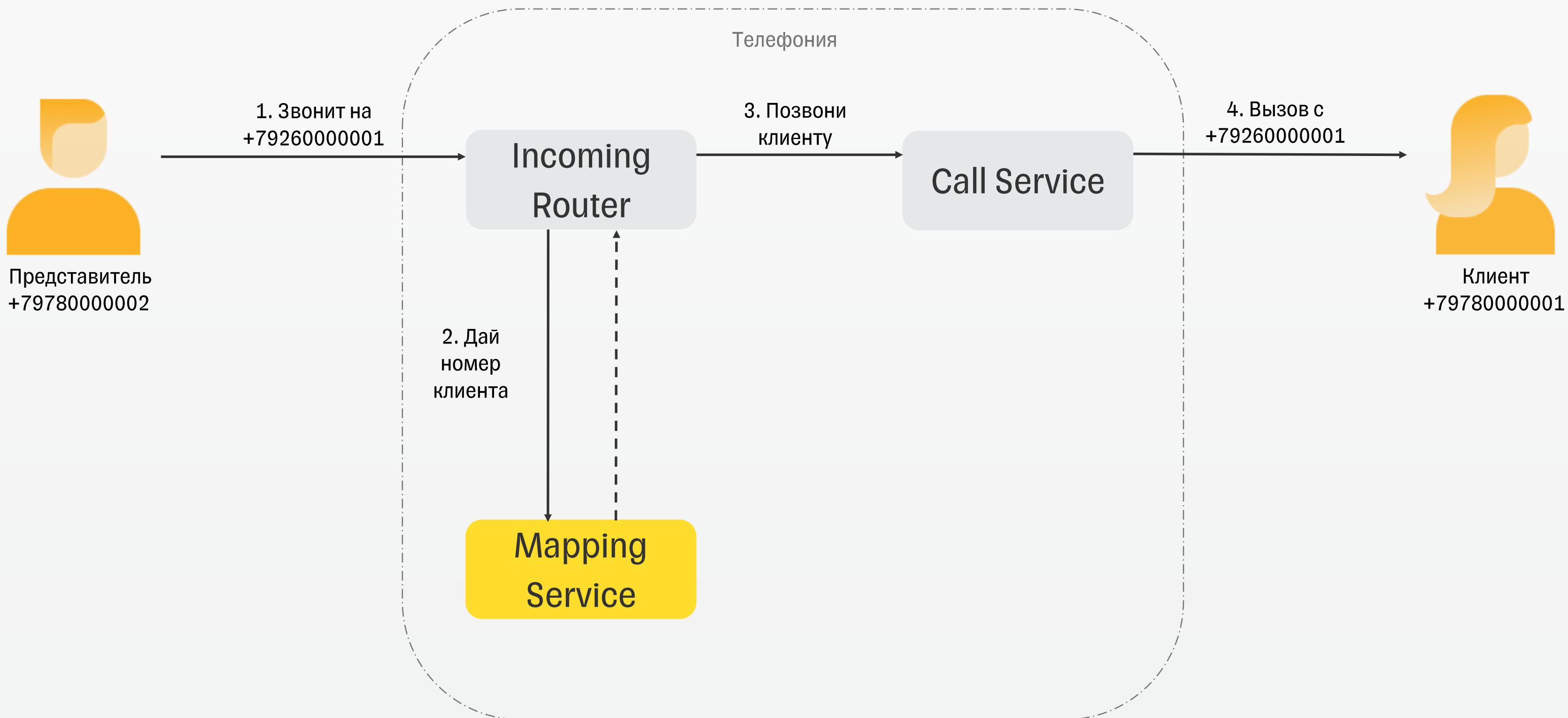
Сокрытие номера клиента



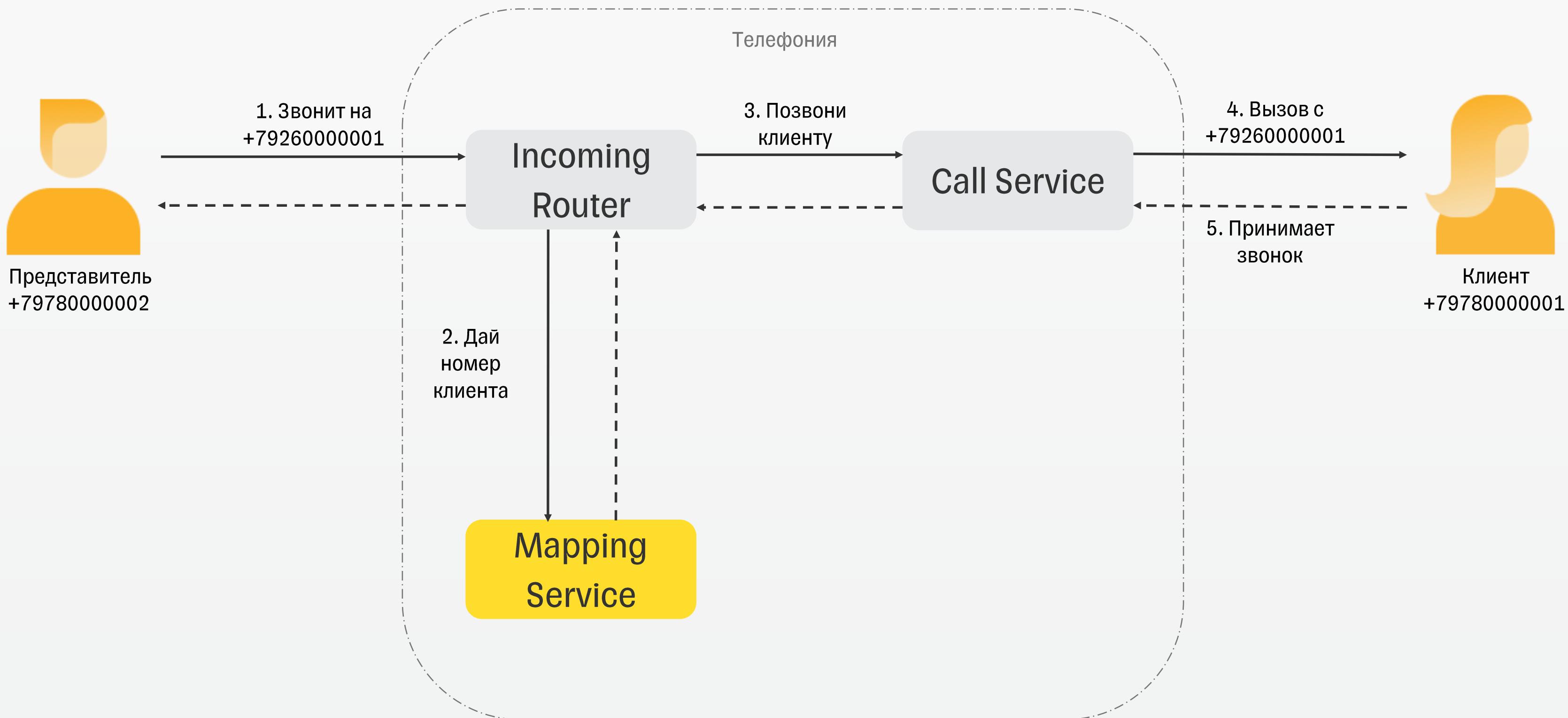
Сокрытие номера клиента

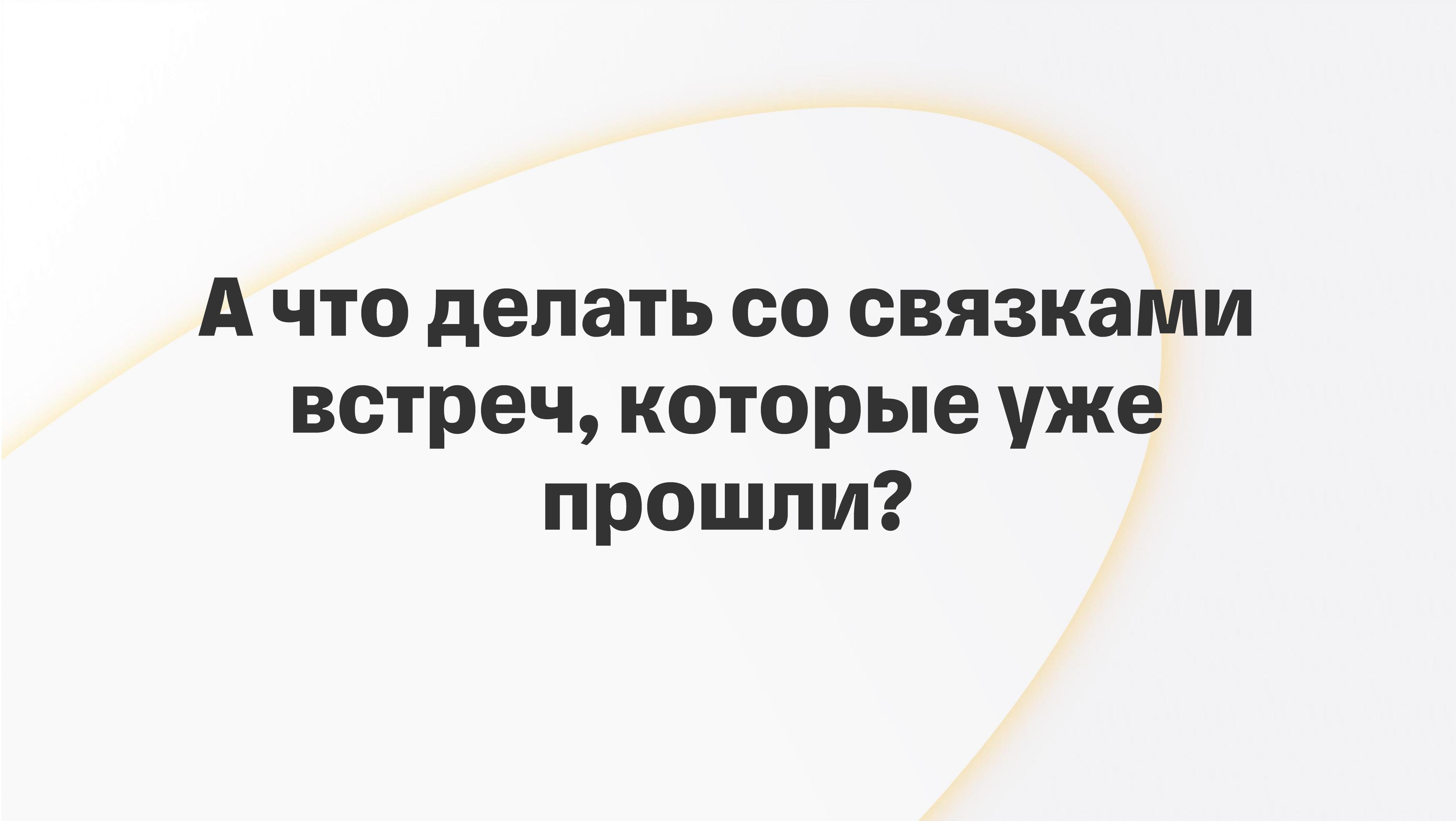


Сокрытие номера клиента



Сокрытие номера клиента





**А что делать со связками
встреч, которые уже
прошли?**

Способы удаления данных из PostgreSQL

01

Через “DELETE”

Минусы удаления через DELETE



Дополнительная нагрузка

Запрос на большой таблице создает дополнительную нагрузку.



Данные очищаются не сразу

Освобождение памяти произойдет только после VACUUM FULL.

Способы удаления данных из PostgreSQL

01

Через “DELETE”

02

Через “TRUNCATE”

Способы удаления данных из PostgreSQL

01

Через “DELETE”

02

Через “TRUNCATE”

03

Удалять партициями

Таблица связок

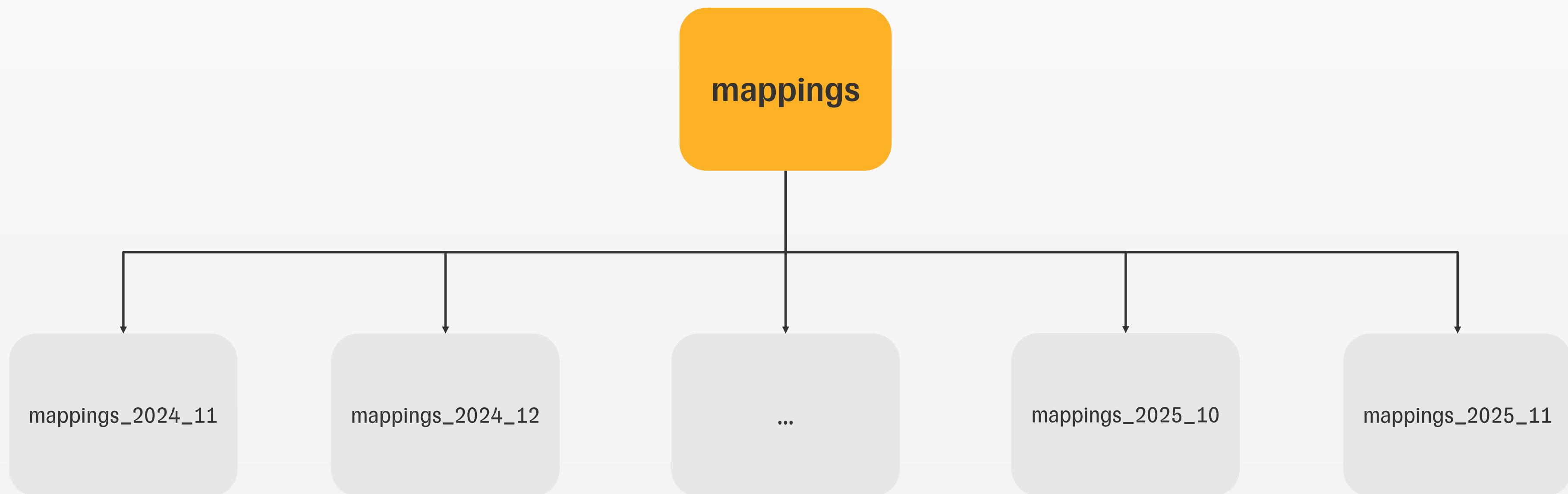
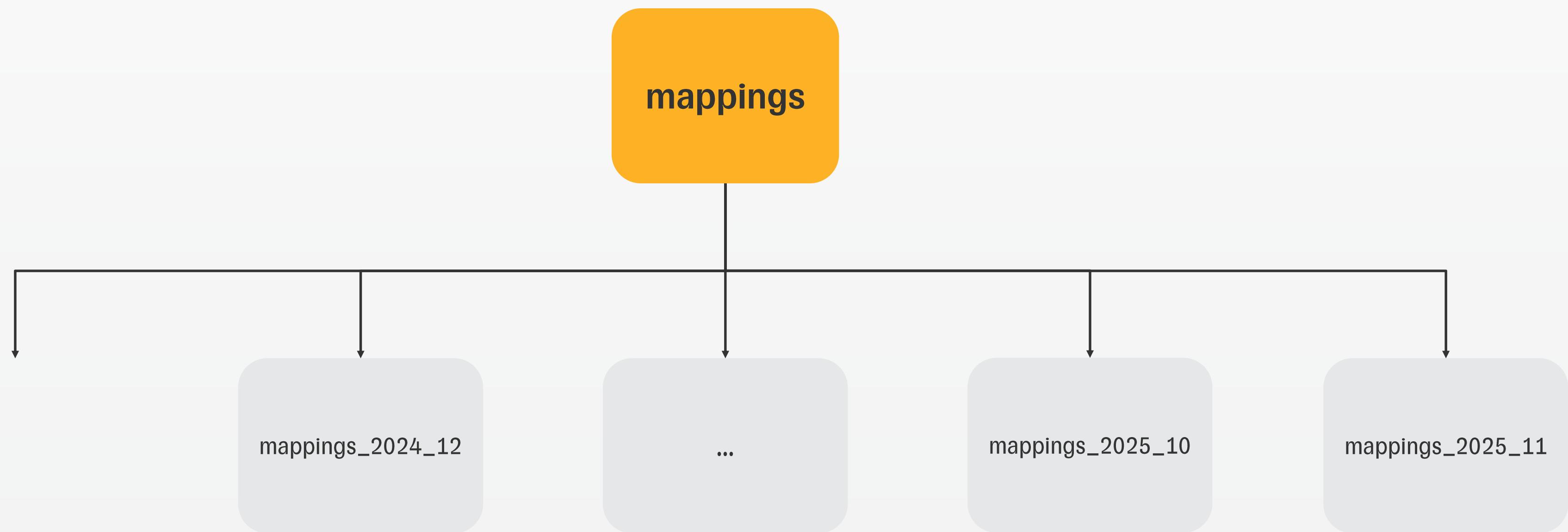


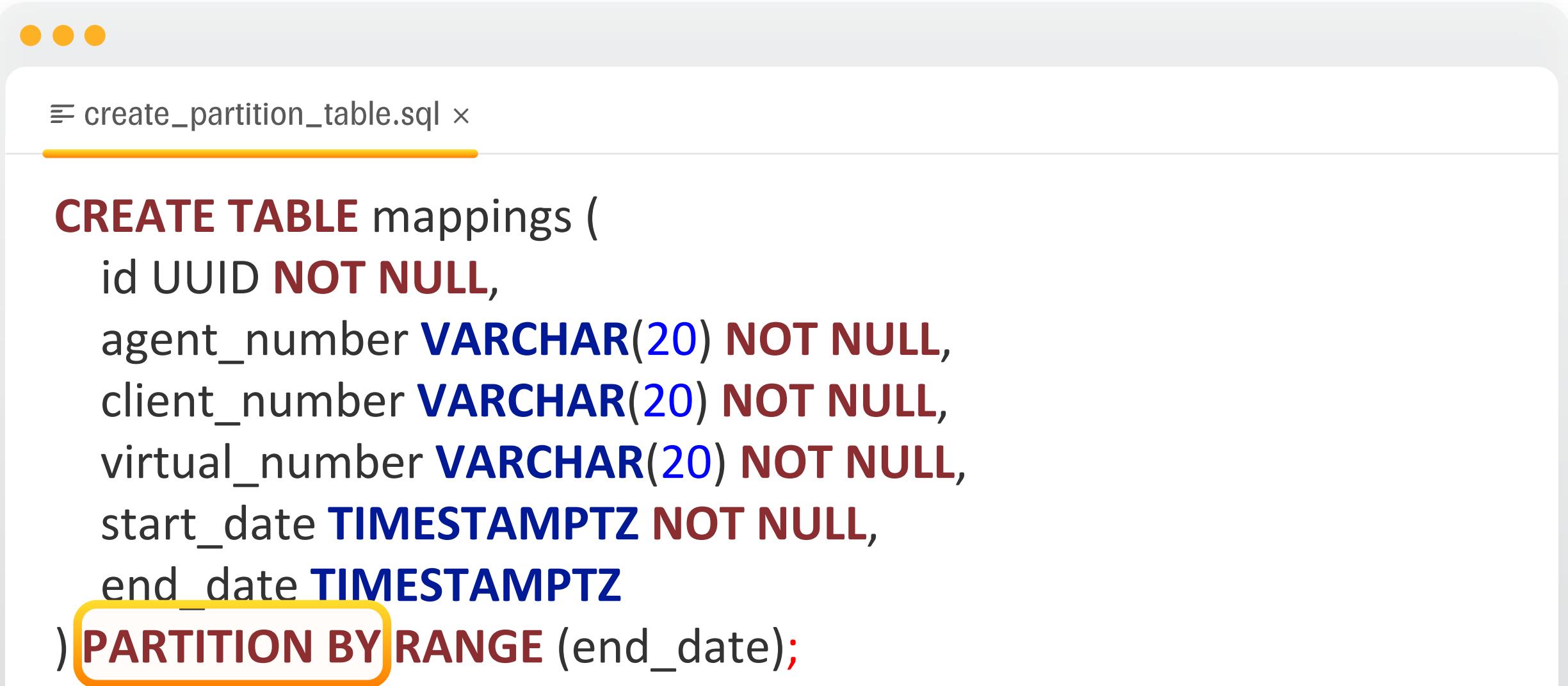
Таблица связок



декларативное партиционирование

Создание партицирован ной таблицы

- Нужно использовать конструкцию
PARTITION BY;



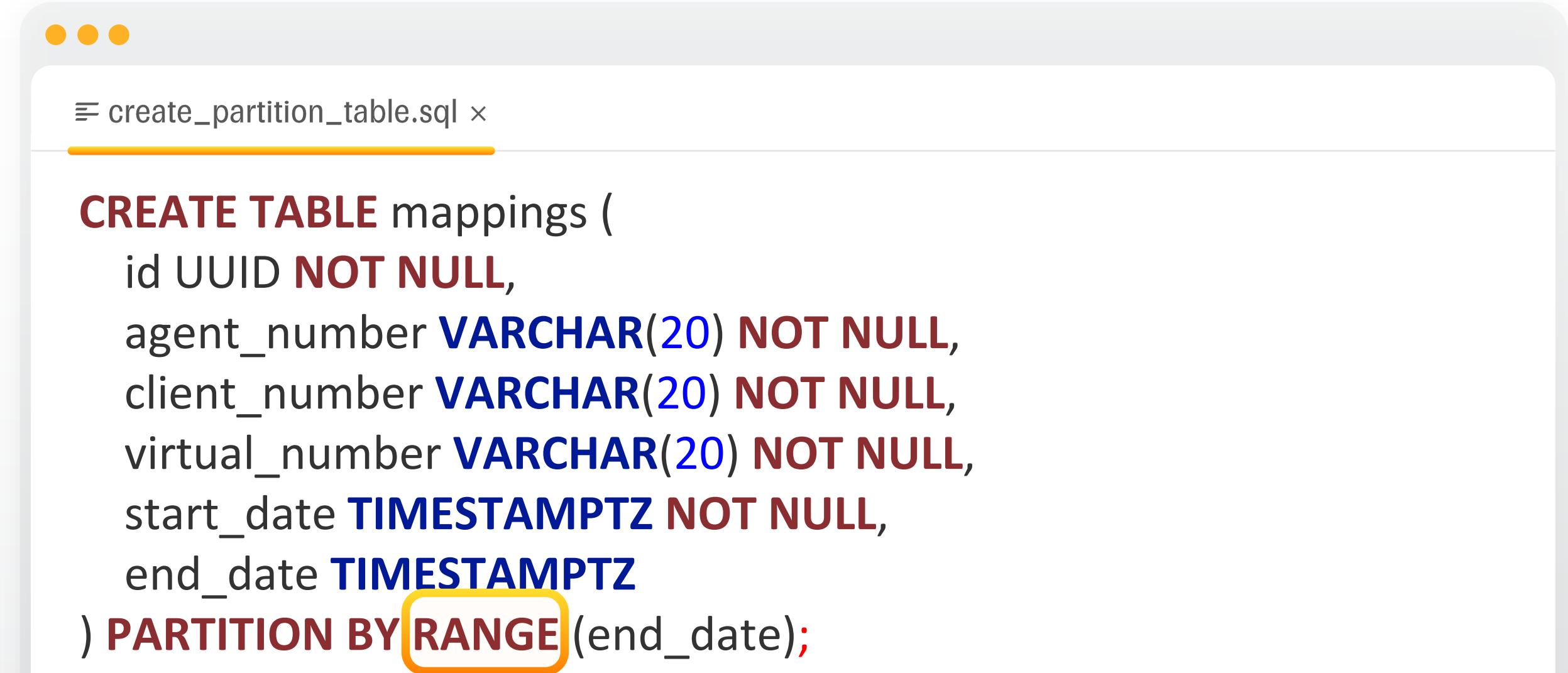
The screenshot shows a code editor window titled "create_partition_table.sql". The code is a SQL `CREATE TABLE` statement:

```
CREATE TABLE mappings (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ
) PARTITION BY RANGE (end_date);
```

The word `PARTITION BY` is highlighted with a yellow rectangular border.

Создание партицирован ной таблицы

- Нужно использовать конструкцию **PARTITION BY**;
- Указать форму partitionирования.
Существует три формы: **RANGE**, **LIST**, **HASH**;



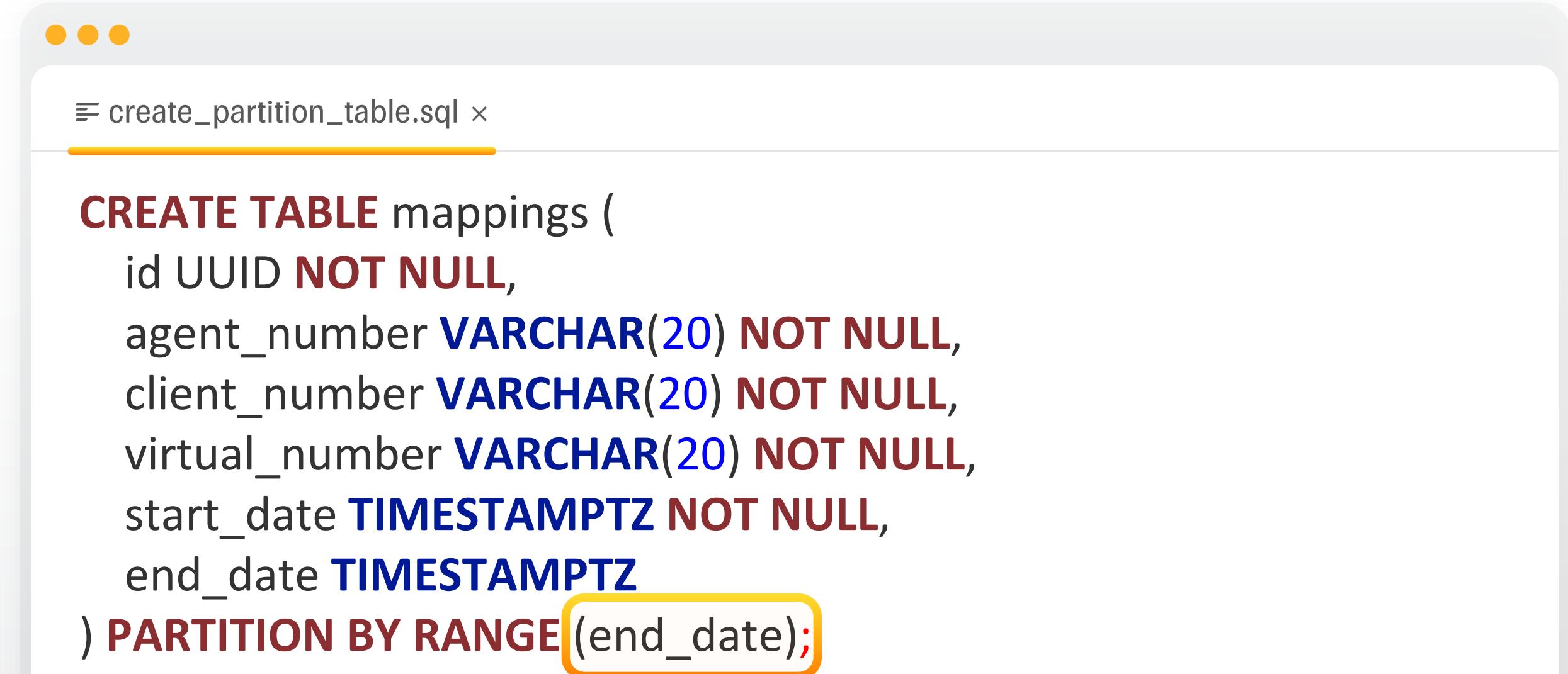
The screenshot shows a code editor window titled "create_partition_table.sql". The code defines a table "mappings" with the following columns and partitioning:

```
CREATE TABLE mappings (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ
) PARTITION BY RANGE(end_date);
```

The word "RANGE" is highlighted with a yellow box.

Создание партицирован ной таблицы

- Нужно использовать конструкцию **PARTITION BY**;
- Указать форму партиционирования. Существует три формы: **RANGE**, **LIST**, **HASH**;
- Указать ключ партиционирования – одну из колонок либо несколько или выражение.



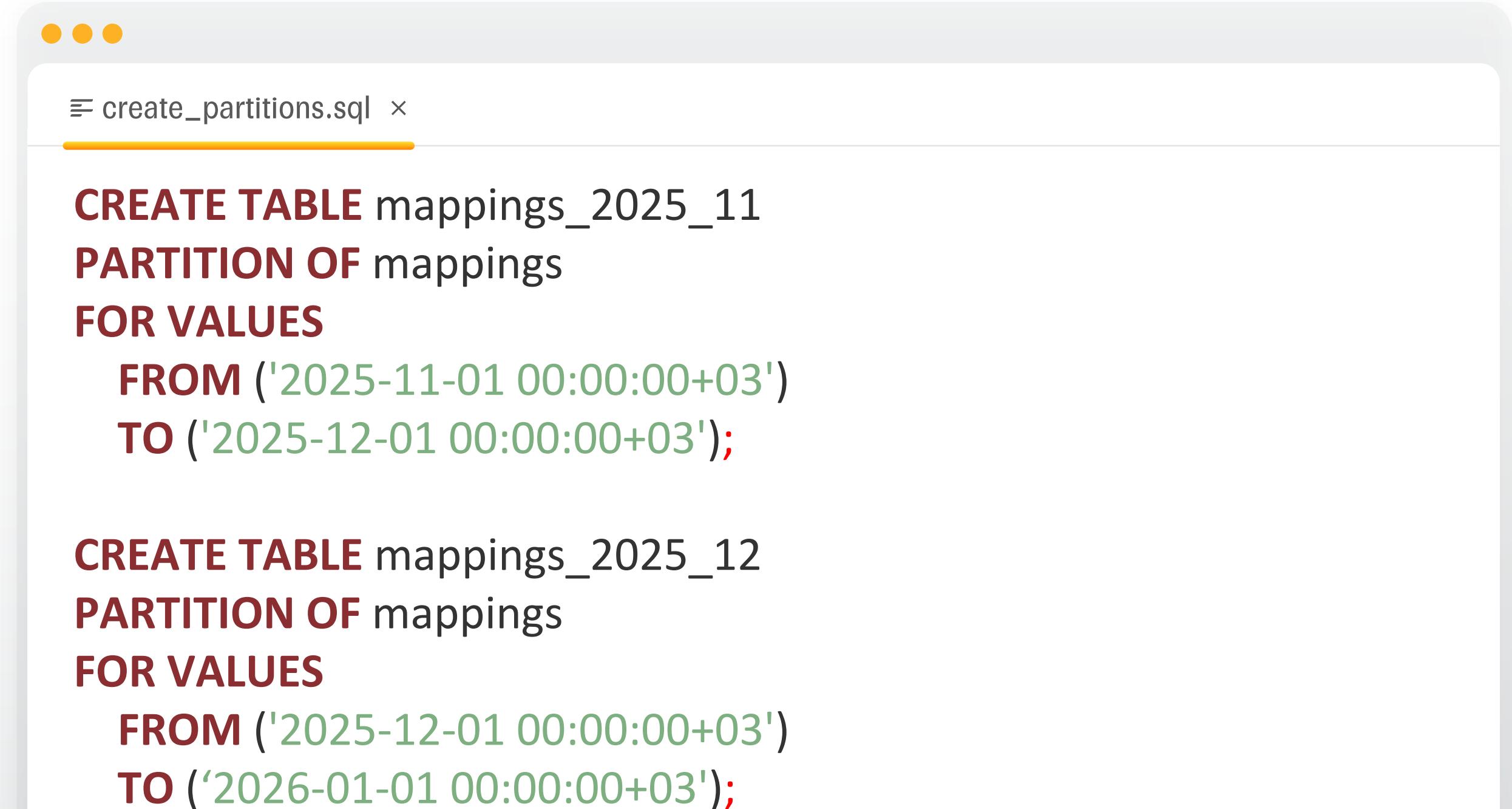
The screenshot shows a code editor window titled "create_partition_table.sql". The code is a SQL `CREATE TABLE` statement:

```
CREATE TABLE mappings (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ
) PARTITION BY RANGE (end_date);
```

The `PARTITION BY RANGE` clause is highlighted with a yellow box.

Создание партиций

- Нужно использовать конструкцию **PARTITION OF** и указать имя партицированной таблицы к которой она будет прикреплена;



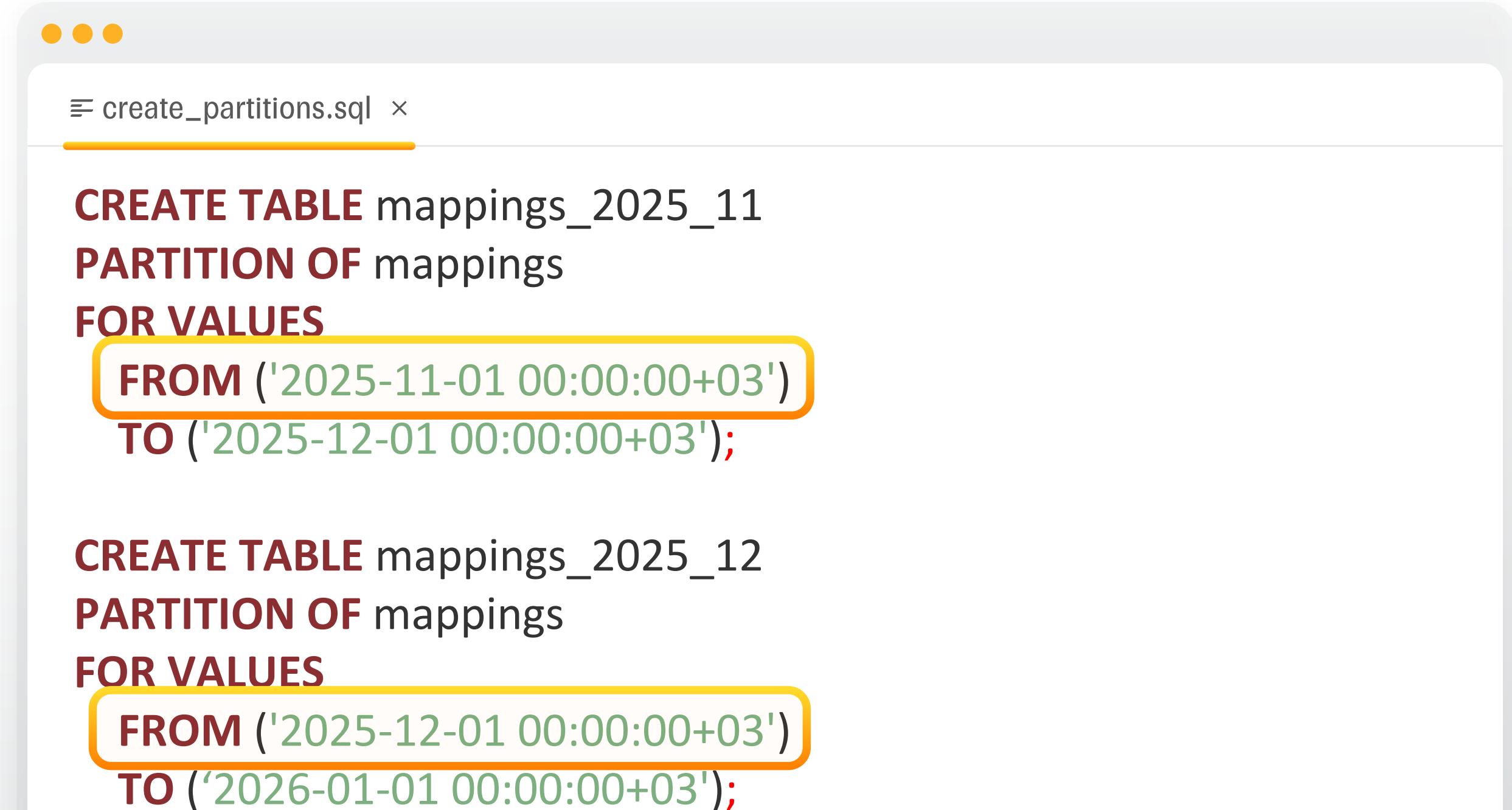
The screenshot shows a code editor window titled "create_partitions.sql". It contains two SQL statements for creating partitions of a table named "mappings".

```
CREATE TABLE mappings_2025_11
PARTITION OF mappings
FOR VALUES
FROM ('2025-11-01 00:00:00+03')
TO ('2025-12-01 00:00:00+03');

CREATE TABLE mappings_2025_12
PARTITION OF mappings
FOR VALUES
FROM ('2025-12-01 00:00:00+03')
TO ('2026-01-01 00:00:00+03');
```

Создание партиций

- Нужно использовать конструкцию **PARTITION OF** и указать имя партицированной таблицы к которой она будет прикреплена;
- Указать диапазон значений. **FROM** включительно



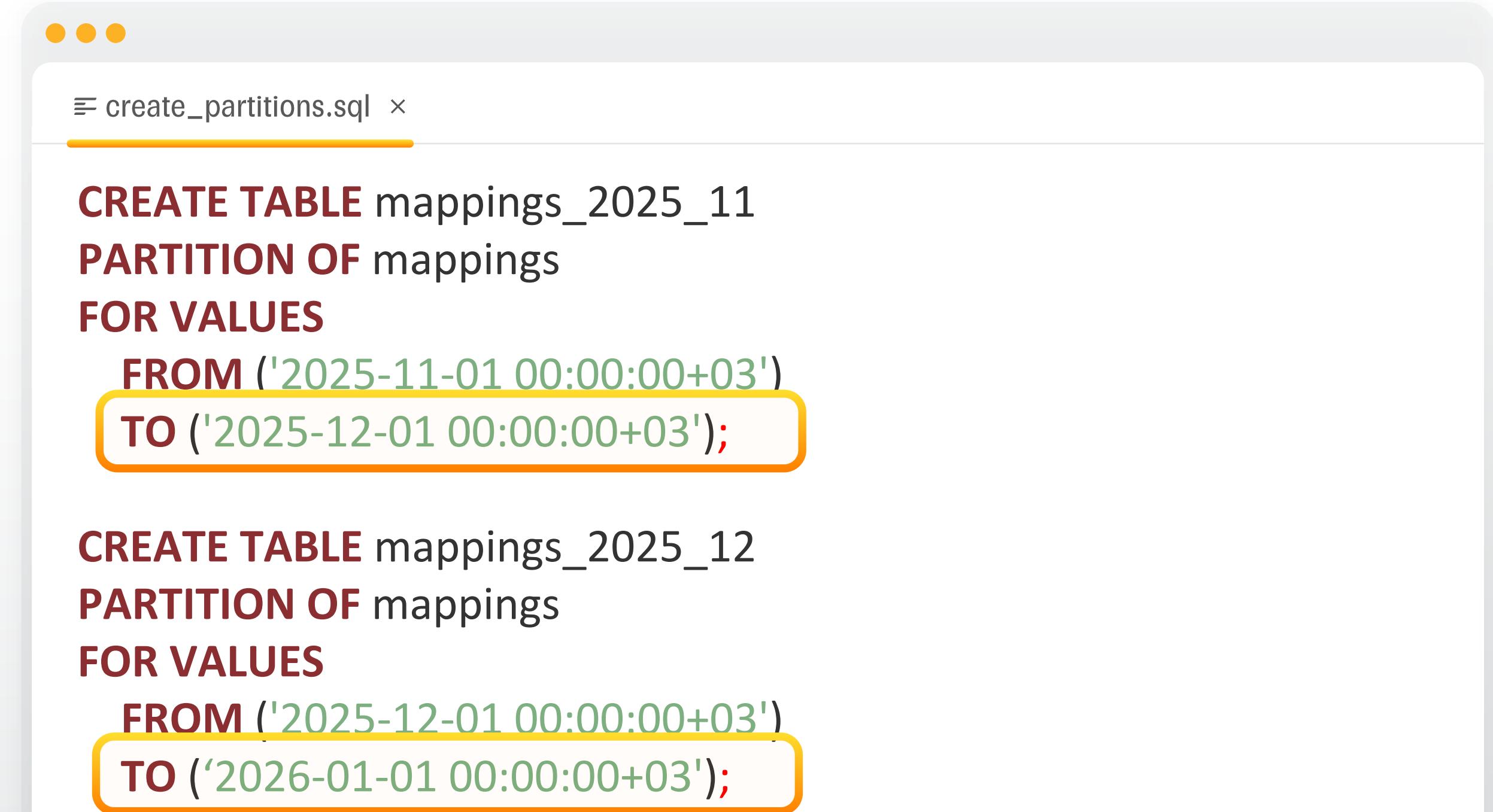
```
create_partitions.sql ×

CREATE TABLE mappings_2025_11
PARTITION OF mappings
FOR VALUES
    FROM ('2025-11-01 00:00:00+03')
    TO ('2025-12-01 00:00:00+03');

CREATE TABLE mappings_2025_12
PARTITION OF mappings
FOR VALUES
    FROM ('2025-12-01 00:00:00+03')
    TO ('2026-01-01 00:00:00+03');
```

Создание партиций

- Нужно использовать конструкцию **PARTITION OF** и указать имя партицированной таблицы к которой она будет прикреплена;
- Указать диапазон значений. **FROM** включительно, **TO** не включительно;



```
create_partitions.sql ×

CREATE TABLE mappings_2025_11
PARTITION OF mappings
FOR VALUES
    FROM ('2025-11-01 00:00:00+03')
    TO ('2025-12-01 00:00:00+03');

CREATE TABLE mappings_2025_12
PARTITION OF mappings
FOR VALUES
    FROM ('2025-12-01 00:00:00+03')
    TO ('2026-01-01 00:00:00+03');
```

Создание партиций

- Нужно использовать конструкцию **PARTITION OF** и указать имя партицированной таблицы к которой она будет прикреплена;
- Указать диапазон значений. **FROM** включительно, **TO** не включительно;
- Также можно создать партицию по умолчанию.

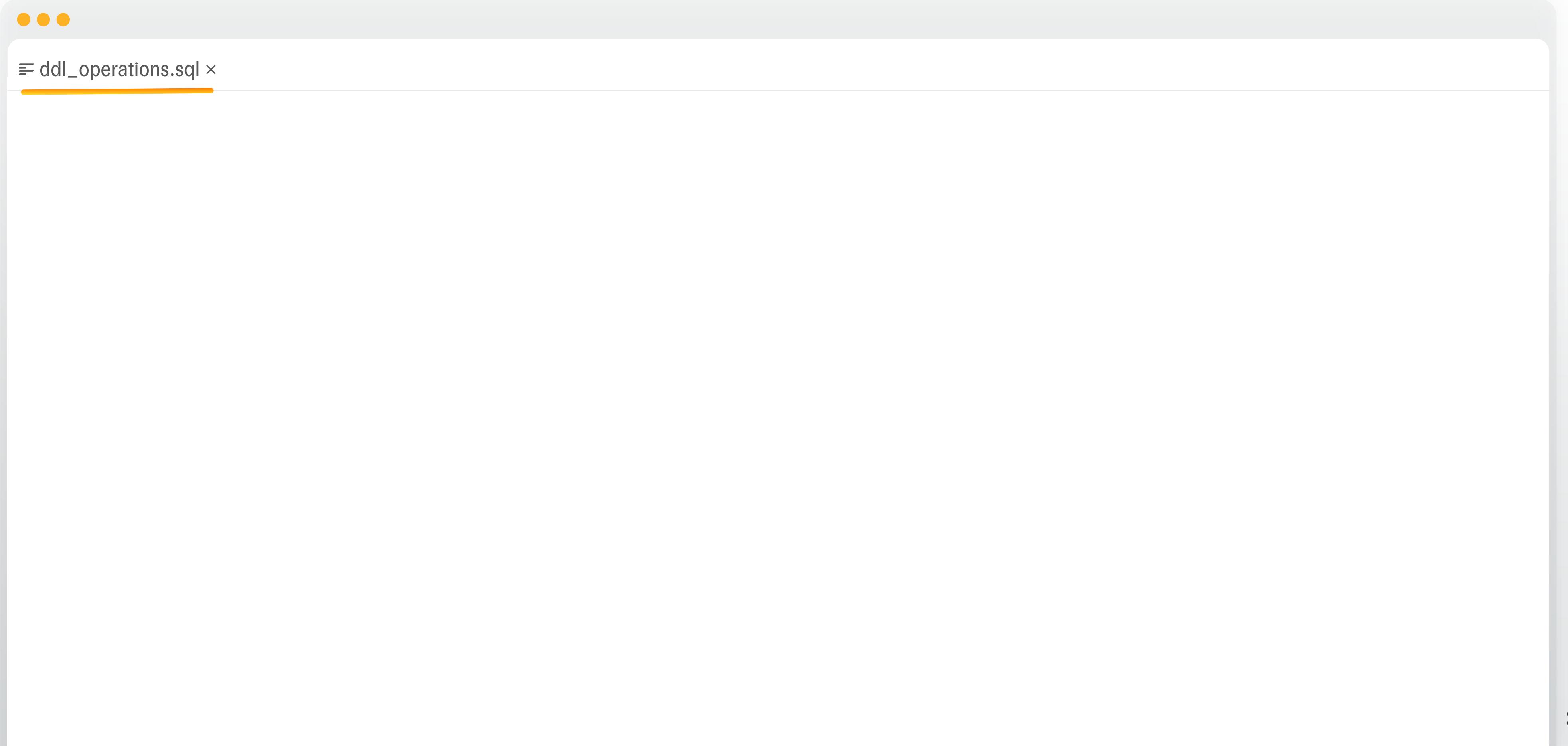
The screenshot shows a code editor window titled "create_partitions.sql". It contains three separate SQL statements for creating tables and partitioning them under a base table named "mappings".

```
CREATE TABLE mappings_2025_11
PARTITION OF mappings
FOR VALUES
    FROM ('2025-11-01 00:00:00+03')
    TO ('2025-12-01 00:00:00+03');

CREATE TABLE mappings_2025_12
PARTITION OF mappings
FOR VALUES
    FROM ('2025-12-01 00:00:00+03')
    TO ('2026-01-01 00:00:00+03');

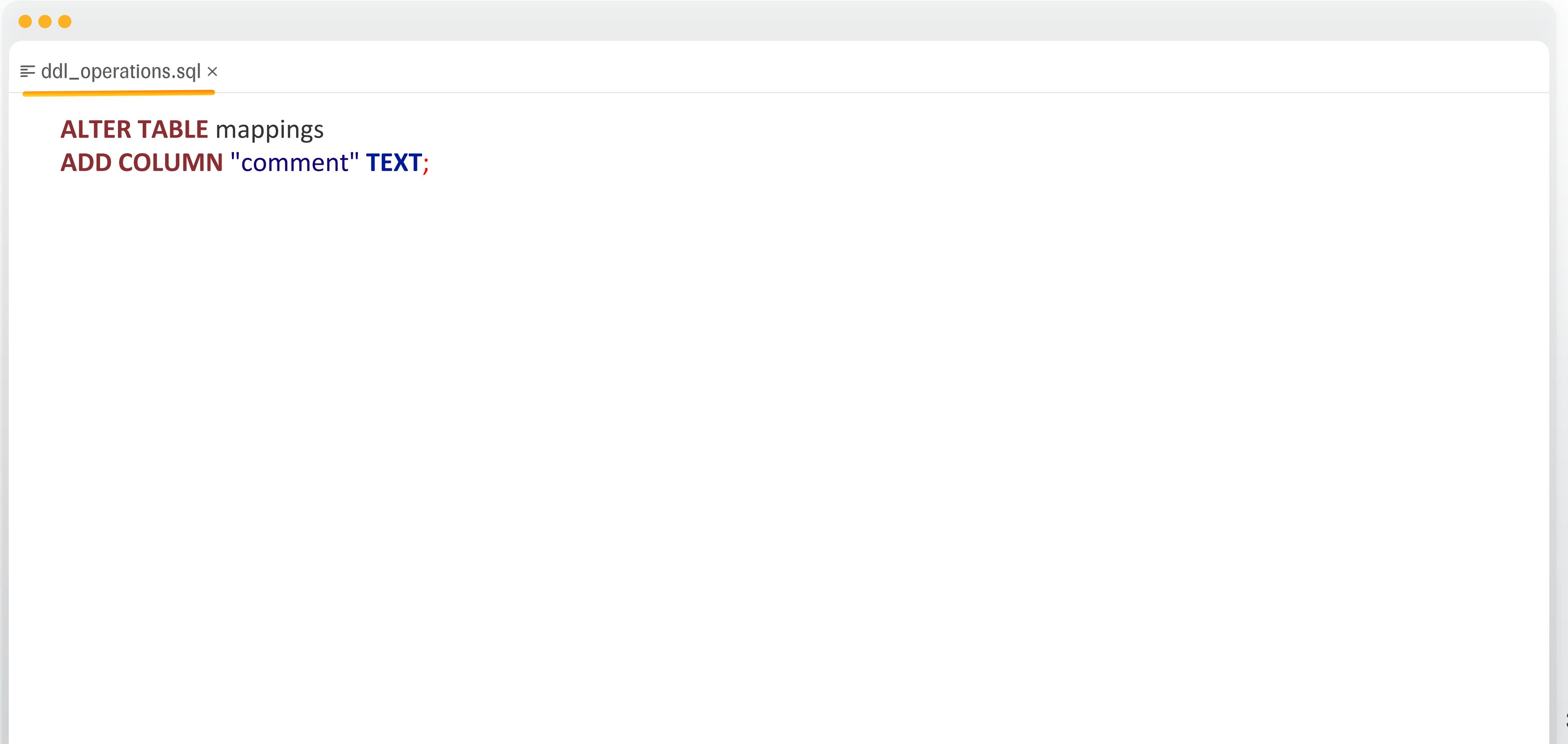
CREATE TABLE mappings_default
PARTITION OF mappings DEFAULT;
```

DDL операции



The image shows a screenshot of a code editor window. The title bar at the top says "ddl_operations.sql ×". The main area of the editor is completely blank, indicating that the file is currently empty. The interface includes standard window controls (minimize, maximize, close) in the top-left corner.

DDL операции

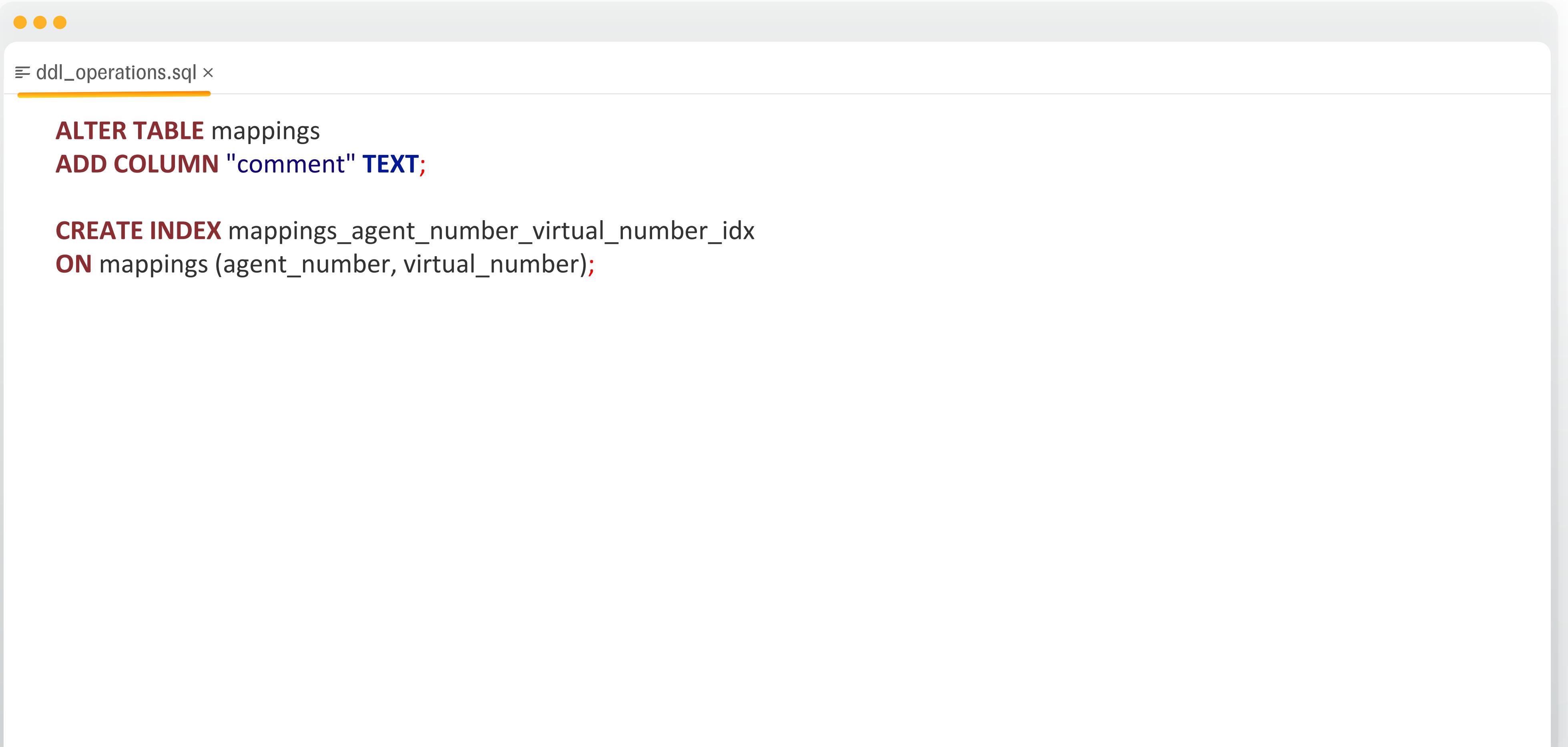


The image shows a screenshot of a code editor window. The title bar at the top says "ddl_operations.sql ×". Below the title bar, there is a toolbar with three yellow circular icons. The main area of the editor contains the following SQL code:

```
ALTER TABLE mappings
ADD COLUMN "comment" TEXT;
```

The code consists of two lines. The first line starts with "ALTER TABLE" followed by the table name "mappings". The second line starts with "ADD COLUMN" followed by the column definition "comment" and "TEXT;". The word "comment" is enclosed in double quotes, and "TEXT;" is in blue, likely indicating it is a keyword or a specific type.

DDL операции



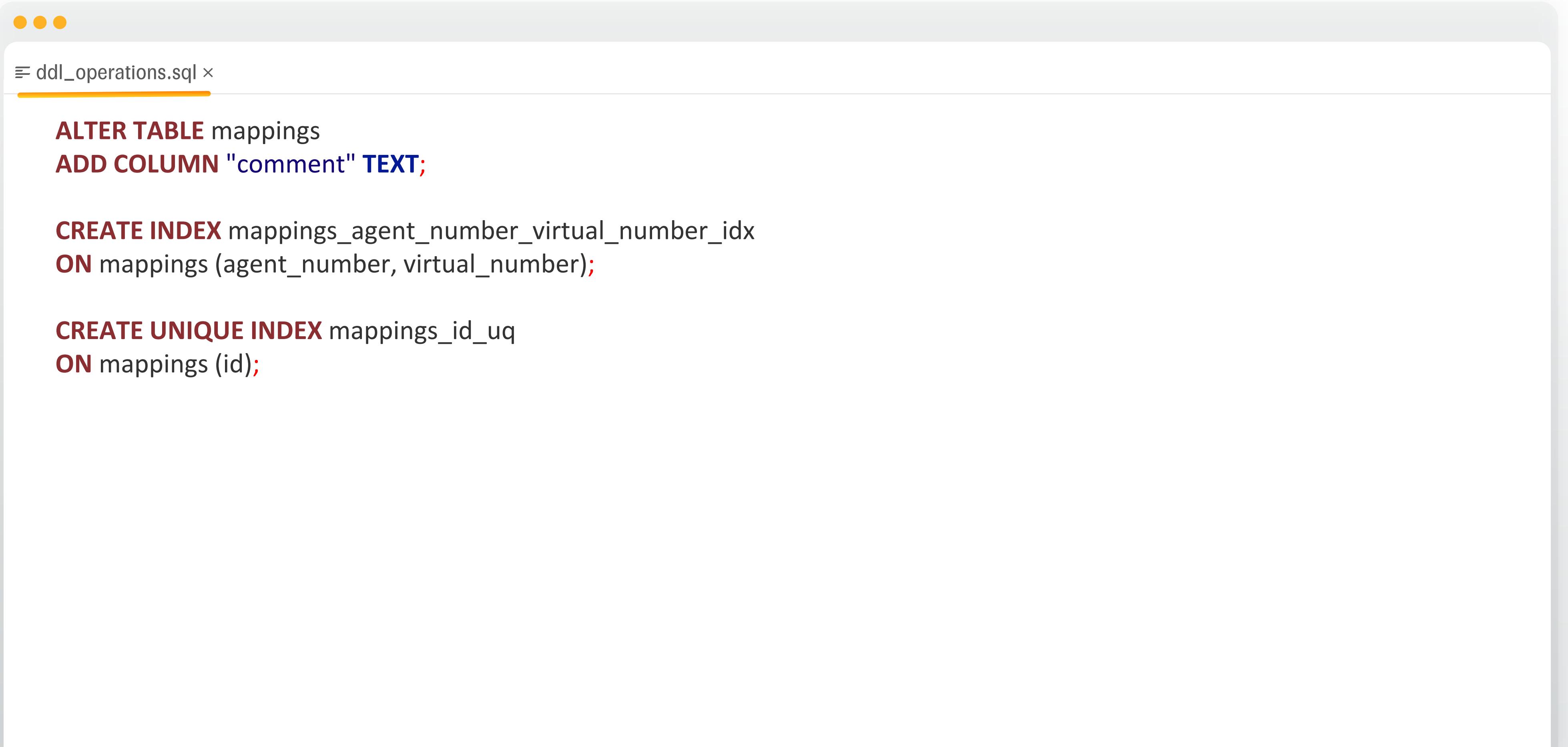
The screenshot shows a code editor window with a tab labeled "ddl_operations.sql". The editor displays two SQL statements:

```
ALTER TABLE mappings
ADD COLUMN "comment" TEXT;

CREATE INDEX mappings_agent_number_virtual_number_idx
ON mappings (agent_number, virtual_number);
```

The code is color-coded: "ALTER TABLE" and "CREATE INDEX" are in red, while the column name "comment" and index name "mappings_agent_number_virtual_number_idx" are in blue. The file path "ddl_operations.sql" is also highlighted in blue at the top of the editor.

DDL операции



The screenshot shows a code editor window with a tab labeled "ddl_operations.sql". The code contains three DDL statements:

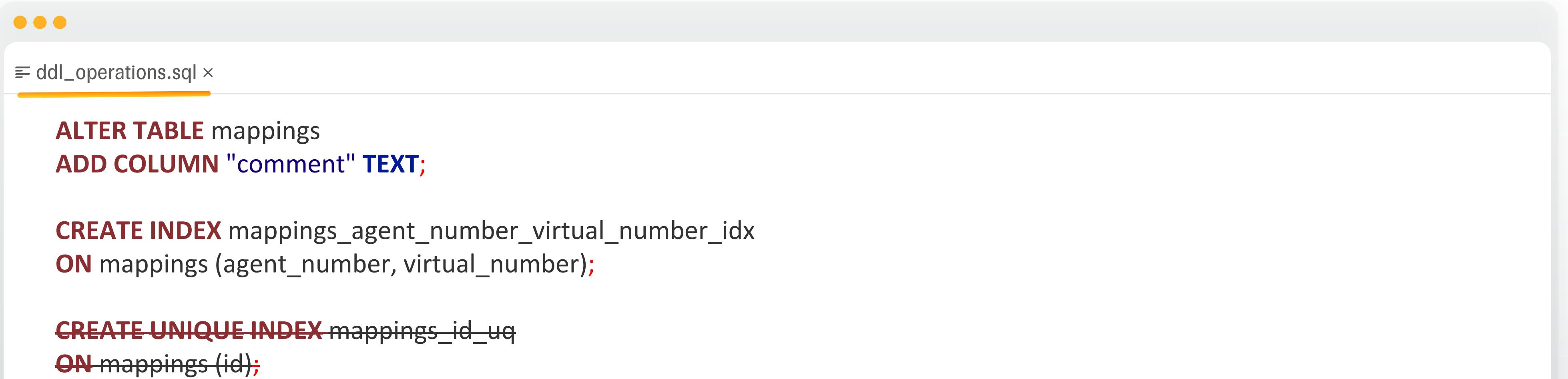
```
ALTER TABLE mappings
ADD COLUMN "comment" TEXT;

CREATE INDEX mappings_agent_number_virtual_number_idx
ON mappings (agent_number, virtual_number);

CREATE UNIQUE INDEX mappings_id_uq
ON mappings (id);
```

The code editor has a dark theme with syntax highlighting. The file name "ddl_operations.sql" is in grey, while the SQL keywords and identifiers are in red and blue respectively.

DDL операции



The screenshot shows a code editor window with a tab labeled "ddl_operations.sql". The code contains three DDL statements:

```
ALTER TABLE mappings
ADD COLUMN "comment" TEXT;

CREATE INDEX mappings_agent_number_virtual_number_idx
ON mappings (agent_number, virtual_number);

CREATE UNIQUE INDEX mappings_id_uq
ON mappings (id);
```

DDL операции

```
• • •  
≡ ddl_operations.sql ×  
  
ALTER TABLE mappings  
ADD COLUMN "comment" TEXT;  
  
CREATE INDEX mappings_agent_number_virtual_number_idx  
ON mappings (agent_number, virtual_number);  
  
CREATE UNIQUE INDEX mappings_id_uq  
ON mappings (id);  
  
CREATE UNIQUE INDEX mappings_id_end_date_uq  
ON mappings (id, end_date);
```

DDL операции

The screenshot shows a code editor window with a tab labeled 'ddl_operations.sql'. The code contains several DDL statements:

```
ALTER TABLE mappings
ADD COLUMN "comment" TEXT;

CREATE INDEX mappings_agent_number_virtual_number_idx
ON mappings (agent_number, virtual_number);

CREATE UNIQUE INDEX mappings_id_uq
ON mappings (id);

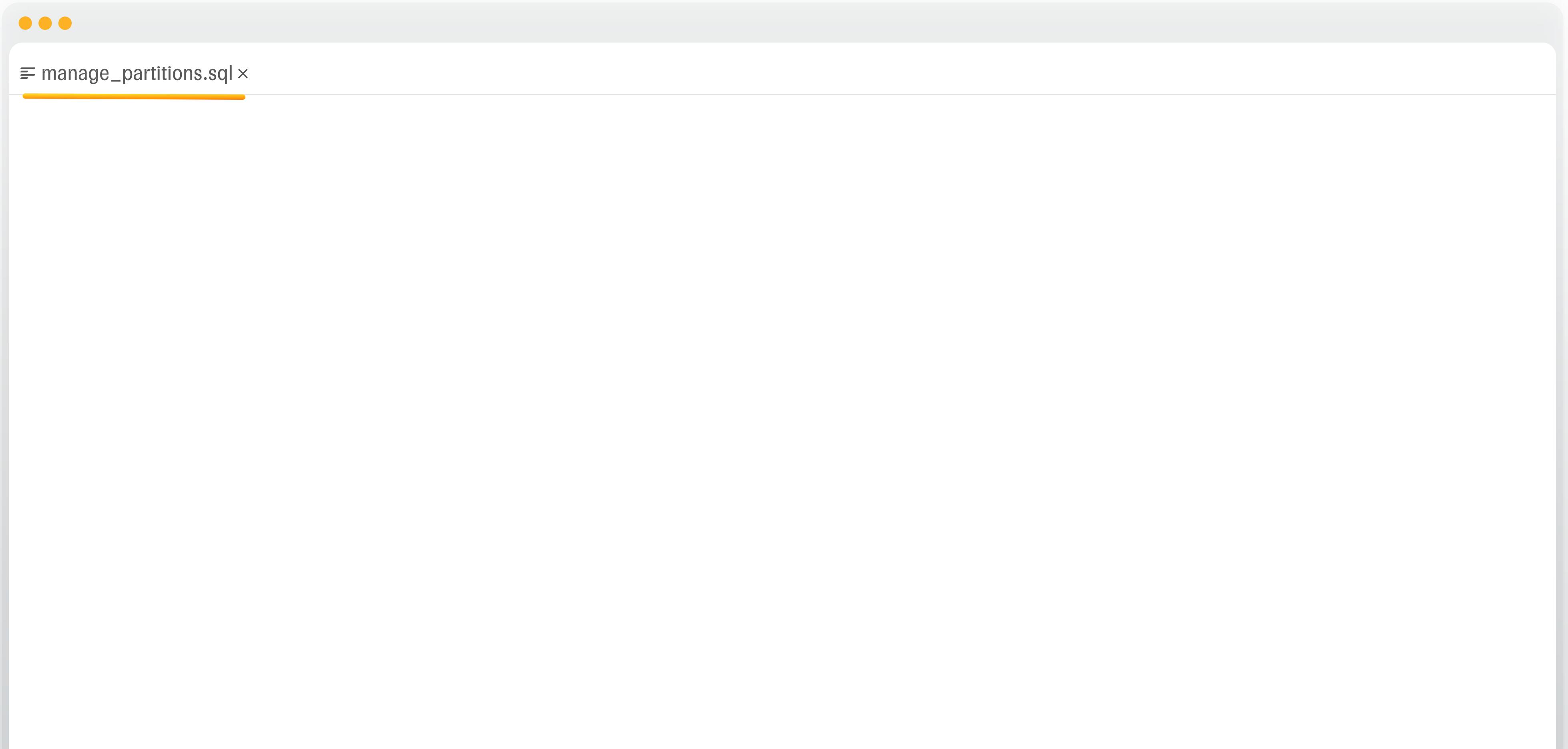
CREATE UNIQUE INDEX mappings_id_end_date_uq
ON mappings (id, end_date);

CREATE UNIQUE INDEX mappings_2025_11_id_uq
ON mappings_2025_11 (id);
```



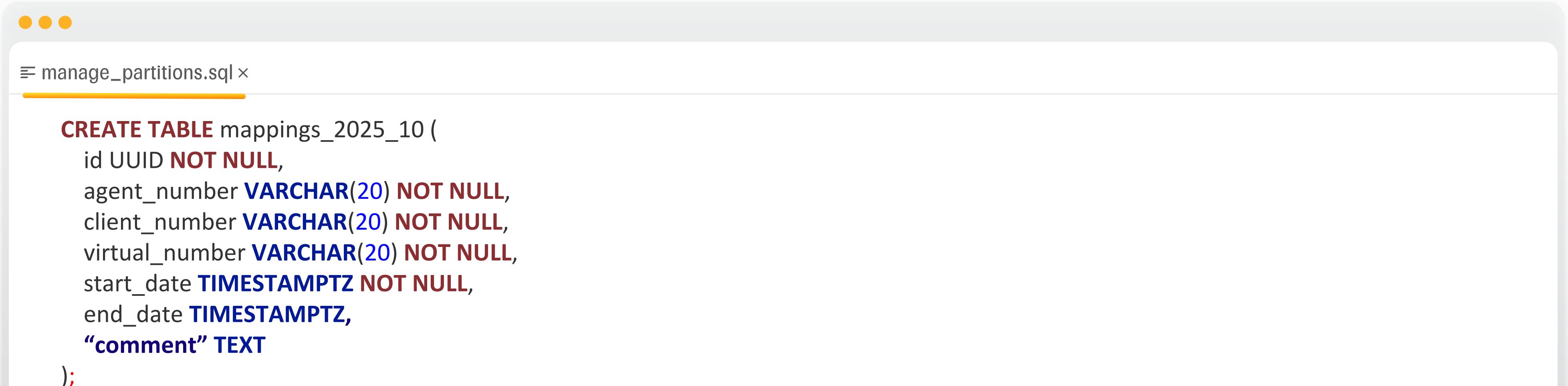
**Партиционированная
таблица – шаблон
дляパーティй.**

Управление партициями



A screenshot of a code editor window titled "manage_partitions.sql". The file is currently empty, indicated by the three yellow dots at the top of the editor area. The file name is displayed in the title bar, and there is a small "x" icon to its right.

Управление партициями



The screenshot shows a code editor window with a tab labeled "manage_partitions.sql". The code itself is a SQL `CREATE TABLE` statement:

```
CREATE TABLE mappings_2025_10 (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ,
    "comment" TEXT
);
```

Управление партициями



The screenshot shows a code editor window with a tab labeled "manage_partitions.sql". The code in the editor is as follows:

```
CREATE TABLE mappings_2025_10 (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ,
    "comment" TEXT
);

ALTER TABLE mappings ATTACH PARTITION mappings_2025_10
FOR VALUES FROM ('2025-10-01 00:00:00+03') TO ('2025-11-01 00:00:00+03');
```

Управление партициями

```
manage_partitions.sql ×

CREATE TABLE mappings_2025_10 (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ,
    "comment" TEXT
);

ALTER TABLE mappings ATTACH PARTITION mappings_2025_10
FOR VALUES FROM ('2025-10-01 00:00:00+03') TO ('2025-11-01 00:00:00+03');
```

```
ALTER TABLE mappings
DETACH PARTITION mappings_2025_10;
```

Управление партициями

```
manage_partitions.sql ×

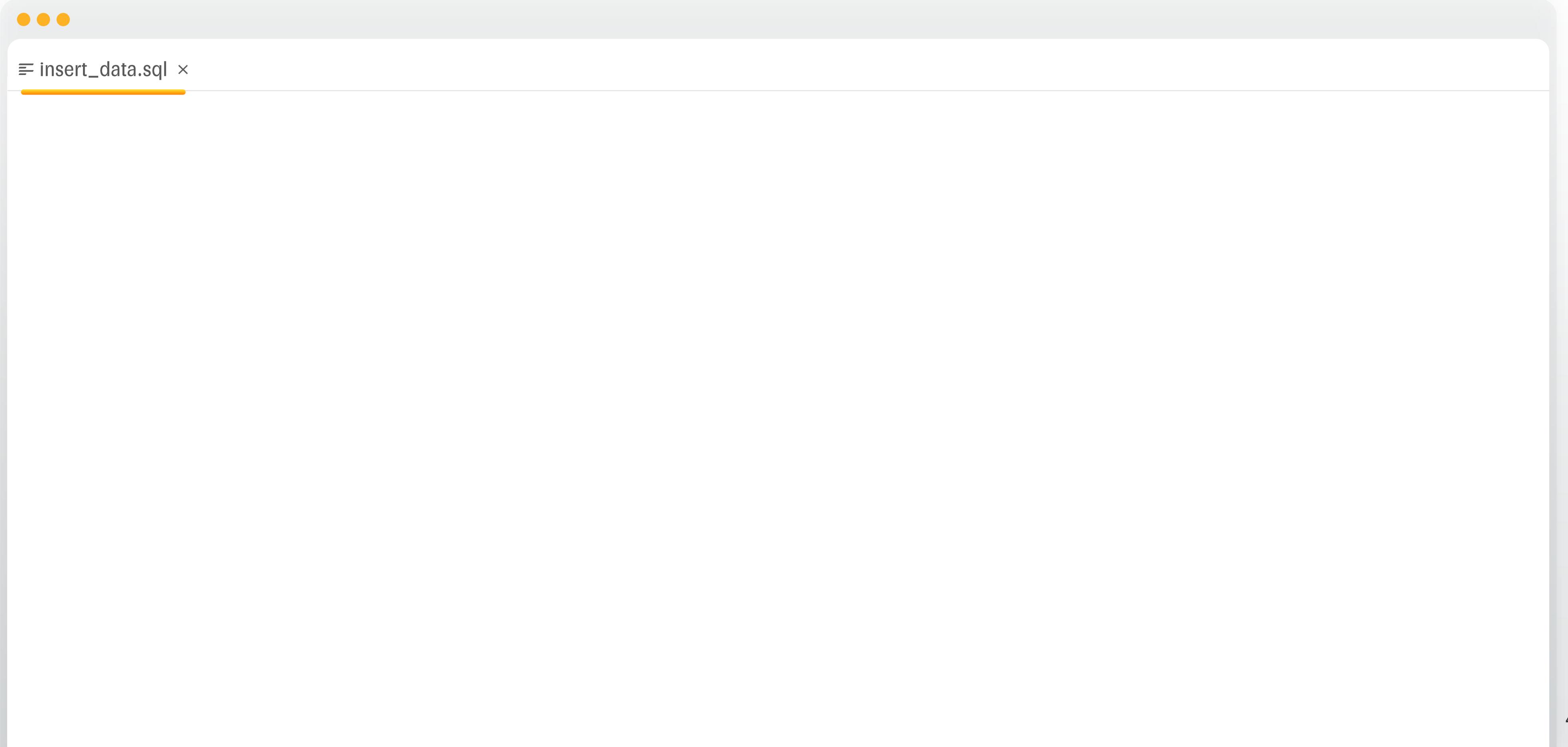
CREATE TABLE mappings_2025_10 (
    id UUID NOT NULL,
    agent_number VARCHAR(20) NOT NULL,
    client_number VARCHAR(20) NOT NULL,
    virtual_number VARCHAR(20) NOT NULL,
    start_date TIMESTAMPTZ NOT NULL,
    end_date TIMESTAMPTZ,
    "comment" TEXT
);

ALTER TABLE mappings ATTACH PARTITION mappings_2025_10
FOR VALUES FROM ('2025-10-01 00:00:00+03') TO ('2025-11-01 00:00:00+03');

ALTER TABLE mappings
DETACH PARTITION mappings_2025_10;

DROP TABLE mappings_2025_10;
```

Вставка данных



A screenshot of a code editor window titled "insert_data.sql". The title bar has three yellow circular icons. The file name "insert_data.sql" is displayed next to the title bar, with a small "x" icon to its right. A horizontal orange line highlights the title bar area. The main editor area is completely blank, showing only the white background of the code editor.

Вставка данных



The image shows a screenshot of a code editor window titled "insert_data.sql". The code editor has a light gray background with a white main area. At the top left, there are three small yellow circular icons. The file name "insert_data.sql" is displayed in a dark gray font, followed by a close button (an "x"). A horizontal orange bar is positioned below the title bar. The main content area contains the following SQL code:

```
INSERT INTO mappings (... , end_date)  
VALUES (... , '2025-11-21 19:00:00+03');
```

The word "INSERT INTO" is highlighted in red, while the rest of the code is in black. The date string "2025-11-21 19:00:00+03" is highlighted in green.

Вставка данных



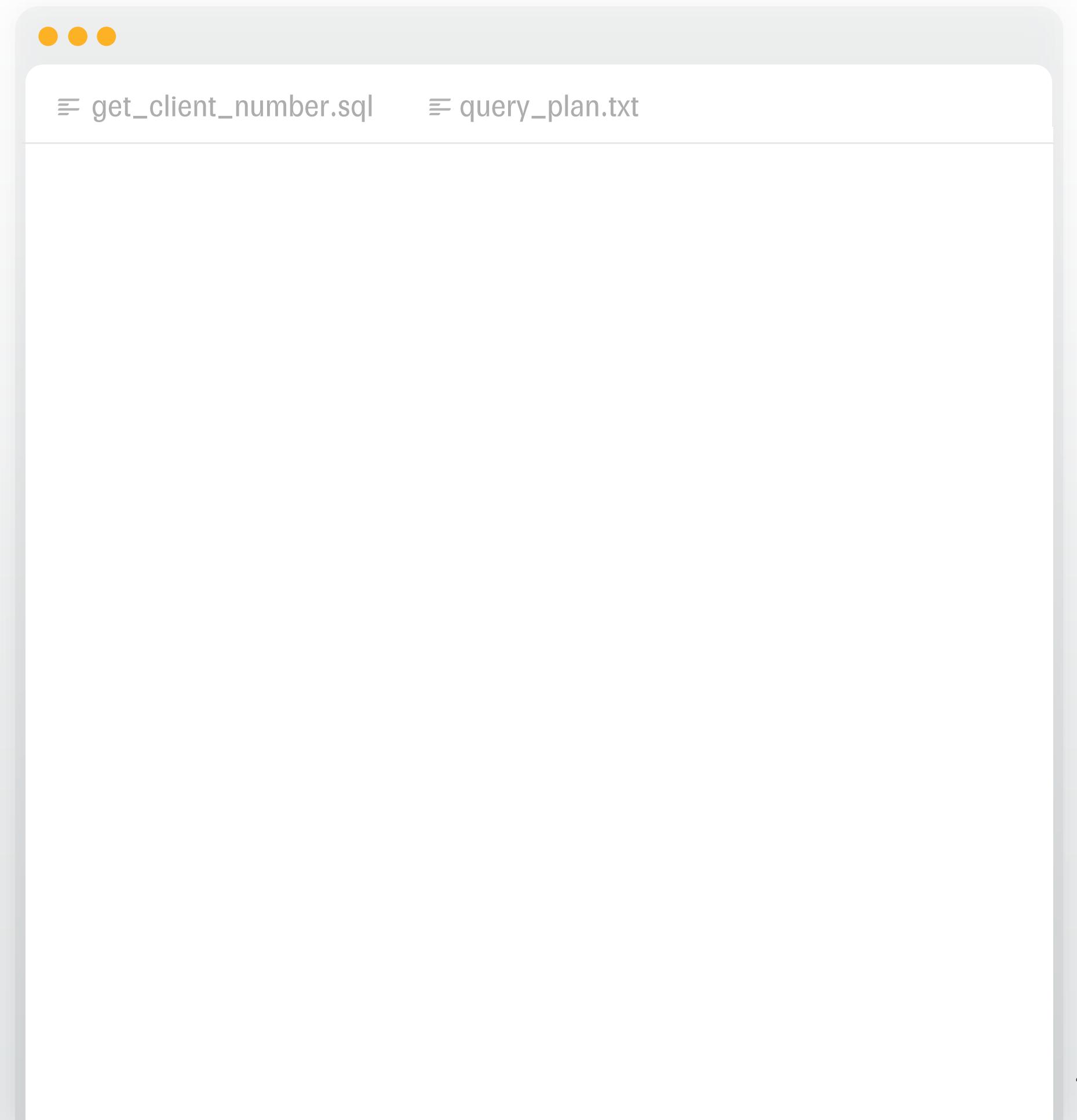
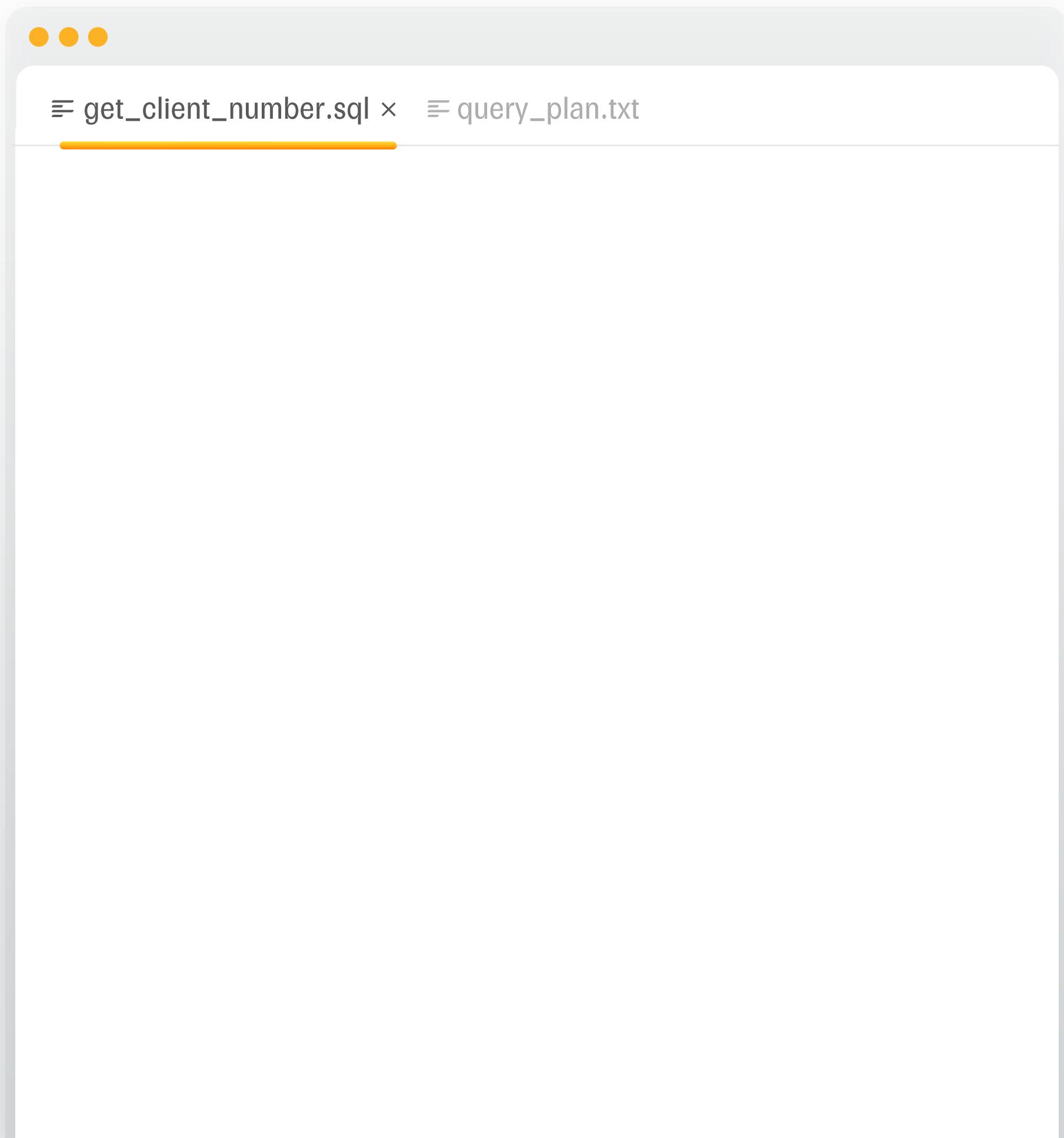
The screenshot shows a code editor window with a tab labeled "insert_data.sql". The content of the file contains two SQL INSERT statements:

```
INSERT INTO mappings (... , end_date)
VALUES (... , '2025-11-21 19:00:00+03');

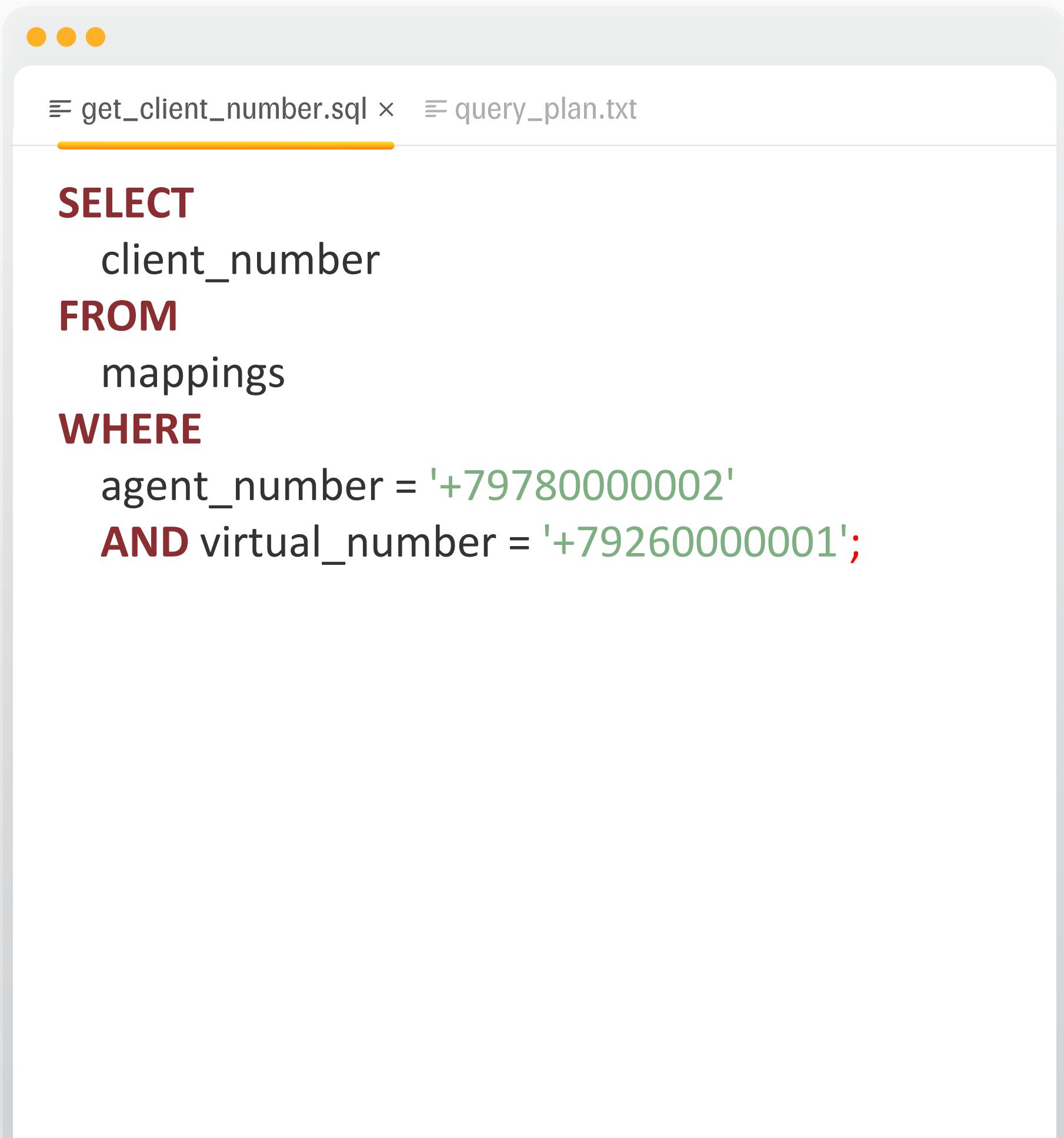
INSERT INTO mappings (... , end_date)
VALUES (... , '2077-01-01 19:00:00+03');
```

The second "VALUES" clause in the second statement is highlighted with a yellow rectangular selection.

Чтение данных

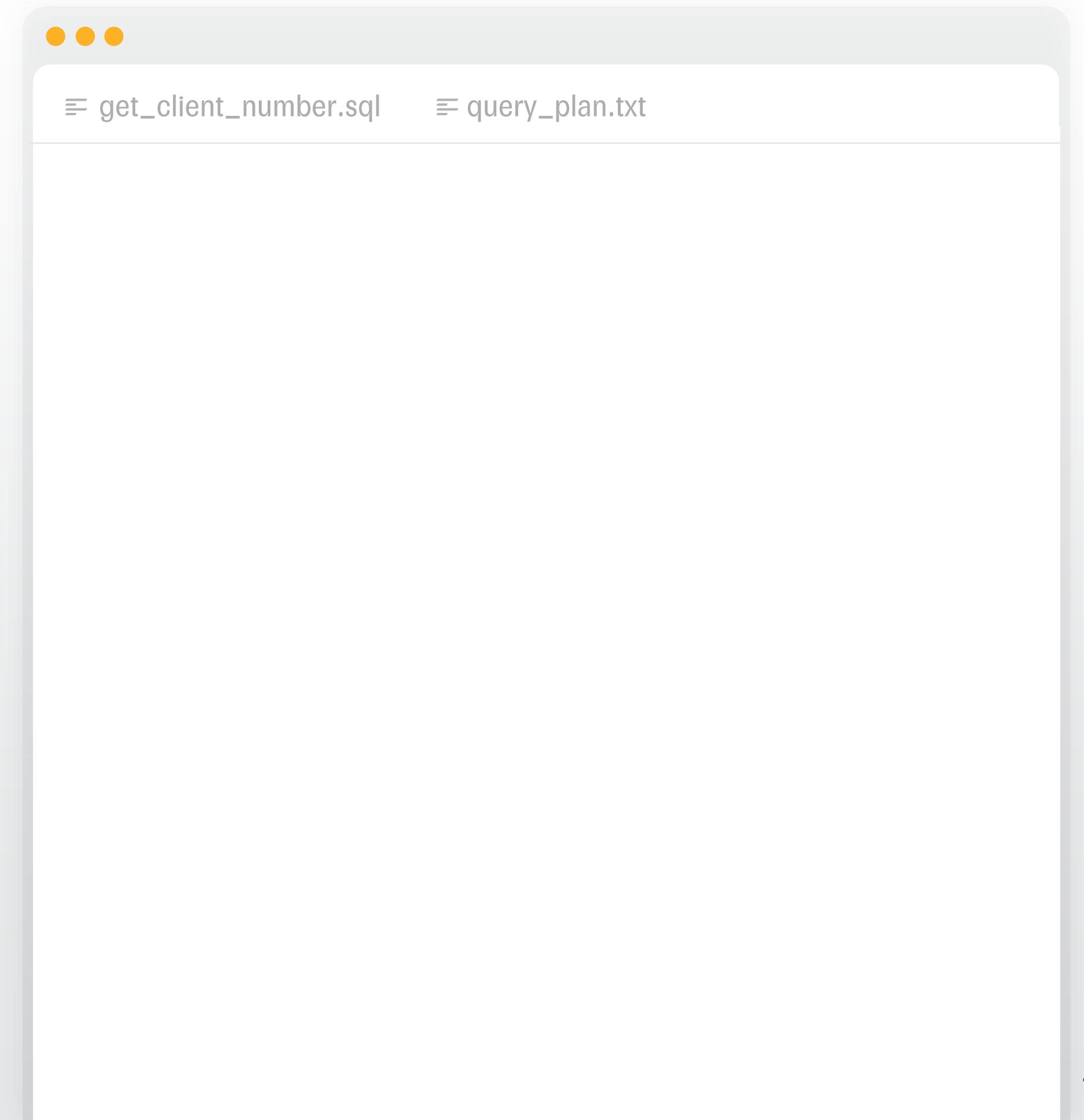


Чтение данных



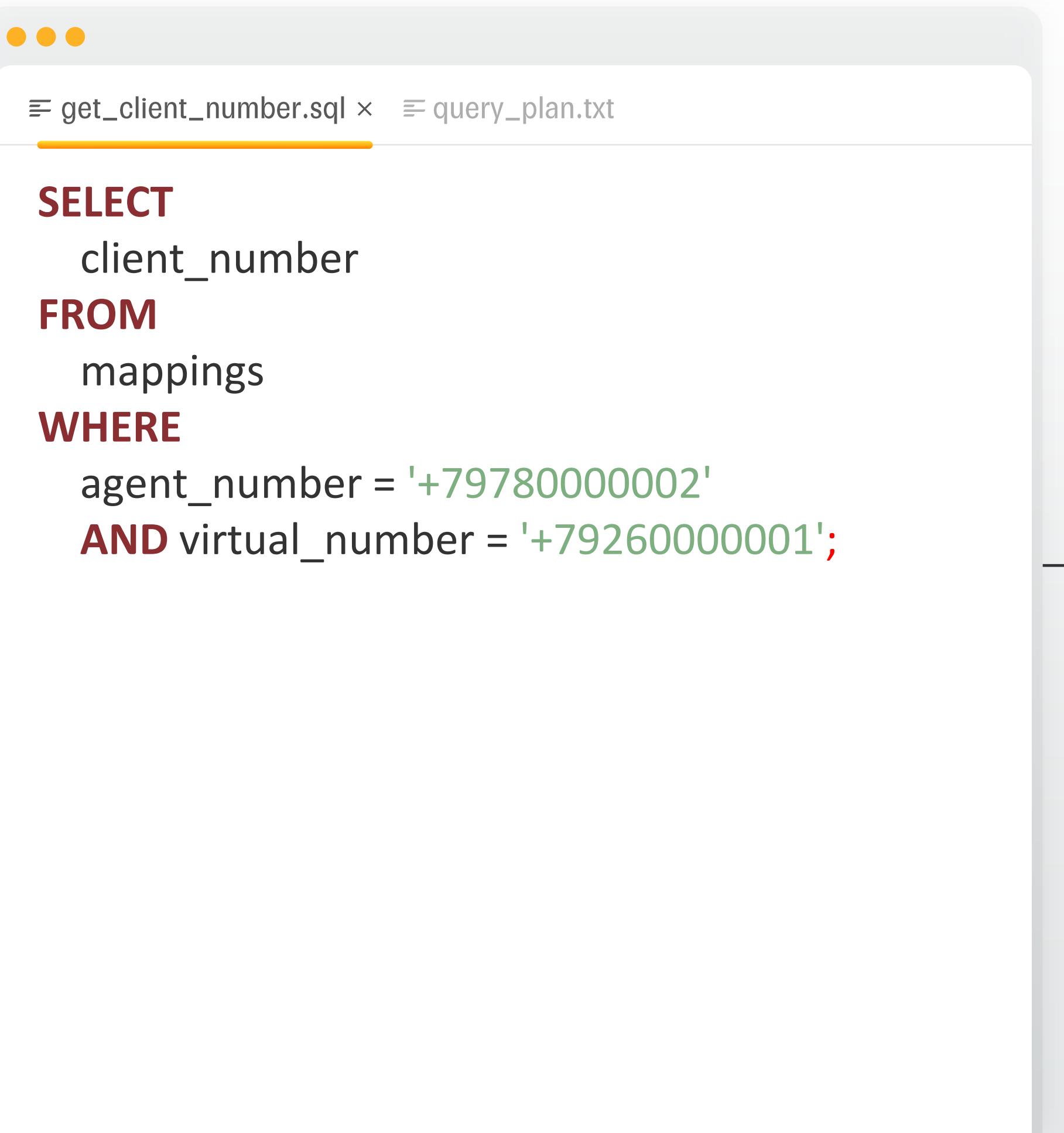
The screenshot shows a Mac OS X application window with two tabs: 'get_client_number.sql' (selected) and 'query_plan.txt'. The code in 'get_client_number.sql' is:

```
SELECT client_number
FROM mappings
WHERE agent_number = '+79780000002'
AND virtual_number = '+79260000001';
```



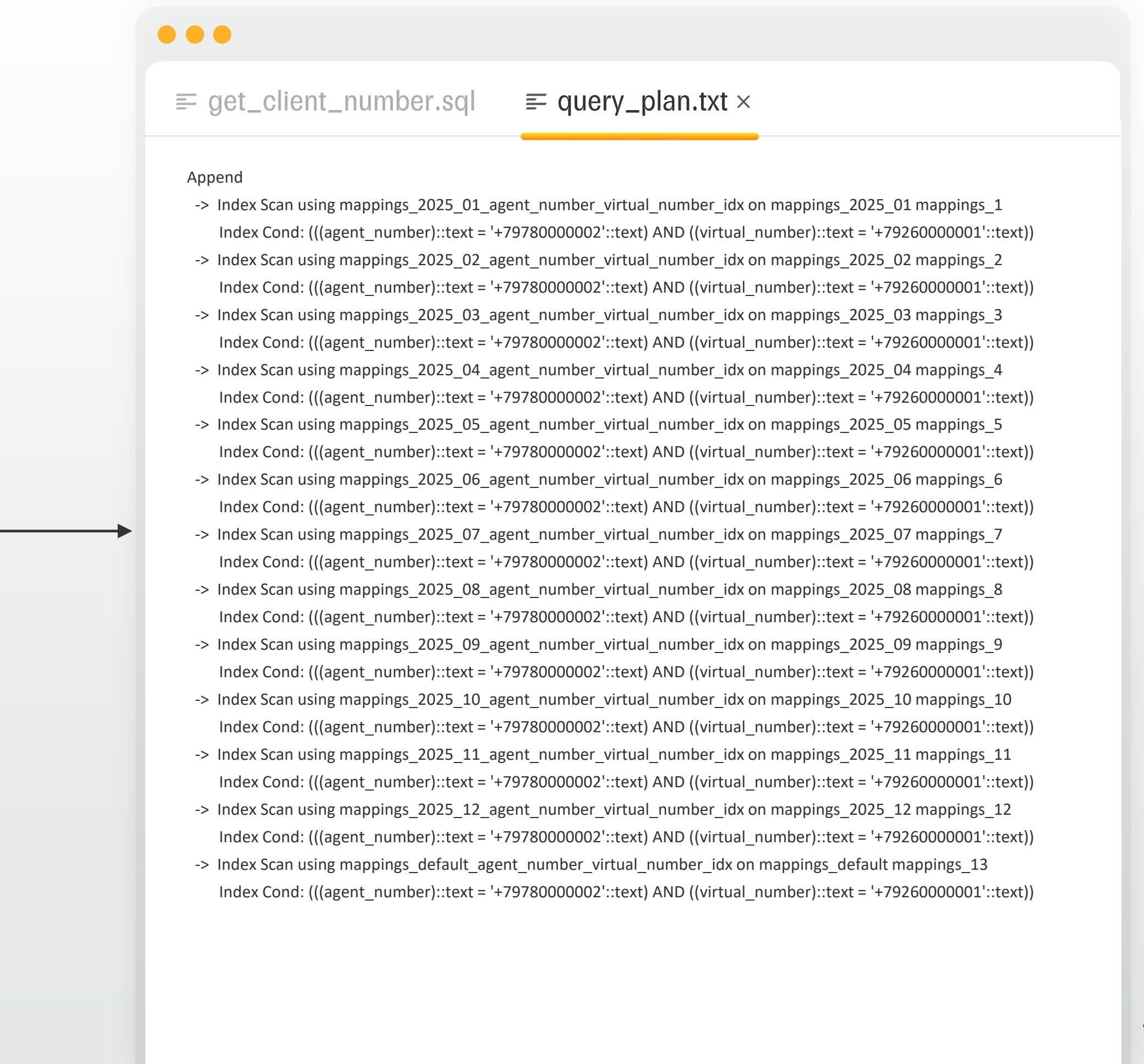
The screenshot shows a Mac OS X application window with two tabs: 'get_client_number.sql' and 'query_plan.txt' (selected). The content of 'query_plan.txt' is empty.

Чтение данных



get_client_number.sql x query_plan.txt

```
SELECT
    client_number
FROM
    mappings
WHERE
    agent_number = '+79780000002'
    AND virtual_number = '+79260000001';
```

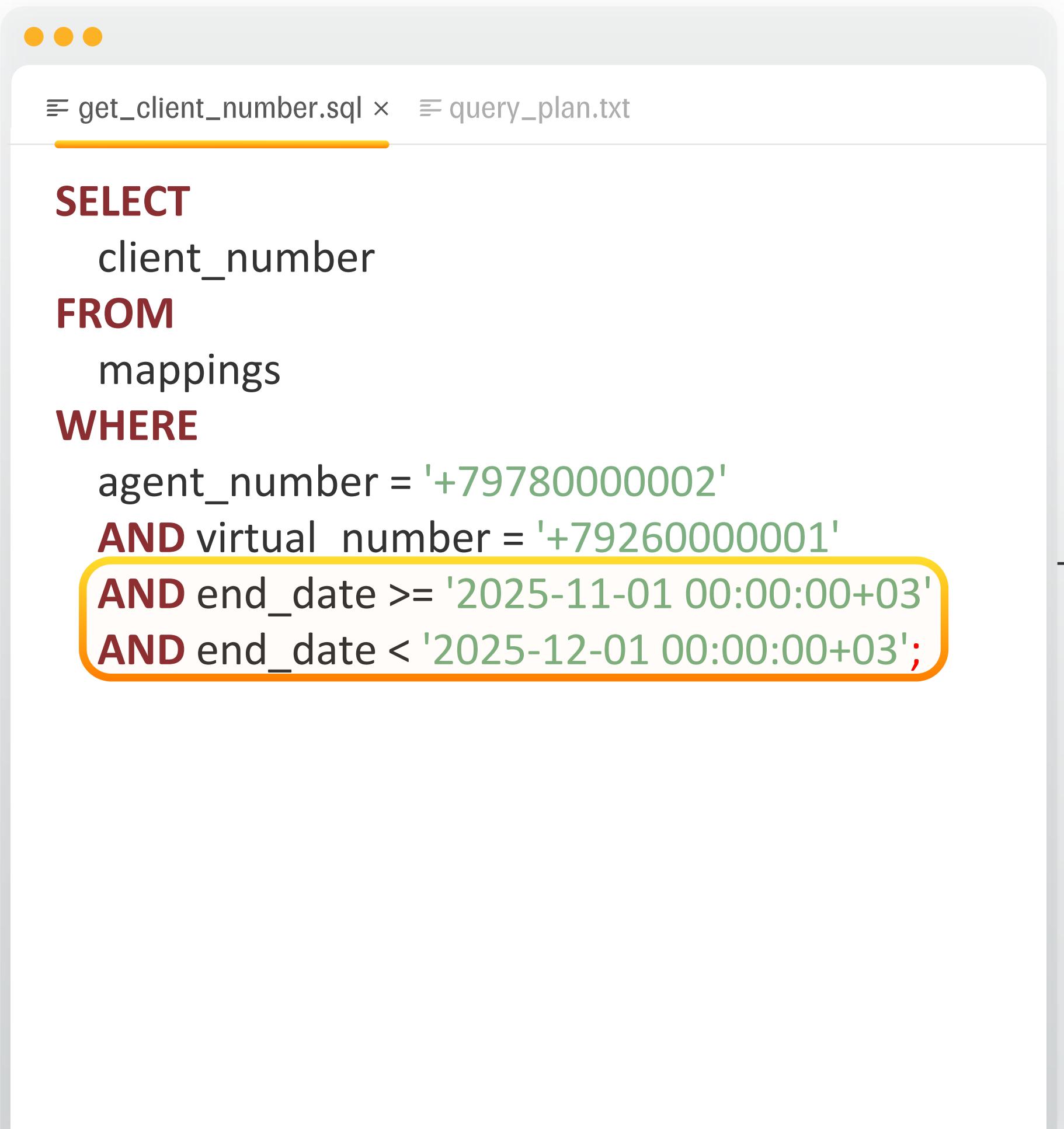


get_client_number.sql x query_plan.txt

Append

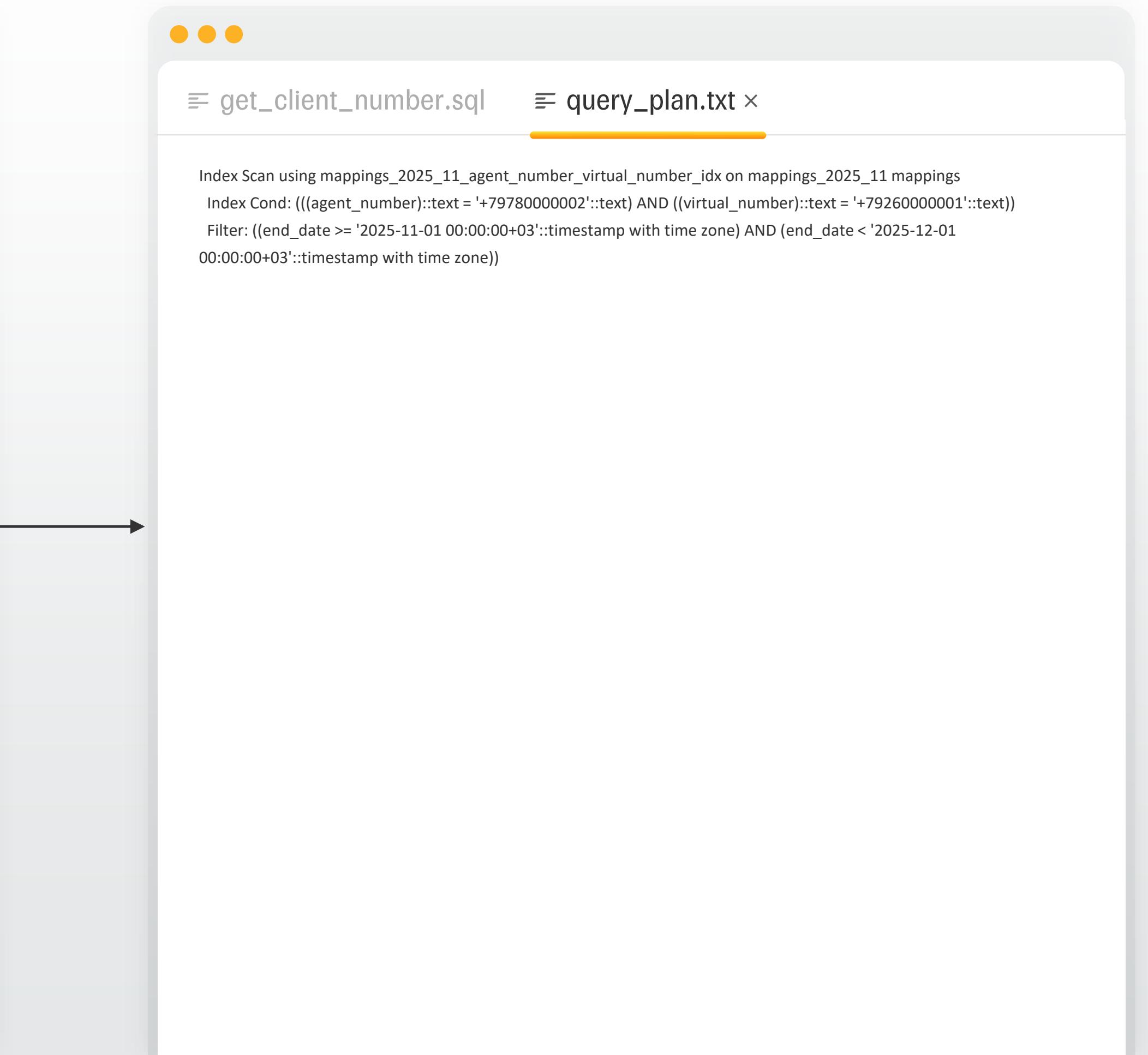
```
-> Index Scan using mappings_2025_01_agent_number_virtual_number_idx on mappings_2025_01 mappings_1
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_02_agent_number_virtual_number_idx on mappings_2025_02 mappings_2
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_03_agent_number_virtual_number_idx on mappings_2025_03 mappings_3
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_04_agent_number_virtual_number_idx on mappings_2025_04 mappings_4
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_05_agent_number_virtual_number_idx on mappings_2025_05 mappings_5
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_06_agent_number_virtual_number_idx on mappings_2025_06 mappings_6
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_07_agent_number_virtual_number_idx on mappings_2025_07 mappings_7
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_08_agent_number_virtual_number_idx on mappings_2025_08 mappings_8
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_09_agent_number_virtual_number_idx on mappings_2025_09 mappings_9
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_10_agent_number_virtual_number_idx on mappings_2025_10 mappings_10
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_11_agent_number_virtual_number_idx on mappings_2025_11 mappings_11
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_2025_12_agent_number_virtual_number_idx on mappings_2025_12 mappings_12
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
-> Index Scan using mappings_default_agent_number_virtual_number_idx on mappings_default mappings_13
    Index Cond: (((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text))
```

Чтение данных



```
get_client_number.sql ×  query_plan.txt

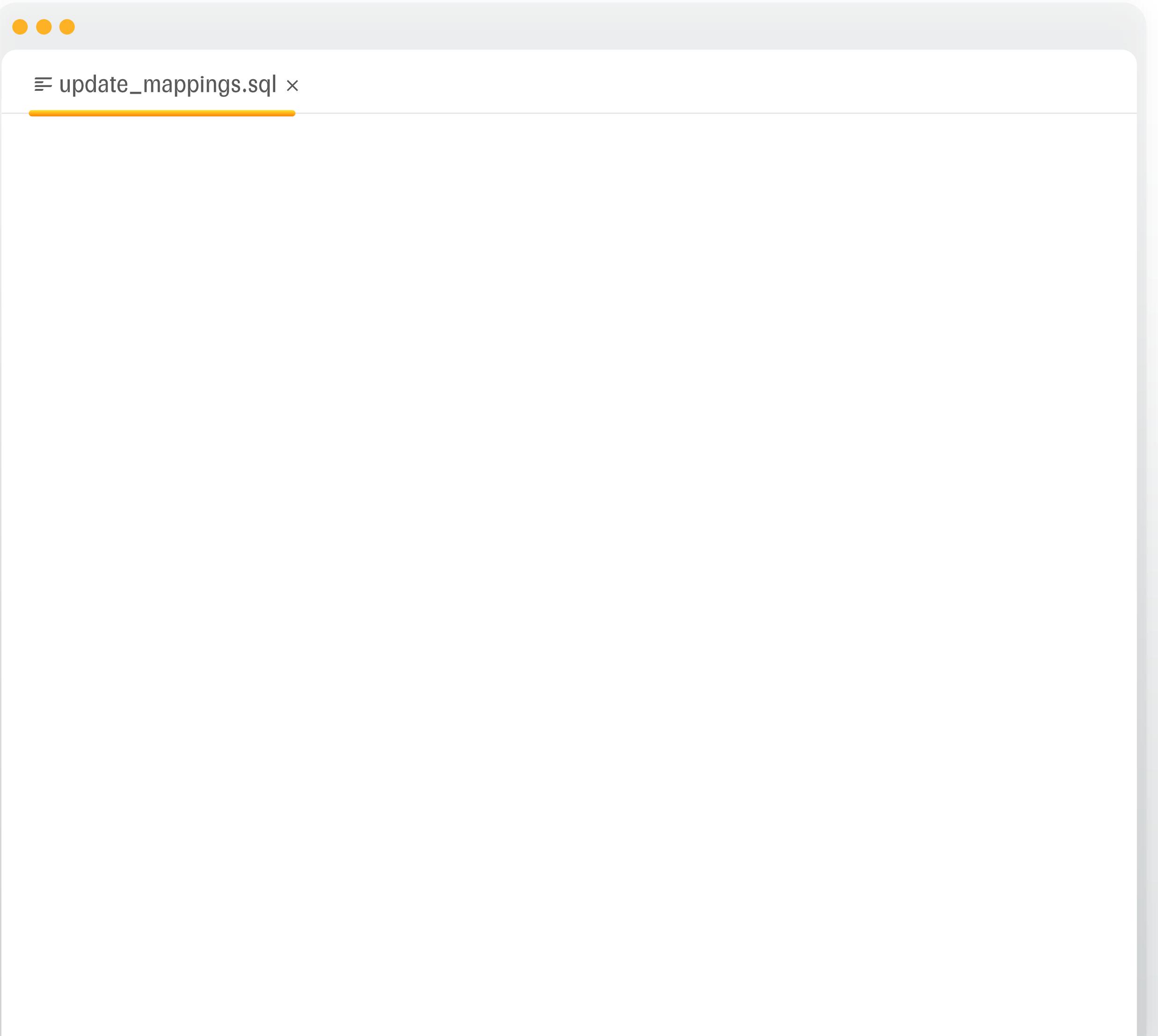
SELECT
    client_number
FROM
    mappings
WHERE
    agent_number = '+79780000002'
    AND virtual_number = '+79260000001'
    AND end_date >= '2025-11-01 00:00:00+03'
    AND end_date < '2025-12-01 00:00:00+03';
```



```
get_client_number.sql  query_plan.txt ×

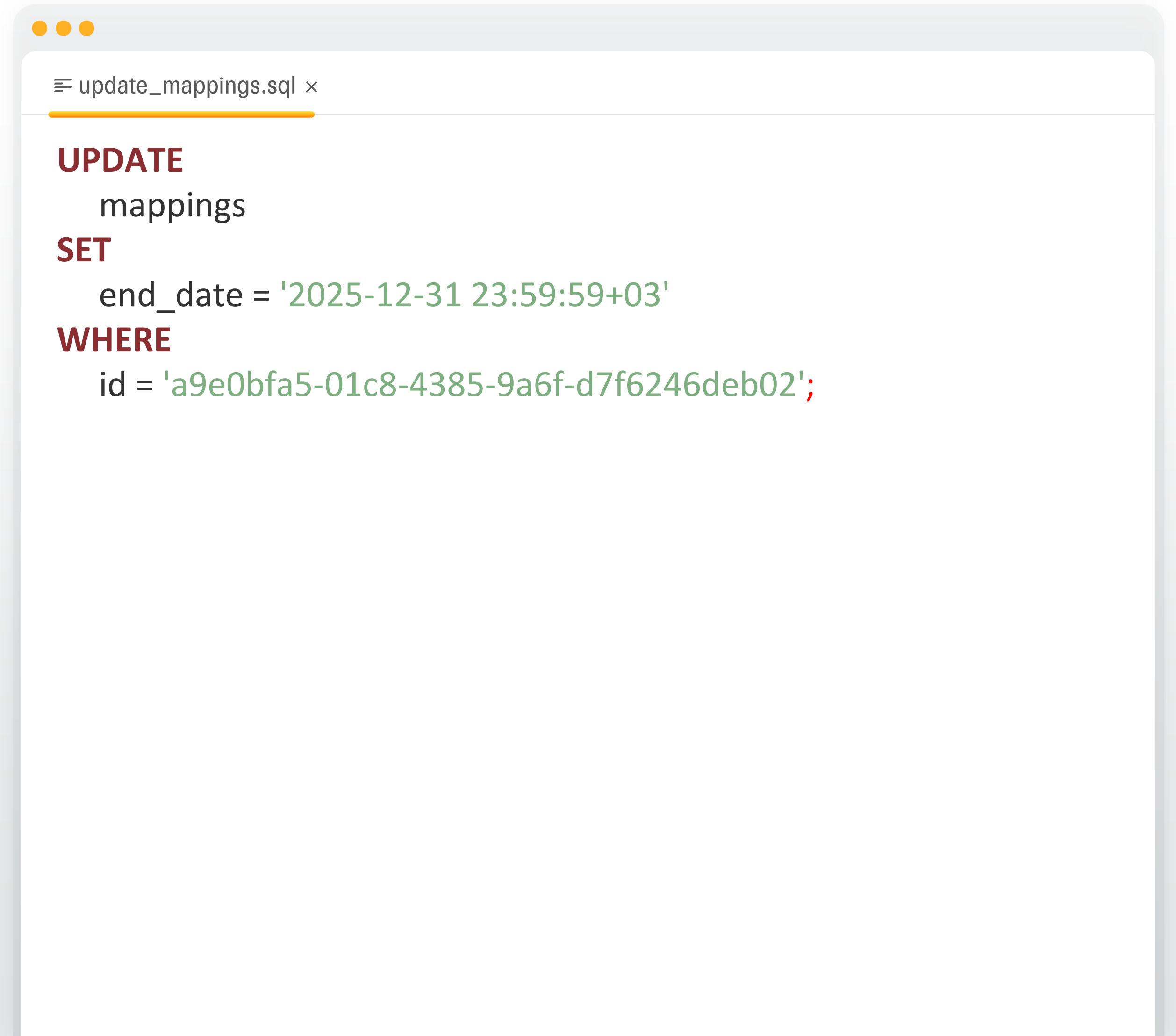
Index Scan using mappings_2025_11_agent_number_virtual_number_idx on mappings_2025_11 mappings
Index Cond: ((agent_number)::text = '+79780000002'::text) AND ((virtual_number)::text = '+79260000001'::text)
Filter: ((end_date >= '2025-11-01 00:00:00+03'::timestamp with time zone) AND (end_date < '2025-12-01 00:00:00+03'::timestamp with time zone))
```

Обновление данных



Обновление данных

- Если обновить значение ключаパーティционирования – то запись автоматически переедет в нужную партицию;

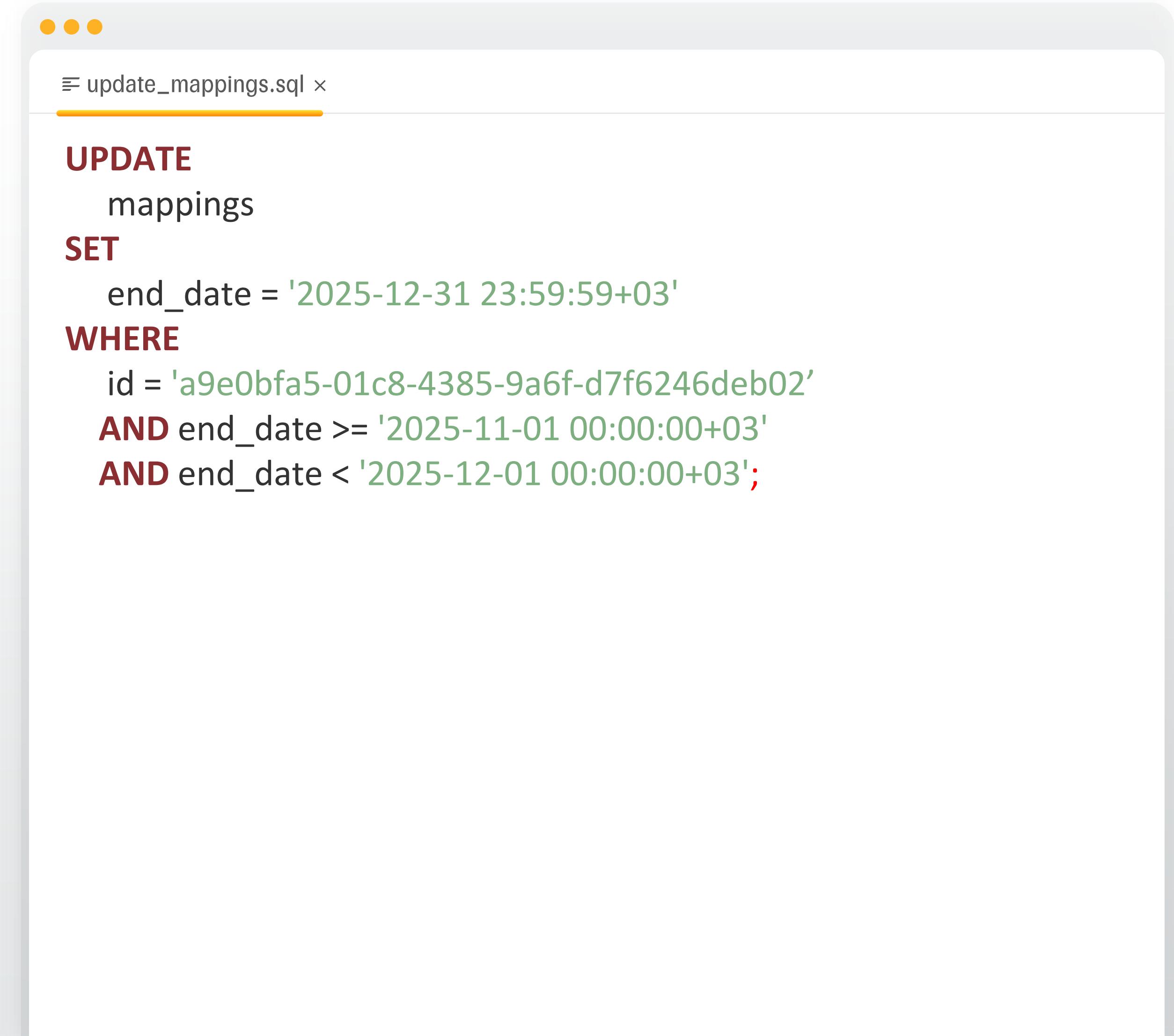


The screenshot shows a code editor window titled "update_mappings.sql". The code contains an UPDATE statement for the "mappings" table. It sets the "end_date" column to '2025-12-31 23:59:59+03' and specifies a WHERE clause where the "id" column is equal to 'a9e0bfa5-01c8-4385-9a6f-d7f6246deb02'. The code is color-coded, with keywords like UPDATE, SET, and WHERE in red, and the date/time and ID values in green.

```
UPDATE
    mappings
SET
    end_date = '2025-12-31 23:59:59+03'
WHERE
    id = 'a9e0bfa5-01c8-4385-9a6f-d7f6246deb02';
```

Обновление данных

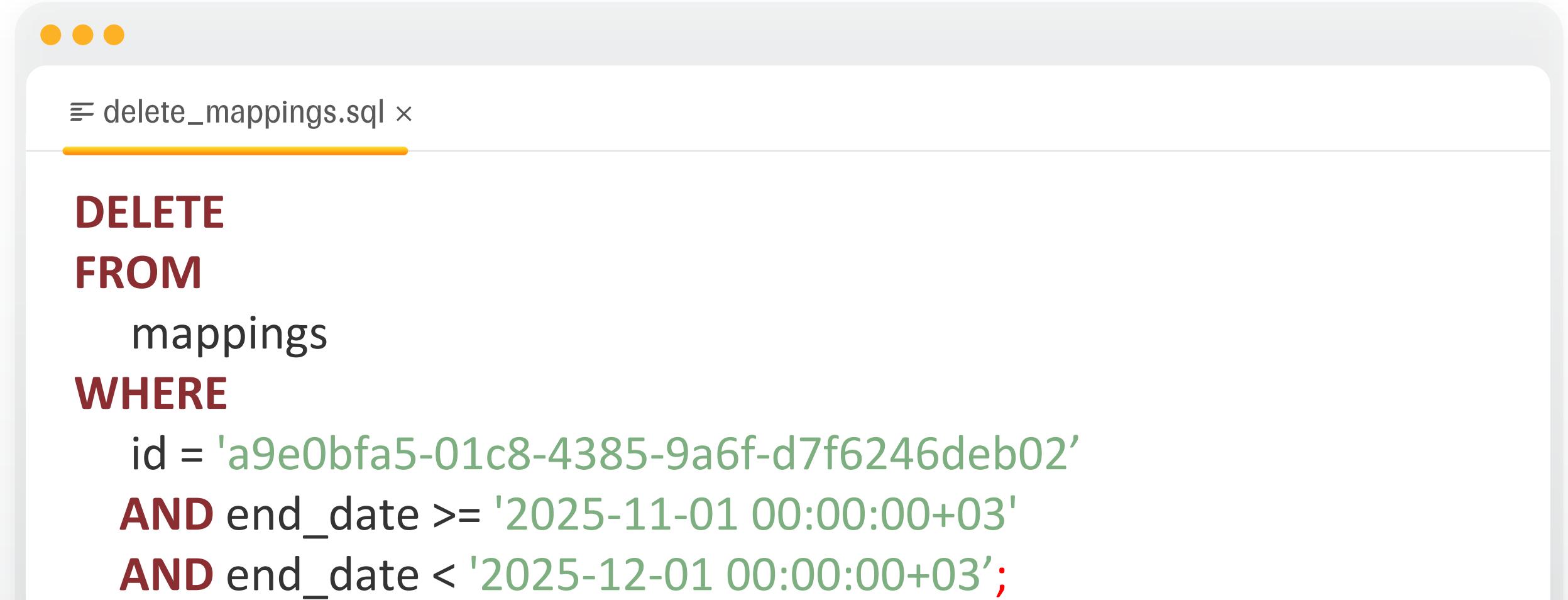
- Если обновить значение ключаパーティционирования – то запись автоматически переедет в нужную партицию;
- И также лучше конкретизировать в какой партиции стоит искать запись.



The screenshot shows a code editor window titled "update_mappings.sql". The code is an UPDATE statement targeting the "mappings" table. It sets the "end_date" column to '2025-12-31 23:59:59+03'. The WHERE clause specifies that the update should only affect rows where the "id" is 'a9e0bfa5-01c8-4385-9a6f-d7f6246deb02' and the "end_date" is between '2025-11-01 00:00:00+03' and '2025-12-01 00:00:00+03'.

```
UPDATE
    mappings
SET
    end_date = '2025-12-31 23:59:59+03'
WHERE
    id = 'a9e0bfa5-01c8-4385-9a6f-d7f6246deb02'
    AND end_date >= '2025-11-01 00:00:00+03'
    AND end_date < '2025-12-01 00:00:00+03';
```

Удаление данных



The image shows a screenshot of a code editor window titled "delete_mappings.sql". The code is a SQL DELETE statement targeting the "mappings" table, where the "id" column matches a specific UUID and the "end_date" column falls within a specified date range.

```
DELETE
FROM
    mappings
WHERE
    id = 'a9e0bfa5-01c8-4385-9a6f-d7f6246deb02'
    AND end_date >= '2025-11-01 00:00:00+03'
    AND end_date < '2025-12-01 00:00:00+03';
```



**Партицированная
таблица – роутер к
данным.**

Что важно запомнить

01

Партицированная таблица – шаблон для
партиций.

Что важно запомнить

01

Партицированная таблица – шаблон для партиций.

02

Партицированная таблица – роутер к данным.

Что важно запомнить

01 Партицированная таблица – шаблон для партиций.

02 Партицированная таблица – роутер к данным.

03 Синтаксически запросы к партицированной таблице такие же.

Что важно запомнить

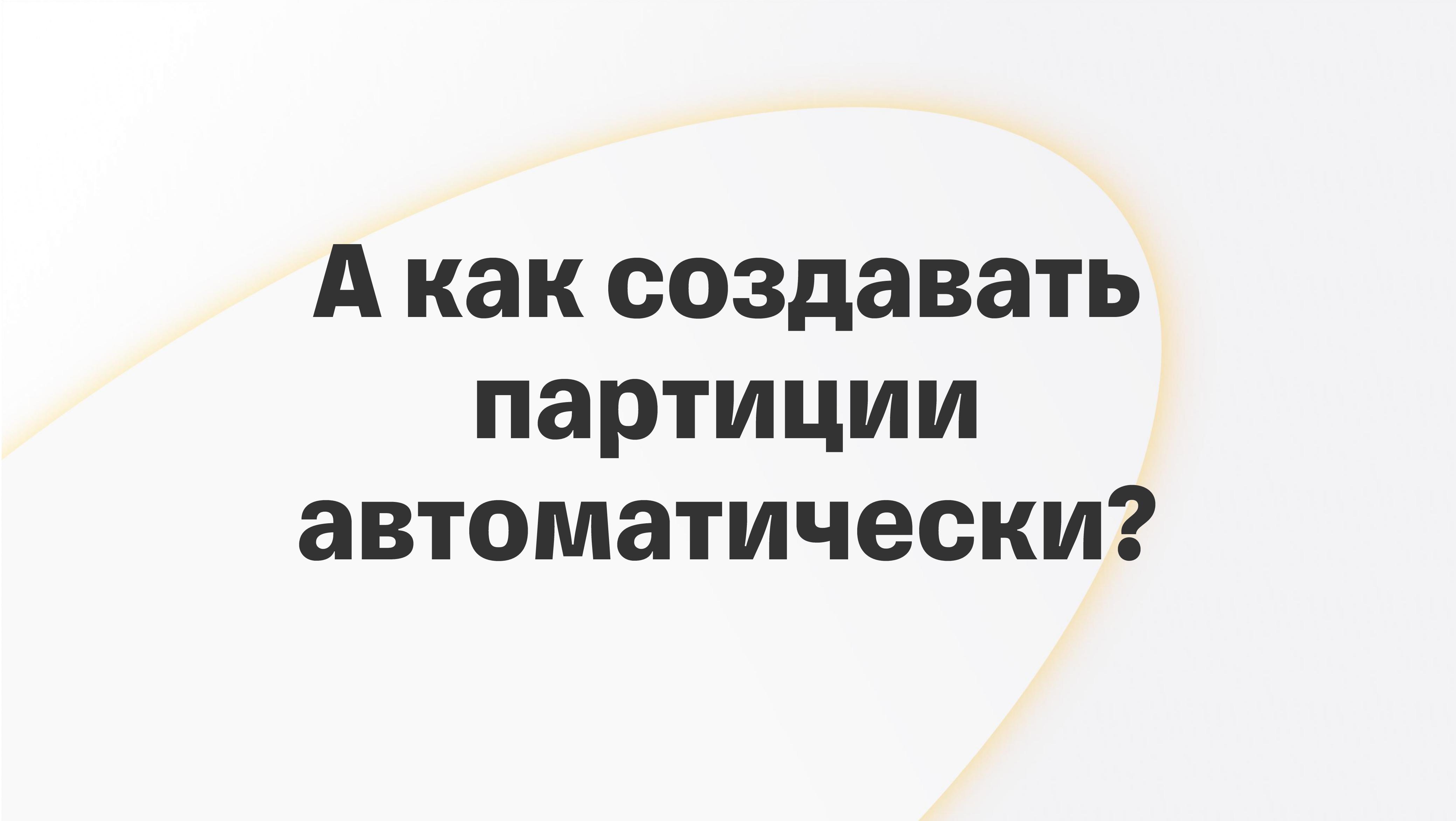
- 01** Партицированная таблица – шаблон для партиций.
- 02** Партицированная таблица – роутер к данным.
- 03** Синтаксически запросы к партицированной таблице такие же.
- 04** РК и уникальные индексы требуют включения в них ключаpartitionирования.

Что важно запомнить

- 01** Партицированная таблица – шаблон для партиций.
- 02** Партицированная таблица – роутер к данным.
- 03** Синтаксически запросы к партицированной таблице такие же.
- 04** РК и уникальные индексы требуют включения в них ключаpartitionирования.
- 05** Партиции могут иметь свои собственные индексы.

Что важно запомнить

- 01** Партицированная таблица – шаблон для партиций.
- 02** Партицированная таблица – роутер к данным.
- 03** Синтаксически запросы к партицированной таблице такие же.
- 04** РК и уникальные индексы требуют включения в них ключаpartitionирования.
- 05** Партиции могут иметь свои собственные индексы.
- 06** Думаем над тем какие партиции будут участвовать в запросе.



**А как создавать
партиции
автоматически?**

Варианты автоматического создания партиций



Средствами базы

Передать ответственность за
партиции на базу данных.



Логикой приложения

Передать ответственность за
партиции на приложение.

Автоматическое создание партиций базой данных



Триггером

Перед каждой вставкой
проверять, что нужная
партиция существует

Автоматическое создание партиций базой данных



Триггером

Перед каждой вставкой
проверять, что нужная
партиция существует



pg_partman

Интересно, но нет
возможности установить
данное расширение

Автоматическое создание партиций базой данных



Триггером

Перед каждой вставкой проверять, что нужная партиция существует



pg_partman

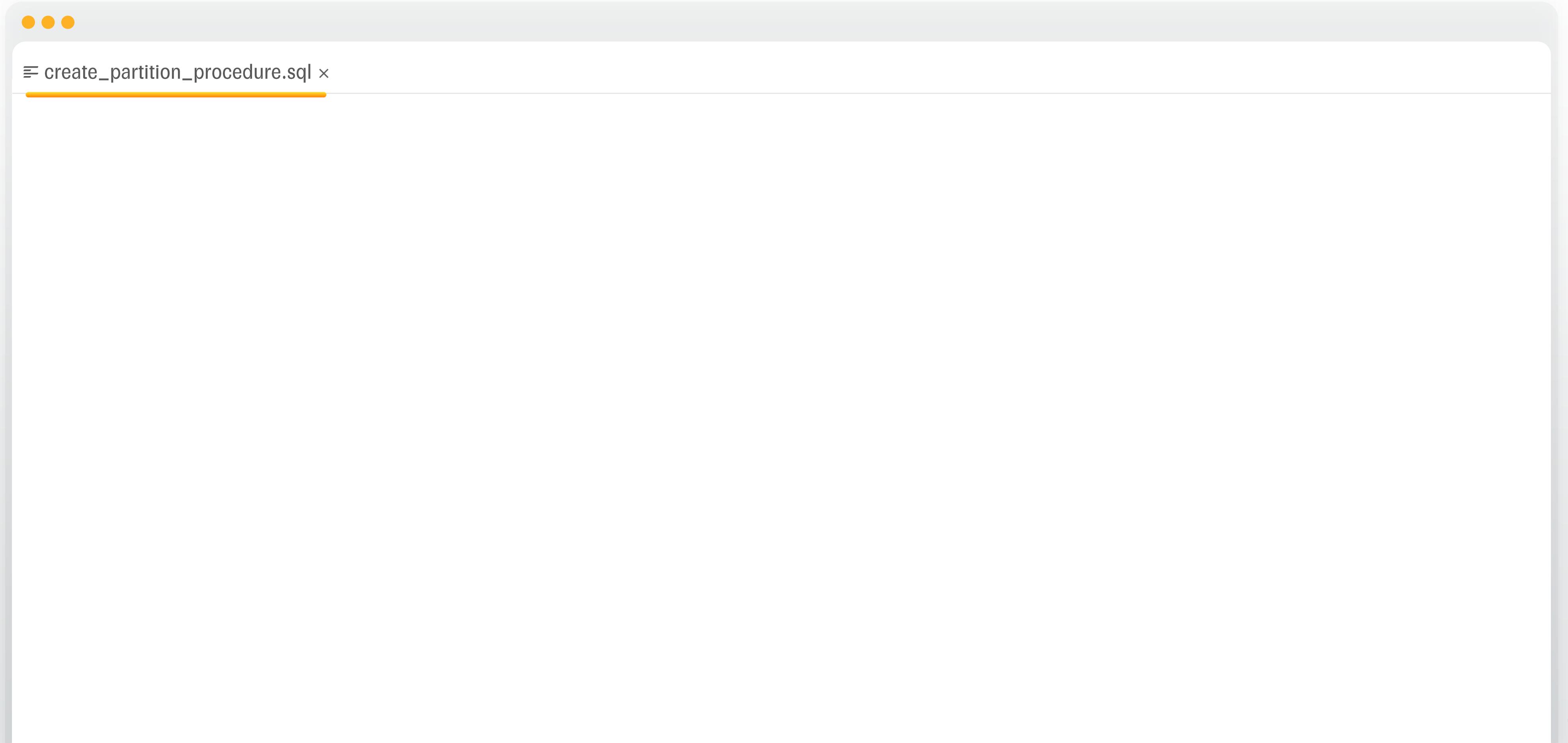
Интересно, но нет возможности установить данное расширение



pg_cron

Расширение можно установить. Нужно только написать процедуры.

Процедура создания партиции



A screenshot of a code editor window titled "create_partition_procedure.sql". The title bar has three yellow circular icons. The main area of the editor is empty, showing only the file name and a closing "x". A thin orange horizontal line is visible just below the title bar.

Процедура создания партиции

```
create_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции

```
create_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции

```
create_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
    partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
    full_partition_name TEXT := format('%1$s_%2$s', partition_name);
    is_partition_exists BOOLEAN := FALSE;
    end_date DATE := start_date + INTERVAL '1 month';
BEGIN
    is_partition_exists := EXISTS (
        SELECT FROM pg_tables
        WHERE tablename = partition_name AND schemaname = 'public'
    );
    IF is_partition_exists THEN
        RAISE NOTICE 'Partition % has already exist', full_partition_name;
        RETURN;
    END IF;
    EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции

```
create_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции



≡ create_partition_procedure.sql ×

```
CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции

```
create_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s PARTITION OF %2$s FOR VALUES FROM (%3$L) TO (%4$L)', full_partition_name, table_name, start_date, end_date);
END;
$body$
```

Процедура создания партиции



≡ create_partition_procedure.sql ×

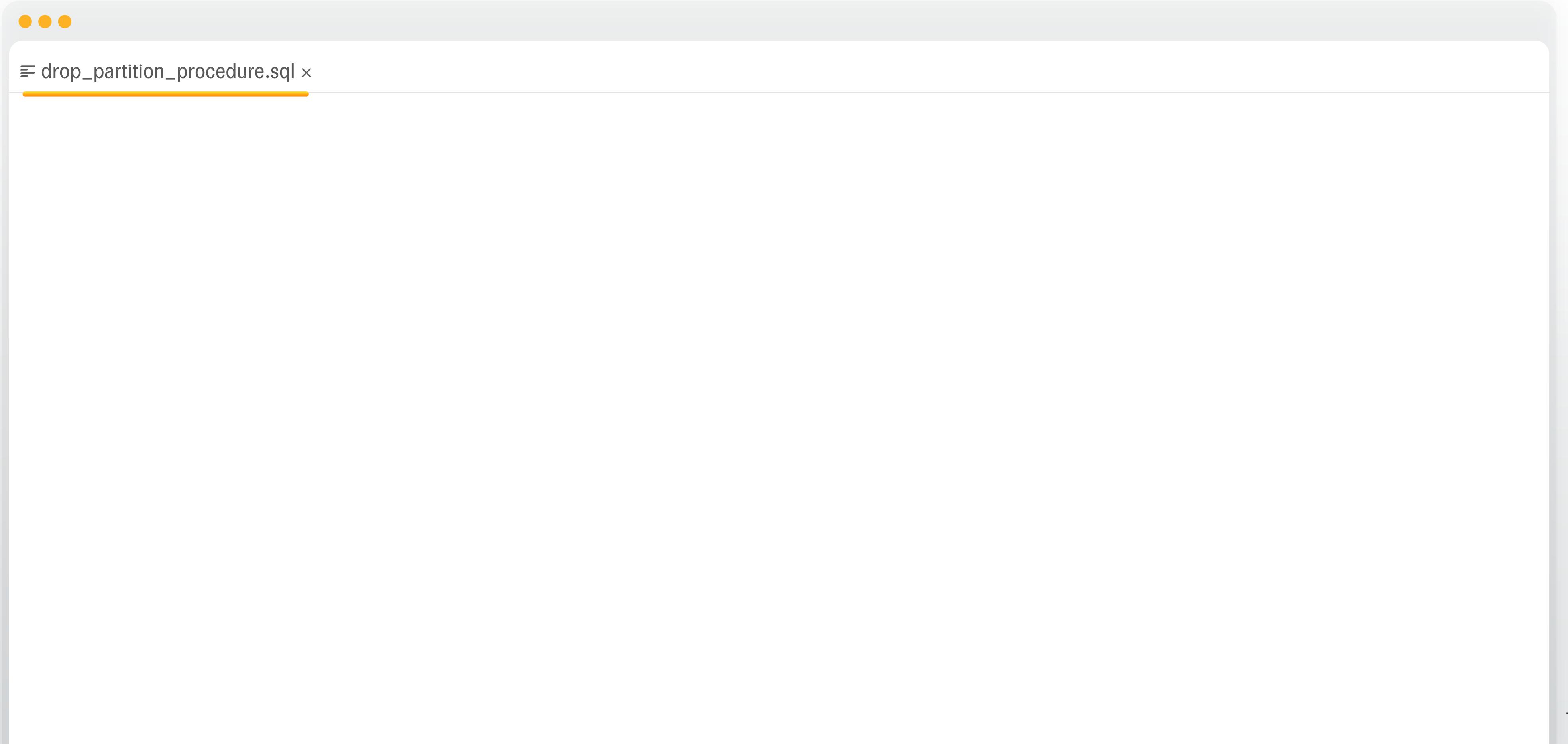
```
CREATE OR REPLACE PROCEDURE create_mapping_partition(start_date DATE)
LANGUAGE plpgsql
AS $body$
DECLARE
partition_name TEXT := format('mappings_%1$s', TO_CHAR(start_date, 'YYYY_MM'));
full_partition_name TEXT := format('%1$s', partition_name);
is_partition_exists BOOLEAN := FALSE;
end_date DATE := start_date + INTERVAL '1 month';
BEGIN
is_partition_exists := EXISTS (
    SELECT FROM pg_tables
    WHERE tablename = partition_name AND schemaname = 'public'
);

IF is_partition_exists THEN
    RAISE NOTICE 'Partition % has already exist', full_partition_name;
    RETURN;
END IF;

EXECUTE format('CREATE TABLE %1$s (LIKE %2$s INCLUDING DEFAULTS INCLUDING CONSTRAINTS)', full_partition_name, 'mappings');
EXECUTE format('ALTER TABLE %1$s ADD CONSTRAINT end_date_check CHECK (end_date >= (%2$L) AND end_date < (%3$L))', full_partition_name, start_date, end_date);
EXECUTE format('INSERT INTO %1$s SELECT * FROM %2$s WHERE end_date >= (%3$L) AND end_date < (%4$L)', full_partition_name, 'mappings_default', start_date, end_date);
EXECUTE format('DELETE FROM %1$s WHERE end_date >= (%2$L) AND end_date < (%3$L)', 'mappings_default', start_date, end_date);
EXECUTE format('ALTER TABLE %1$s ATTACH PARTITION %2$s FOR VALUES FROM (%3$L) TO (%4$L)', 'mappings', full_partition_name, start_date, end_date);
EXECUTE format('ALTER TABLE %1$s DROP CONSTRAINT end_date_check', full_partition_name);

END,
$body$
```

Процедура удаления партиции



The image shows a screenshot of a code editor window. The title bar at the top says "drop_partition_procedure.sql x". Below the title bar is a toolbar with three yellow circular icons. The main area of the editor is completely blank, indicating that the file is currently empty.

Процедура удаления партиции

```
drop_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE drop_mapping_partitions(older_than INTERVAL)
LANGUAGE plpgsql
AS $body$
DECLARE
    min_date DATE := DATE_TRUNC('month', CURRENT_DATE - older_than)::DATE;
    partition_name VARCHAR(64);
    partition_date DATE;
    temprow RECORD;
BEGIN
    FOR temprow IN
        SELECT nsp_child.nspname || '.' || child.relname AS partition_name
        FROM pg_inherits
        JOIN pg_class parent ON pg_inherits.inhparent = parent.oid
        JOIN pg_class child ON pg_inherits.inherid = child.oid
        JOIN pg_namespace nsp_parent ON nsp_parent.oid = parent.relnamespace
        JOIN pg_namespace nsp_child ON nsp_child.oid = child.relnamespace
        WHERE parent.relname = 'mappings' AND nsp_parent.nspname = 'public' AND nsp_child.nspname = 'public' AND child.relname != 'mappings_default'
        ORDER BY partition_name
    LOOP
        partition_name := temprow.partition_name;
        partition_date := MAKE_DATE(SPLIT_PART(partition_name, '_', 2)::INT, SPLIT_PART(partition_name, '_', 3)::INT, 1);
        IF partition_date <= min_date THEN
            EXECUTE FORMAT('DROP TABLE %1$s', partition_name);
            RAISE NOTICE 'Partition deleted %', partition_name;
        END IF;
    END LOOP;
END;
$body$
```

Процедура удаления партиции

```
drop_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE drop_mapping_partitions(older_than INTERVAL)
LANGUAGE plpgsql
AS $body$
DECLARE
min_date DATE := DATE_TRUNC('month', CURRENT_DATE - older_than)::DATE;
partition_name VARCHAR(64);
partition_date DATE;
temprow RECORD;
BEGIN
FOR temprow IN
SELECT nsp_child.nspname || '.' || child.relname AS partition_name
FROM pg_inherits
JOIN pg_class parent ON pg_inherits.inhparent = parent.oid
JOIN pg_class child ON pg_inherits.inherid = child.oid
JOIN pg_namespace nsp_parent ON nsp_parent.oid = parent.relnamespace
JOIN pg_namespace nsp_child ON nsp_child.oid = child.relnamespace
WHERE parent.relname = 'mappings' AND nsp_parent.nspname = 'public' AND nsp_child.nspname = 'public' AND child.relname != 'mappings_default'
ORDER BY partition_name
LOOP
partition_name := temprow.partition_name;
partition_date := MAKE_DATE(SPLIT_PART(partition_name, '_', 2)::INT, SPLIT_PART(partition_name, '_', 3)::INT, 1);
IF partition_date <= min_date THEN
EXECUTE FORMAT('DROP TABLE %1$s', partition_name);
RAISE NOTICE 'Partition deleted %', partition_name;
END IF;
END LOOP;
END;
$body$
```

Процедура удаления партиции

```
drop_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE drop_mapping_partitions(older_than INTERVAL)
LANGUAGE plpgsql
AS $body$  
DECLARE
    min_date DATE := DATE_TRUNC('month', CURRENT_DATE - older_than)::DATE;
    partition_name VARCHAR(64);
    partition_date DATE;
    temprow RECORD;
BEGIN
    FOR temprow IN
        SELECT nsp_child.nspname || '.' || child.relname AS partition_name
        FROM pg_inherits
        JOIN pg_class parent ON pg_inherits.inhparent = parent.oid
        JOIN pg_class child ON pg_inherits.inherid = child.oid
        JOIN pg_namespace nsp_parent ON nsp_parent.oid = parent.relnamespace
        JOIN pg_namespace nsp_child ON nsp_child.oid = child.relnamespace
        WHERE parent.relname = 'mappings' AND nsp_parent.nspname = 'public' AND nsp_child.nspname = 'public' AND child.relname != 'mappings_default'
        ORDER BY partition_name
    LOOP
        partition_name := temprow.partition_name;
        partition_date := MAKE_DATE(SPLIT_PART(partition_name, '_', 2)::INT, SPLIT_PART(partition_name, '_', 3)::INT, 1);
        IF partition_date <= min_date THEN
            EXECUTE FORMAT('DROP TABLE %1$s', partition_name);
            RAISE NOTICE 'Partition deleted %', partition_name;
        END IF;
    END LOOP;
END;
$body$
```

Процедура удаления партиции

```
drop_partition_procedure.sql ×

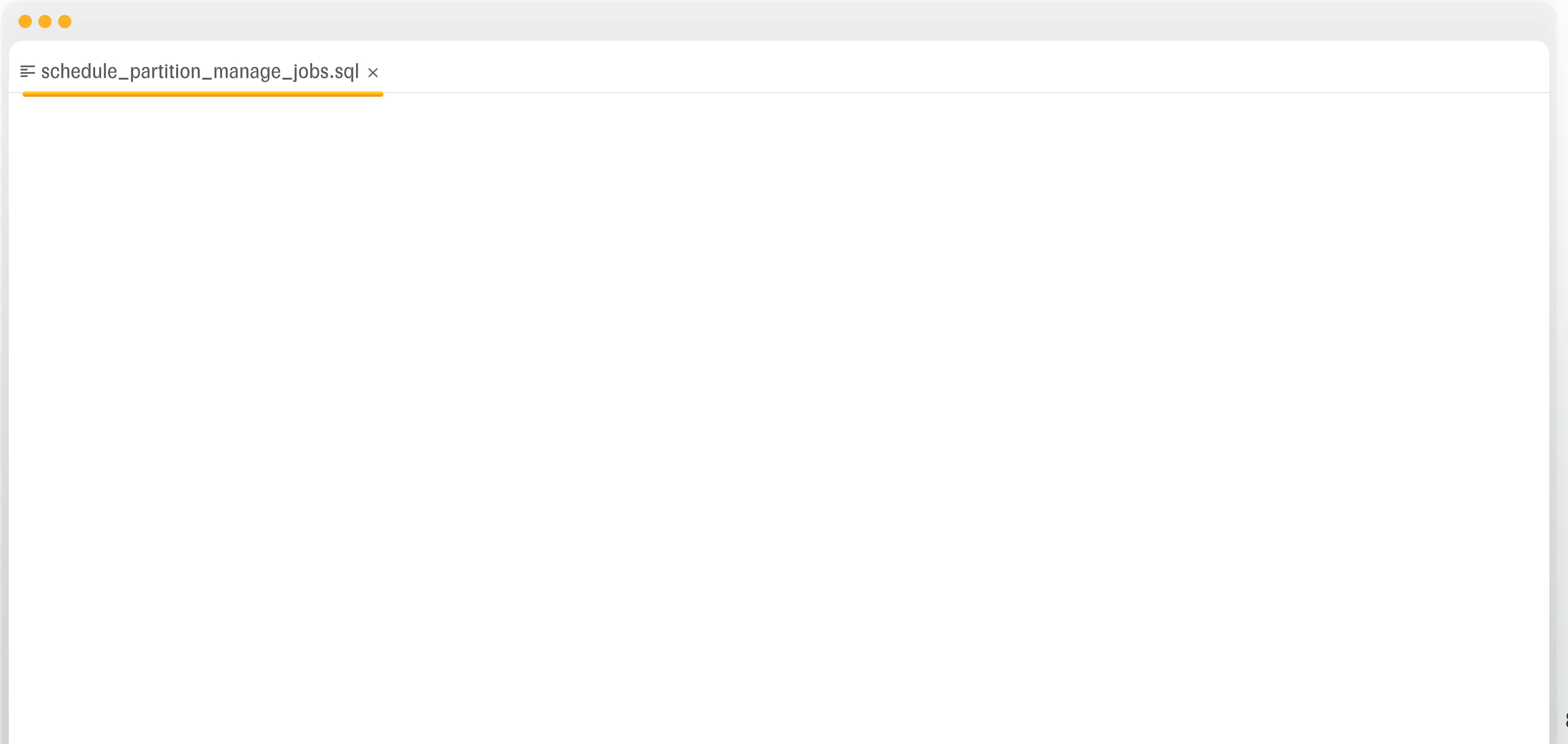
CREATE OR REPLACE PROCEDURE drop_mapping_partitions(older_than INTERVAL)
LANGUAGE plpgsql
AS $body$
DECLARE
min_date DATE := DATE_TRUNC('month', CURRENT_DATE - older_than)::DATE;
partition_name VARCHAR(64);
partition_date DATE;
temprow RECORD;
BEGIN
FOR temprow IN
SELECT nsp_child.nspname || '.' || child.relname AS partition_name
FROM pg_inherits
JOIN pg_class parent ON pg_inherits.inhparent = parent.oid
JOIN pg_class child ON pg_inherits.inherid = child.oid
JOIN pg_namespace nsp_parent ON nsp_parent.oid = parent.relnamespace
JOIN pg_namespace nsp_child ON nsp_child.oid = child.relnamespace
WHERE parent.relname = 'mappings' AND nsp_parent.nspname = 'public' AND nsp_child.nspname = 'public' AND child.relname != 'mappings_default'
ORDER BY partition_name
LOOP
partition_name := temprow.partition_name;
partition_date := MAKE_DATE(SPLIT_PART(partition_name, '_', 2)::INT, SPLIT_PART(partition_name, '_', 3)::INT, 1);
IF partition_date <= min_date THEN
EXECUTE FORMAT('DROP TABLE %1$s', partition_name);
RAISE NOTICE 'Partition deleted %', partition_name;
END IF;
END LOOP;
END;
$body$
```

Процедура удаления партиции

```
drop_partition_procedure.sql ×

CREATE OR REPLACE PROCEDURE drop_mapping_partitions(older_than INTERVAL)
LANGUAGE plpgsql
AS $body$
DECLARE
    min_date DATE := DATE_TRUNC('month', CURRENT_DATE - older_than)::DATE;
    partition_name VARCHAR(64);
    partition_date DATE;
    temprow RECORD;
BEGIN
    FOR temprow IN
        SELECT nsp_child.nspname || '.' || child.relname AS partition_name
        FROM pg_inherits
        JOIN pg_class parent ON pg_inherits.inhparent = parent.oid
        JOIN pg_class child ON pg_inherits.inherid = child.oid
        JOIN pg_namespace nsp_parent ON nsp_parent.oid = parent.relnamespace
        JOIN pg_namespace nsp_child ON nsp_child.oid = child.relnamespace
        WHERE parent.relname = 'mappings' AND nsp_parent.nspname = 'public' AND nsp_child.nspname = 'public' AND child.relname != 'mappings_default'
        ORDER BY partition_name
    LOOP
        partition_name := temprow.partition_name;
        partition_date := MAKE_DATE(SPLIT_PART(partition_name, '_', 2)::INT, SPLIT_PART(partition_name, '_', 3)::INT, 1);
        IF partition_date <= min_date THEN
            EXECUTE FORMAT('DROP TABLE %1$s', partition_name);
            RAISE NOTICE 'Partition deleted %', partition_name;
        END IF;
    END LOOP;
END;
$body$
```

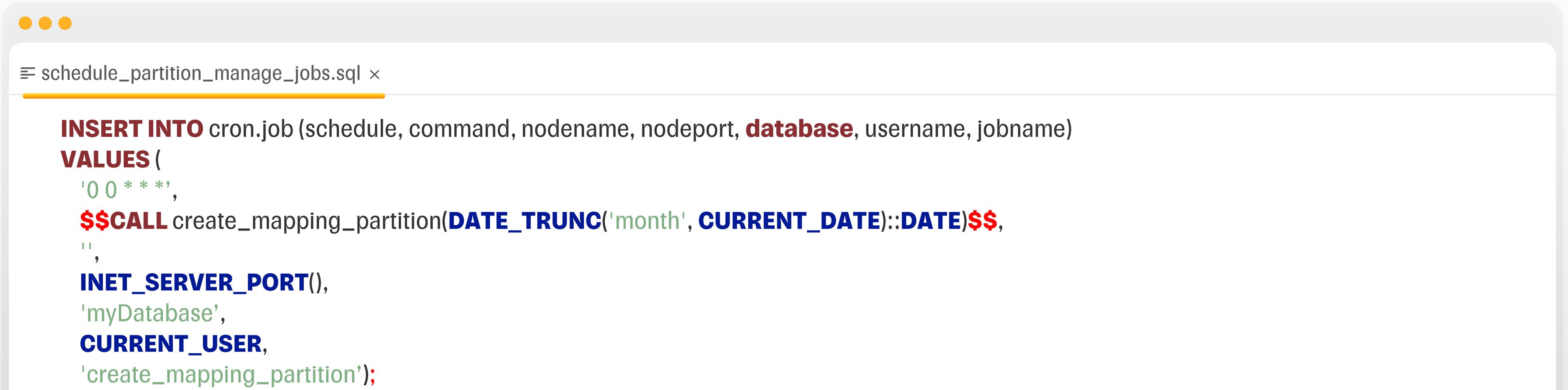
Планирование запуска процедур



The image shows a screenshot of a code editor window. The title bar at the top has three yellow circular icons. The main area displays a file titled "schedule_partition_manage_jobs.sql" with a small "x" icon in the top right corner. A horizontal orange line highlights the beginning of the file's content area. Below the title bar, there is a thin grey border.

```
☰ schedule_partition_manage_jobs.sql ×
```

Планирование запуска процедур

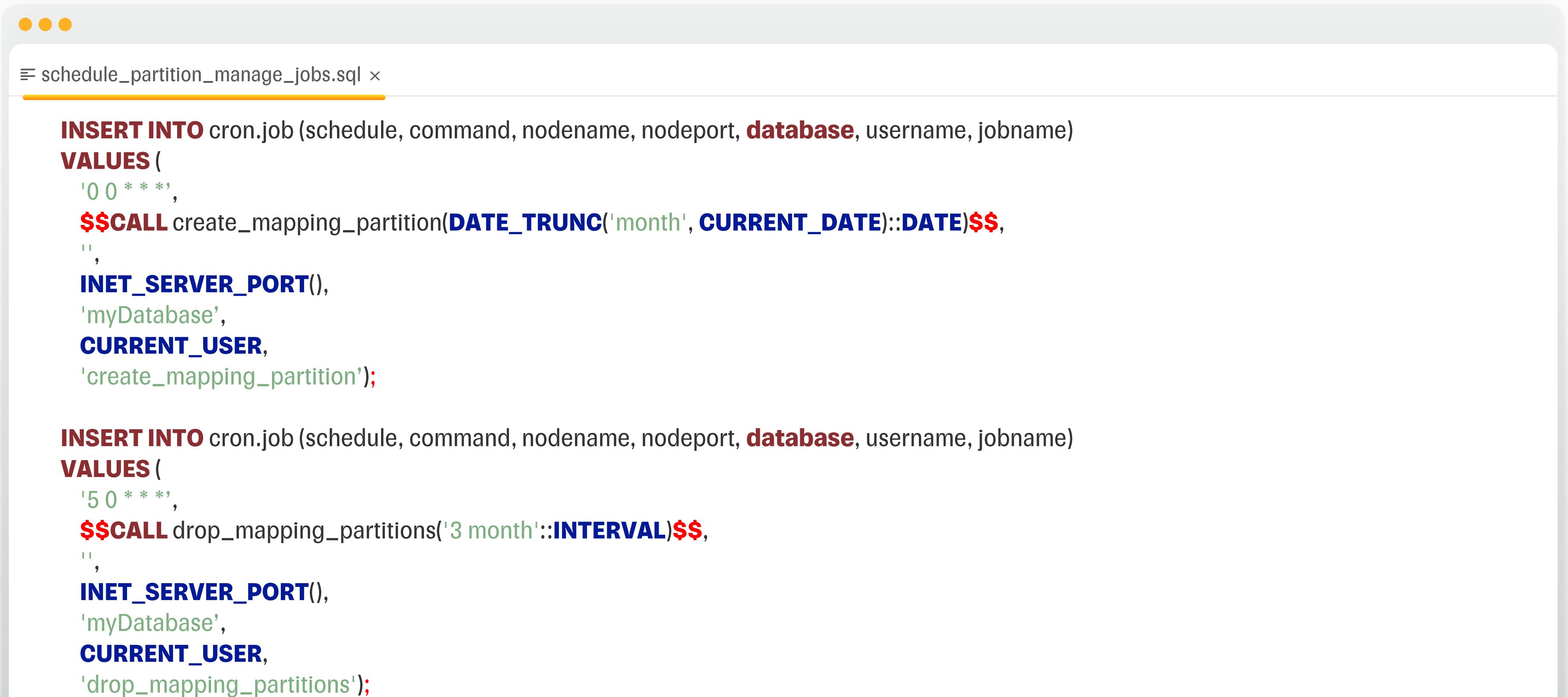


The screenshot shows a code editor window with a tab labeled "schedule_partition_manage_jobs.sql". The code itself is an SQL INSERT statement:

```
INSERT INTO cron.job (schedule, command, nodename, nodeport, database, username, jobname)
VALUES (
    '0 0 * * *',
    $$CALL create_mapping_partition(DATE_TRUNC('month', CURRENT_DATE)::DATE$$,
    '',
    INET_SERVER_PORT(),
    'myDatabase',
    CURRENT_USER,
    'create_mapping_partition');
```

The code uses several PostgreSQL-specific functions and variables, such as `DATE_TRUNC`, `CURRENT_DATE`, and `CURRENT_USER`. The editor's syntax highlighting colors the code elements: red for `INSERT INTO`, `VALUES`, and `$$CALL`; blue for `DATE_TRUNC`, `CURRENT_DATE`, and `CURRENT_USER`; green for strings like `'0 0 * * *'` and `'myDatabase'`; and black for identifiers like `cron.job`, `command`, etc.

Планирование запуска процедур

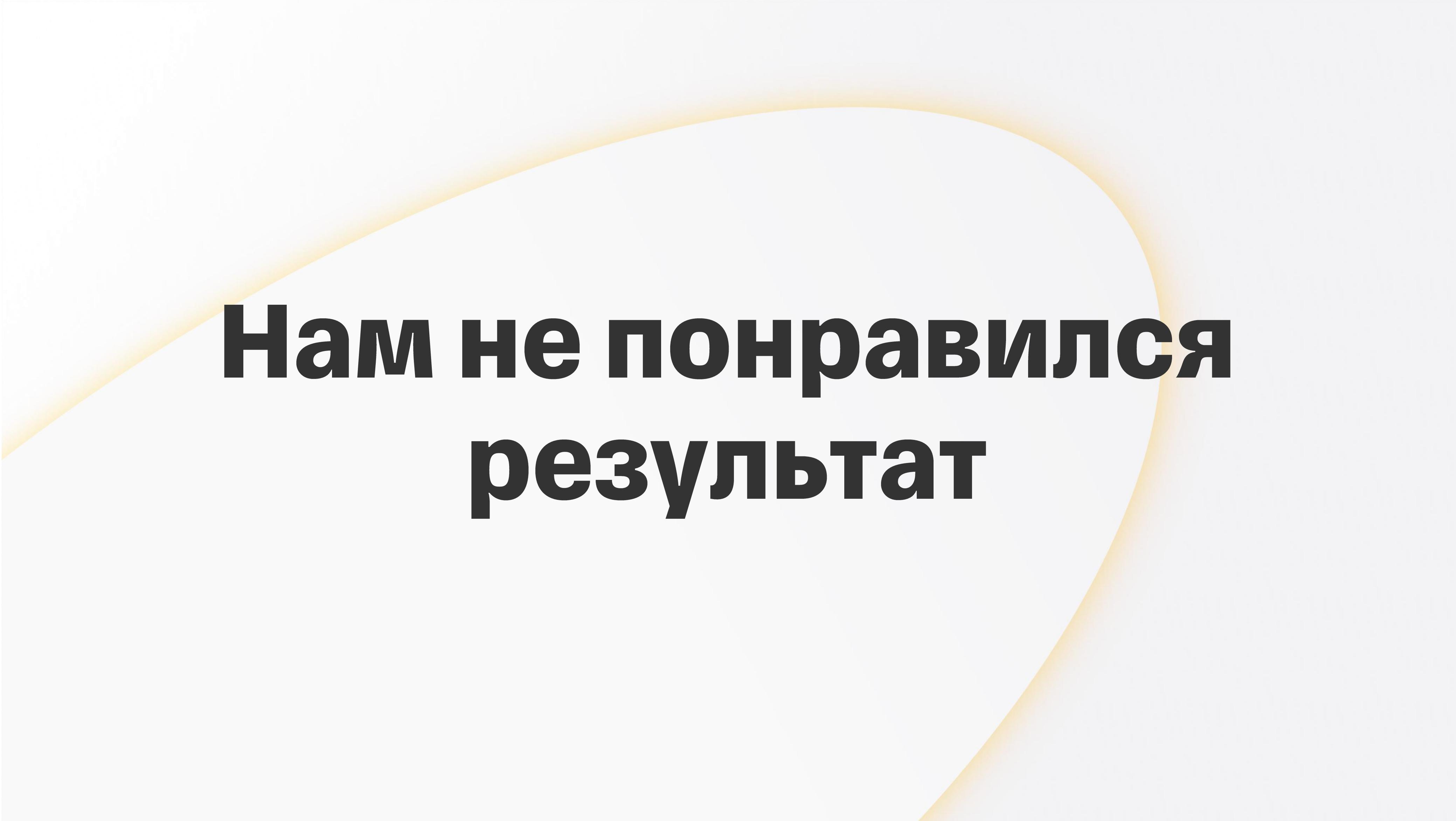


The screenshot shows a code editor window with two SQL scripts. The top script is titled 'schedule_partition_manage_jobs.sql' and contains code to insert a new job into the 'cron.job' table. The bottom script contains a similar structure for another job.

```
schedule_partition_manage_jobs.sql ×

INSERT INTO cron.job (schedule, command, nodename, nodeport, database, username, jobname)
VALUES (
    '0 0 * * *',
    $$CALL create_mapping_partition(DATE_TRUNC('month', CURRENT_DATE)::DATE$$,
    '',
    INET_SERVER_PORT(),
    'myDatabase',
    CURRENT_USER,
    'create_mapping_partition');

INSERT INTO cron.job (schedule, command, nodename, nodeport, database, username, jobname)
VALUES (
    '5 0 * * *',
    $$CALL drop_mapping_partitions('3 month'::INTERVAL$$,
    '',
    INET_SERVER_PORT(),
    'myDatabase',
    CURRENT_USER,
    'drop_mapping_partitions');
```



**Нам не понравился
результат**

Проблемы

- ➡ Процедуры сложно переиспользовать в других сервисах.

Проблемы

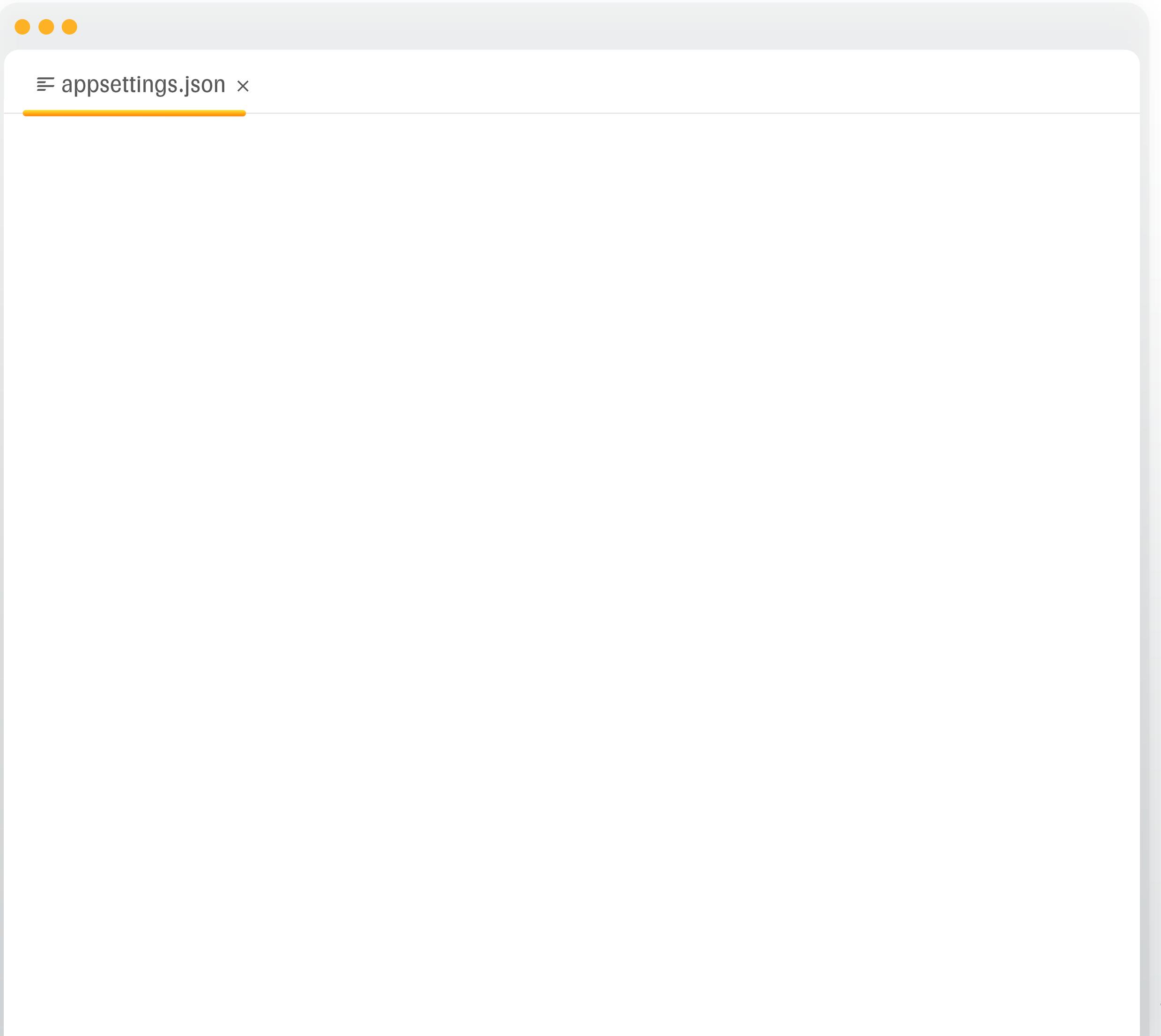
- ➔ Процедуры сложно переиспользовать в других сервисах.
- ➔ pg_cron устанавливается только в одну базу данных на одном сервере PostgreSQL.

Проблемы

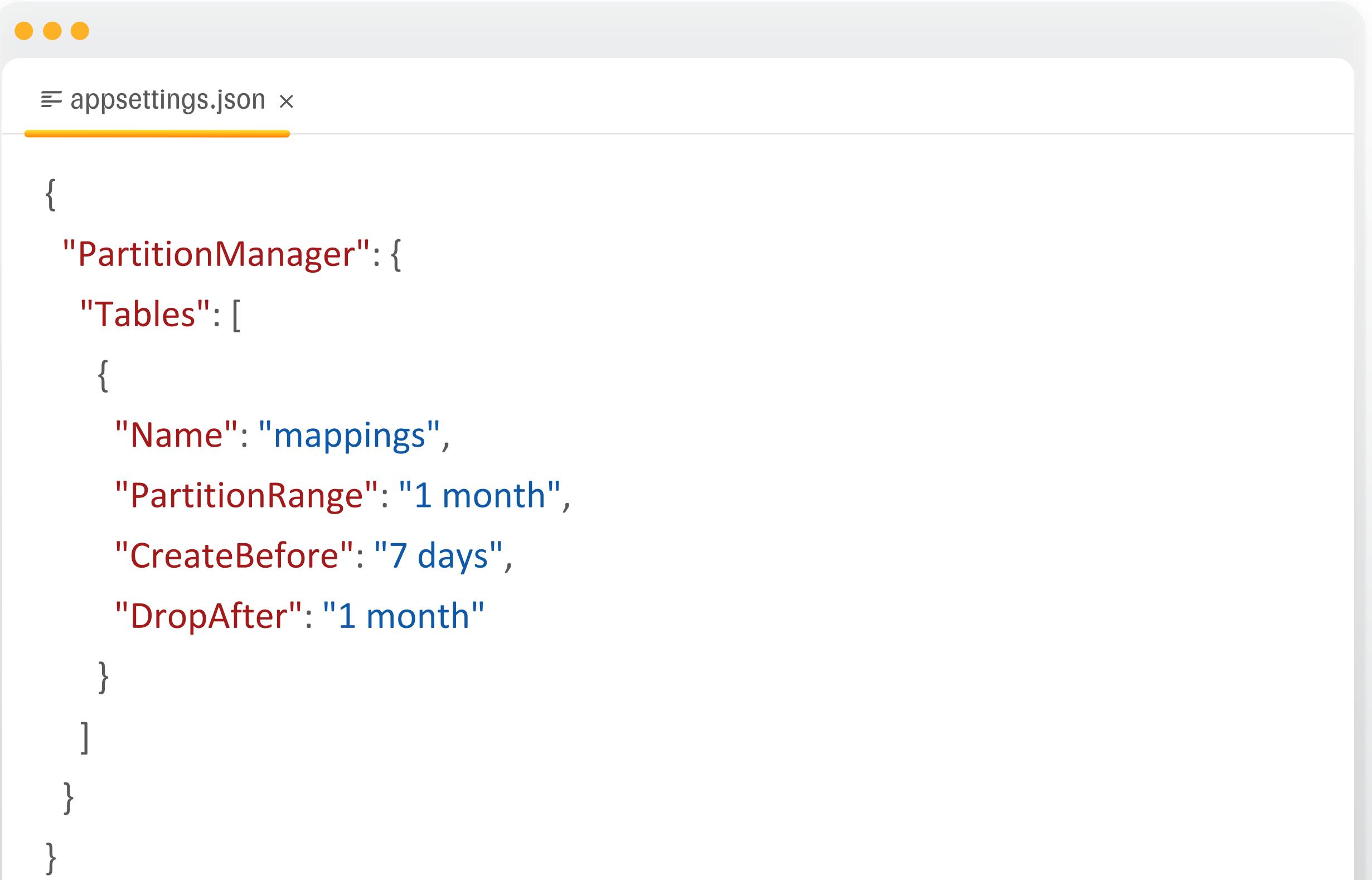
- ➔ Процедуры сложно переиспользовать в других сервисах.
- ➔ pg_cron устанавливается только в одну базу данных на одном сервере PostgreSQL.
- ➔ Отсутствует мониторинг. Если партиция не создалась, то об этом узнаем не сразу.

Создаем PartitionManager

Настройки менеджера партиций



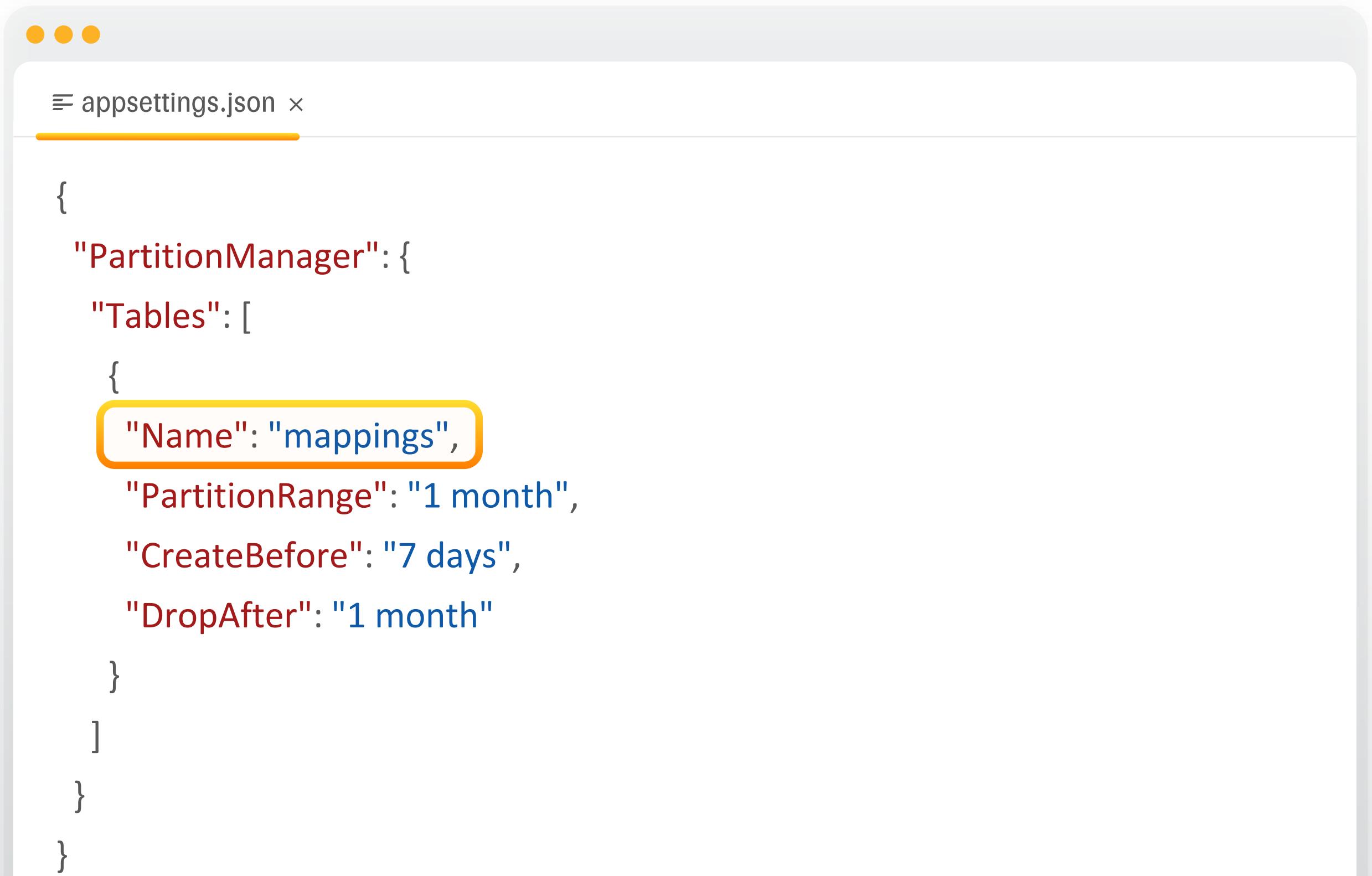
Настройки менеджераパーティций



```
appsettings.json x
{
  "PartitionManager": {
    "Tables": [
      {
        "Name": "mappings",
        "PartitionRange": "1 month",
        "CreateBefore": "7 days",
        "DropAfter": "1 month"
      }
    ]
  }
}
```

Настройки менеджераパーティций

- **Name** – название партиционированной таблицы;

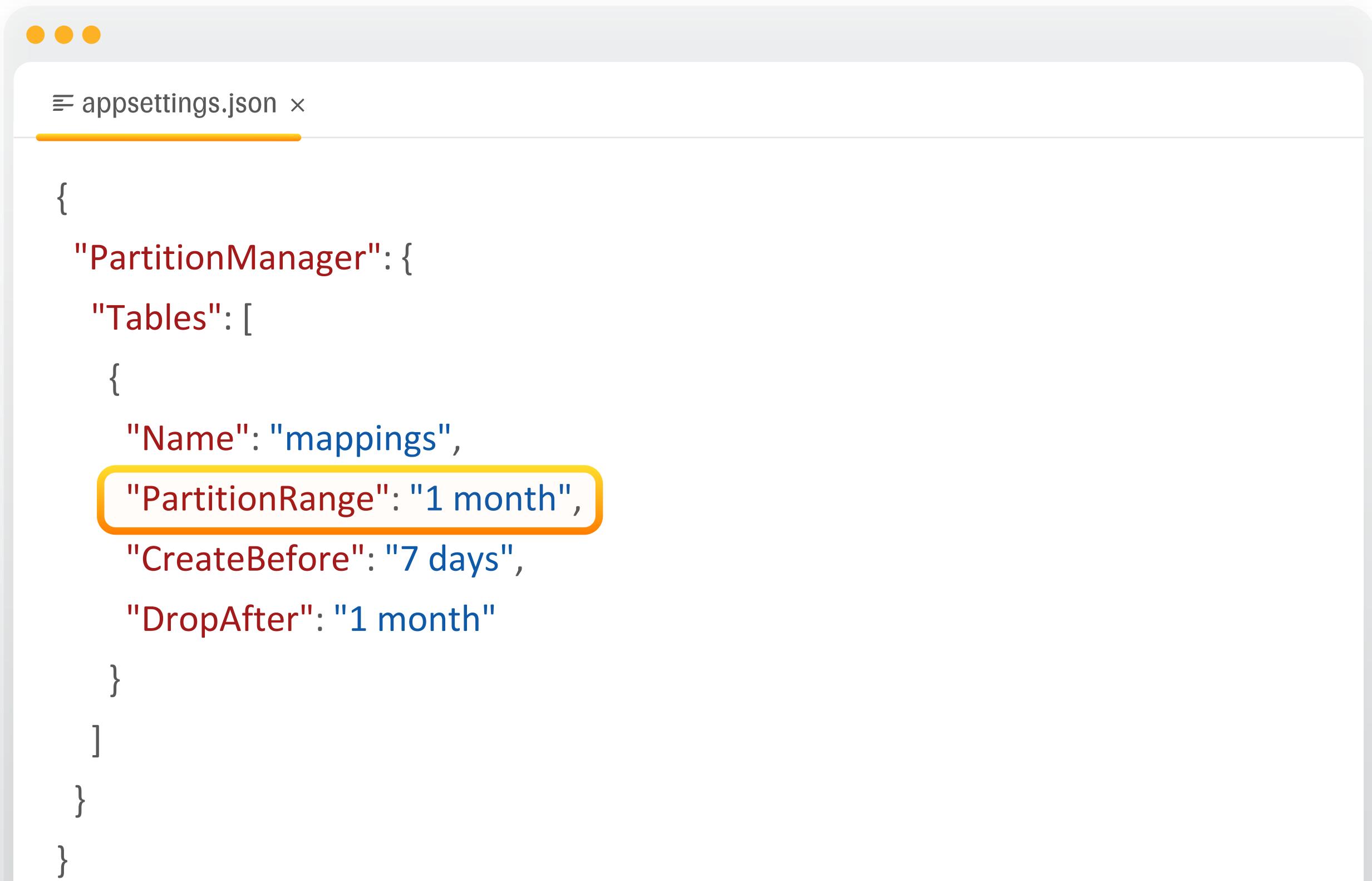


The image shows a screenshot of a code editor window titled "appsettings.json". The JSON configuration defines a "PartitionManager" for a table named "mappings". The configuration includes settings for partition ranges, creation before drop, and drop after. The "Name" field is highlighted with a yellow border.

```
{  
  "PartitionManager": {  
    "Tables": [  
      {  
        "Name": "mappings",  
        "PartitionRange": "1 month",  
        "CreateBefore": "7 days",  
        "DropAfter": "1 month"  
      }  
    ]  
  }  
}
```

Настройки менеджераパーティций

- **Name** – название партиционированной таблицы;
- **PartitionRange** – размер партиции;



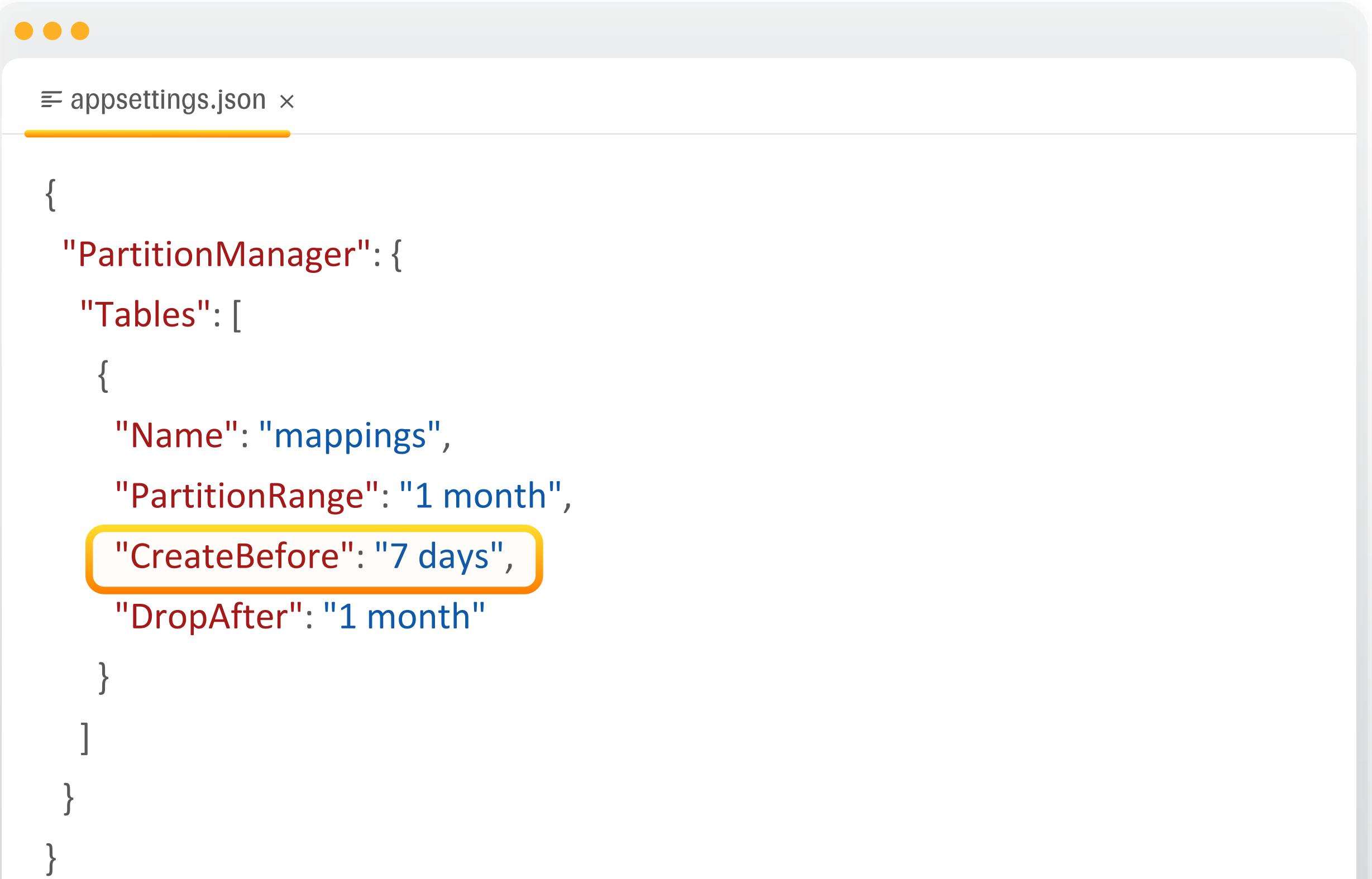
The image shows a screenshot of a code editor window titled "appsettings.json x". The JSON configuration defines a "PartitionManager" for a table named "mappings". The "PartitionRange" is set to "1 month", which is highlighted with a yellow-orange rounded rectangle. Other settings include "CreateBefore": "7 days" and "DropAfter": "1 month". The code is structured with curly braces and square brackets.

```
appsettings.json x

{
  "PartitionManager": {
    "Tables": [
      {
        "Name": "mappings",
        "PartitionRange": "1 month",
        "CreateBefore": "7 days",
        "DropAfter": "1 month"
      }
    ]
  }
}
```

Настройки менеджераパーティций

- **Name** – название партиционированной таблицы;
- **PartitionRange** – размер партиции;
- **CreateBefore** – за сколько дней **до** нужно создать партицию;



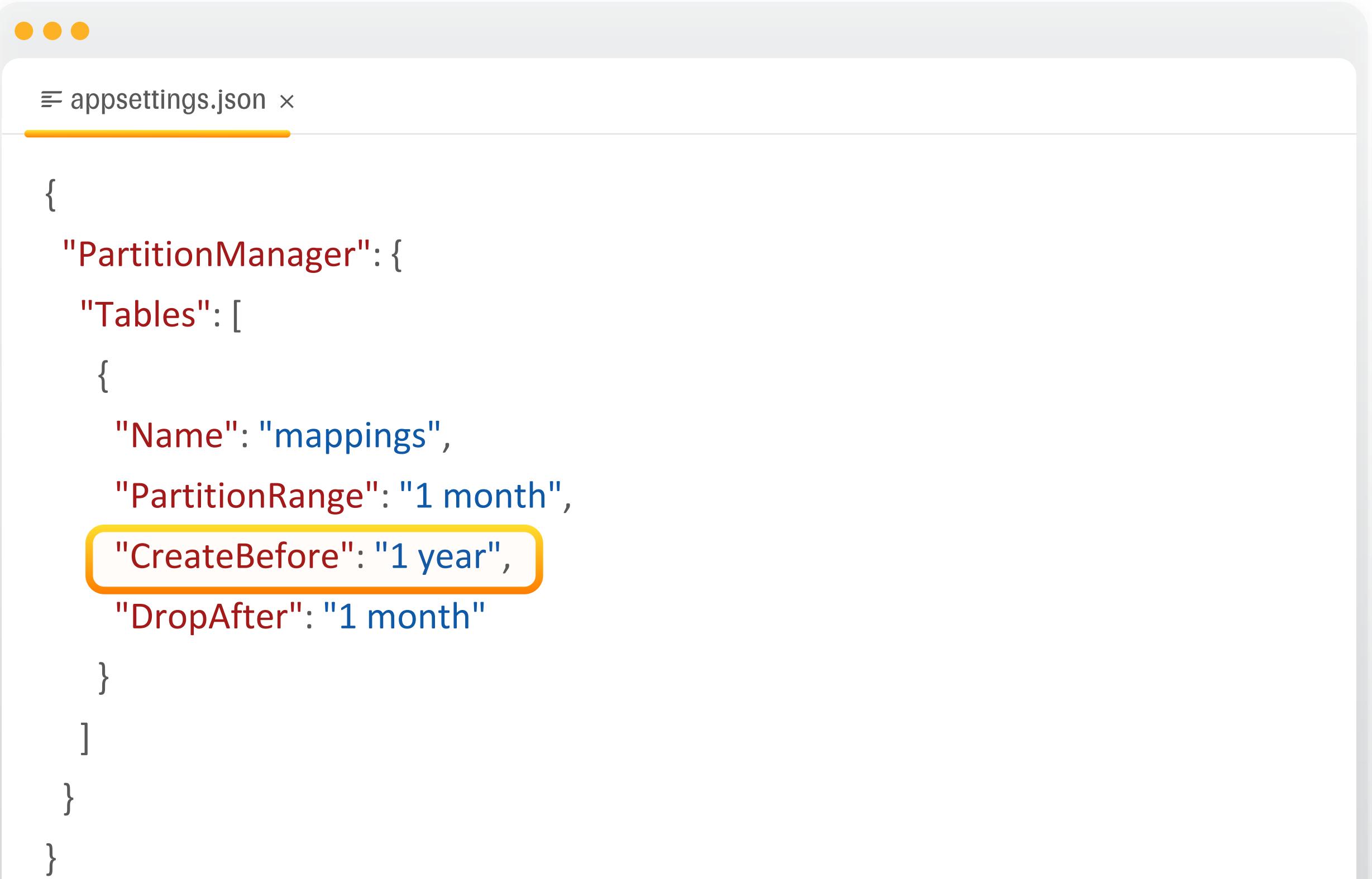
```
appsettings.json
```

```
{  
    "PartitionManager": {  
        "Tables": [  
            {  
                "Name": "mappings",  
                "PartitionRange": "1 month",  
                "CreateBefore": "7 days",  
                "DropAfter": "1 month"  
            }  
        ]  
    }  
}
```

The screenshot shows a code editor window with the file name 'appsettings.json' at the top. The JSON configuration for the 'PartitionManager' is displayed. A yellow box highlights the 'CreateBefore' field, which is set to '7 days'. This field corresponds to the 'CreateBefore' parameter mentioned in the list of requirements.

Настройки менеджераパーティций

- **Name** – название партиционированной таблицы;
- **PartitionRange** – размер партиции;
- **CreateBefore** – за сколько дней **до** нужно создать партицию;

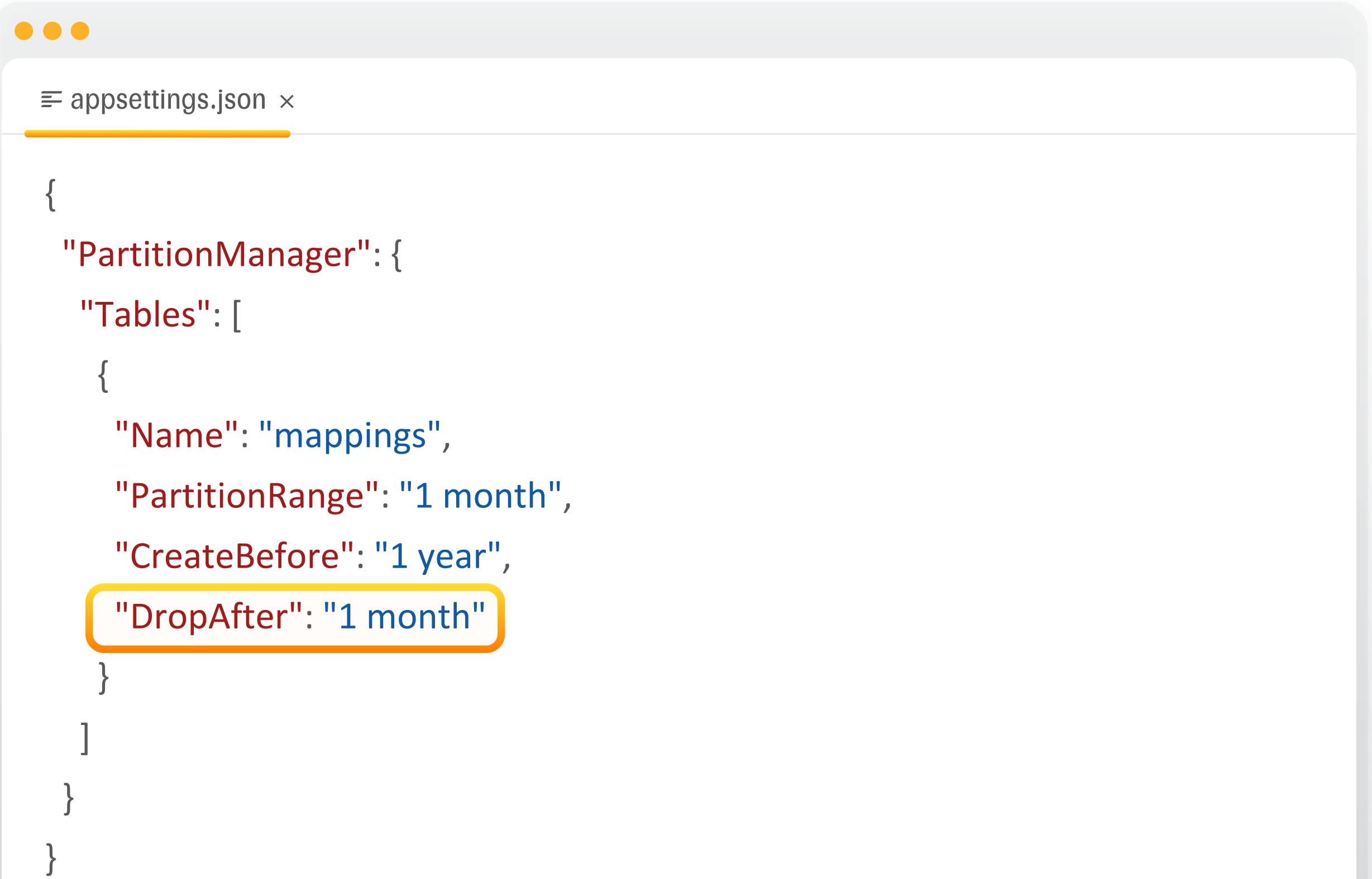


The image shows a screenshot of a code editor window titled "appsettings.json x". The JSON configuration defines a "PartitionManager" for a table named "mappings". The "CreateBefore" setting is highlighted with a yellow border, indicating it is the focus of the current discussion.

```
{  
  "PartitionManager": {  
    "Tables": [  
      {  
        "Name": "mappings",  
        "PartitionRange": "1 month",  
        "CreateBefore": "1 year",  
        "DropAfter": "1 month"  
      }  
    ]  
  }  
}
```

Настройки менеджераパーティций

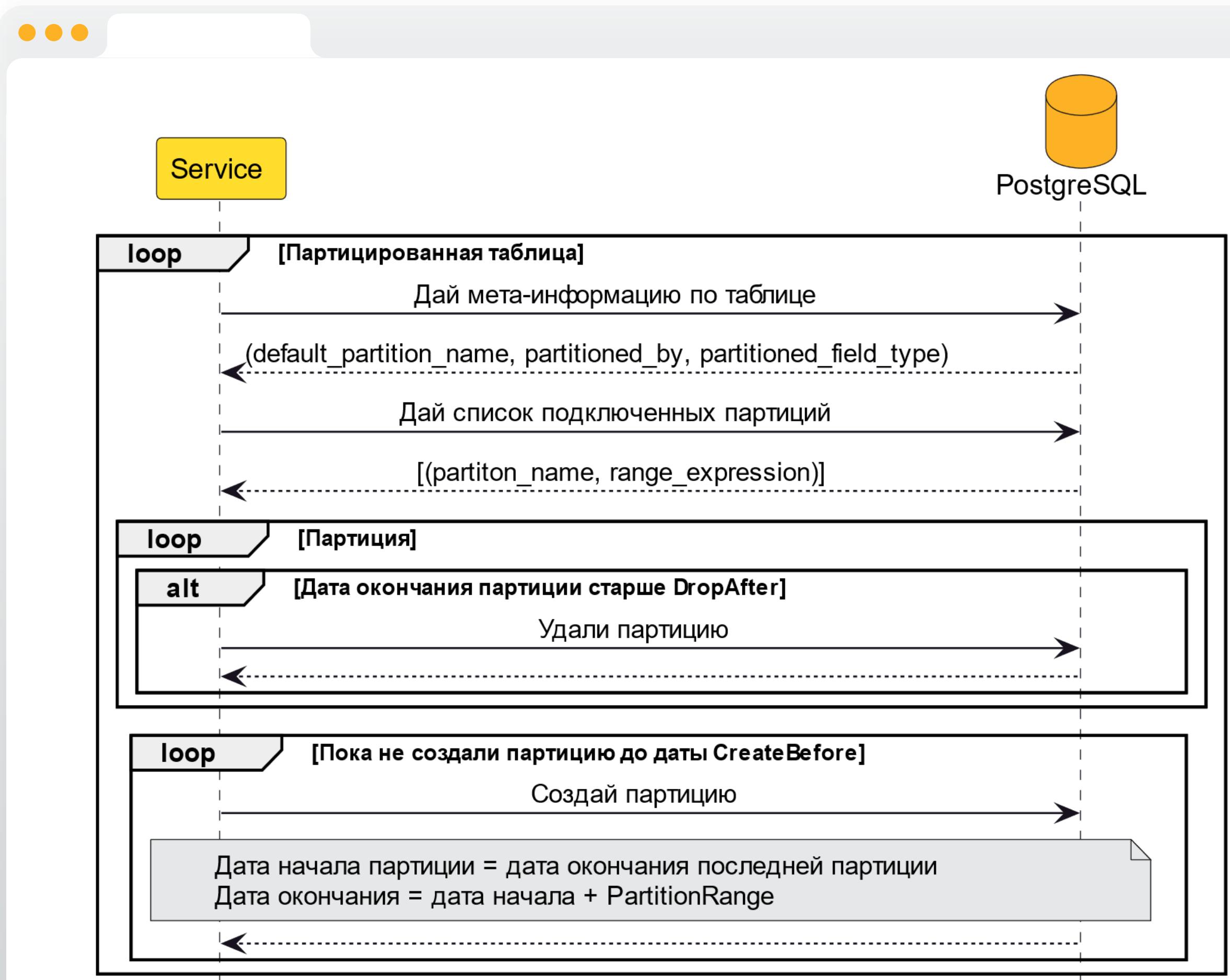
- **Name** – название партиционированной таблицы;
- **PartitionRange** – размер партиции;
- **CreateBefore** – за сколько дней **ДО** нужно создать партицию;
- **DropAfter** – через сколько **ПОСЛЕ** нужно удалить партицию.



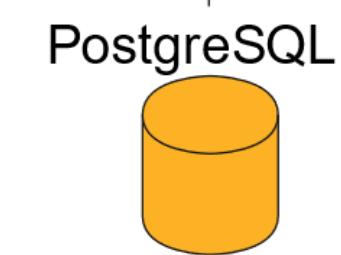
```
appsettings.json x
{
  "PartitionManager": {
    "Tables": [
      {
        "Name": "mappings",
        "PartitionRange": "1 month",
        "CreateBefore": "1 year",
        "DropAfter": "1 month"
      }
    ]
  }
}
```

The image shows a screenshot of a code editor window titled "appsettings.json x". The JSON configuration defines a "PartitionManager" object with a "Tables" array containing one table named "mappings". This table has properties: "Name" set to "mappings", "PartitionRange" set to "1 month", "CreateBefore" set to "1 year", and "DropAfter" highlighted with an orange border set to "1 month". The code editor interface includes standard elements like tabs, file icons, and a status bar at the bottom.

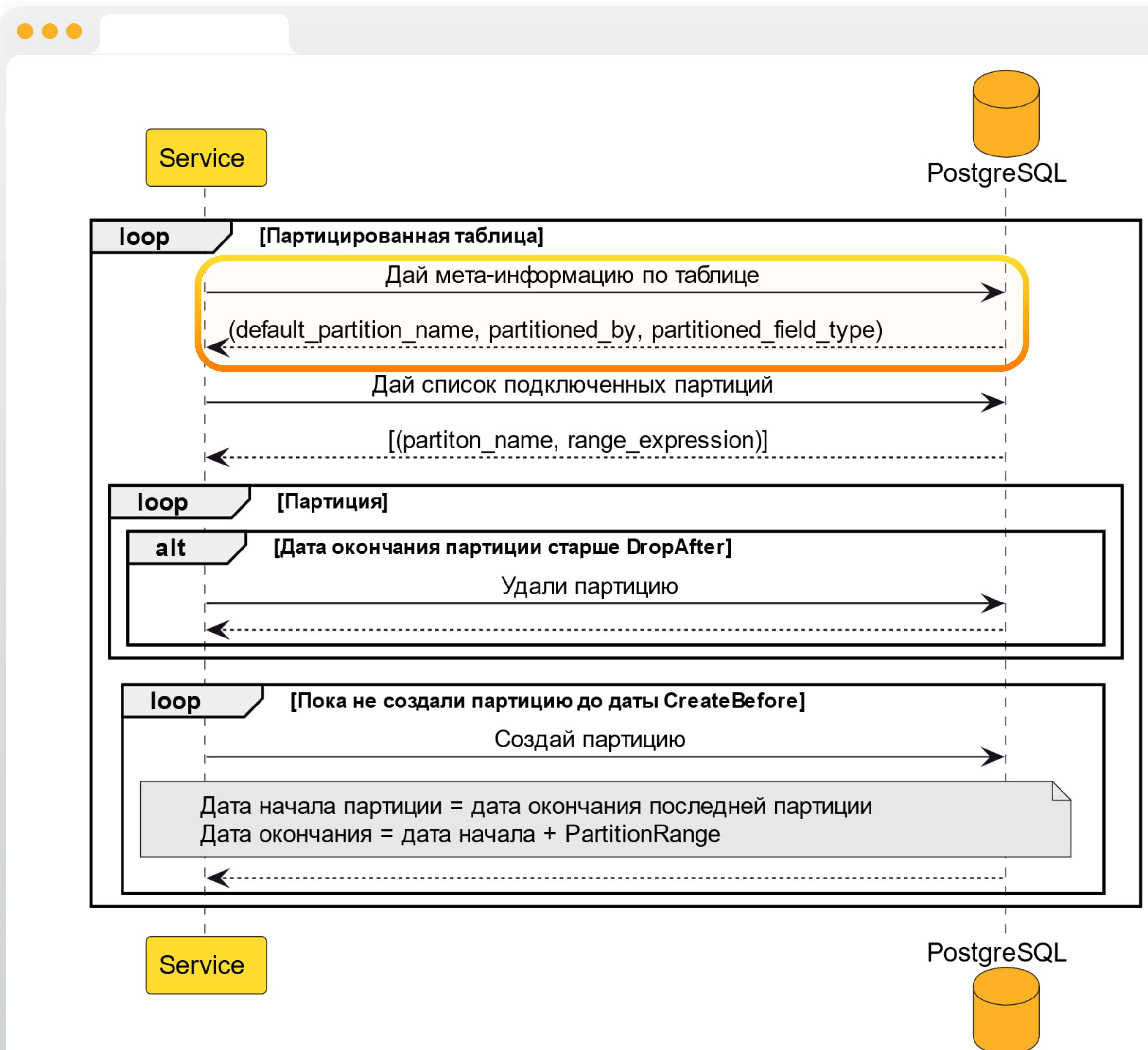
Алгоритм работы



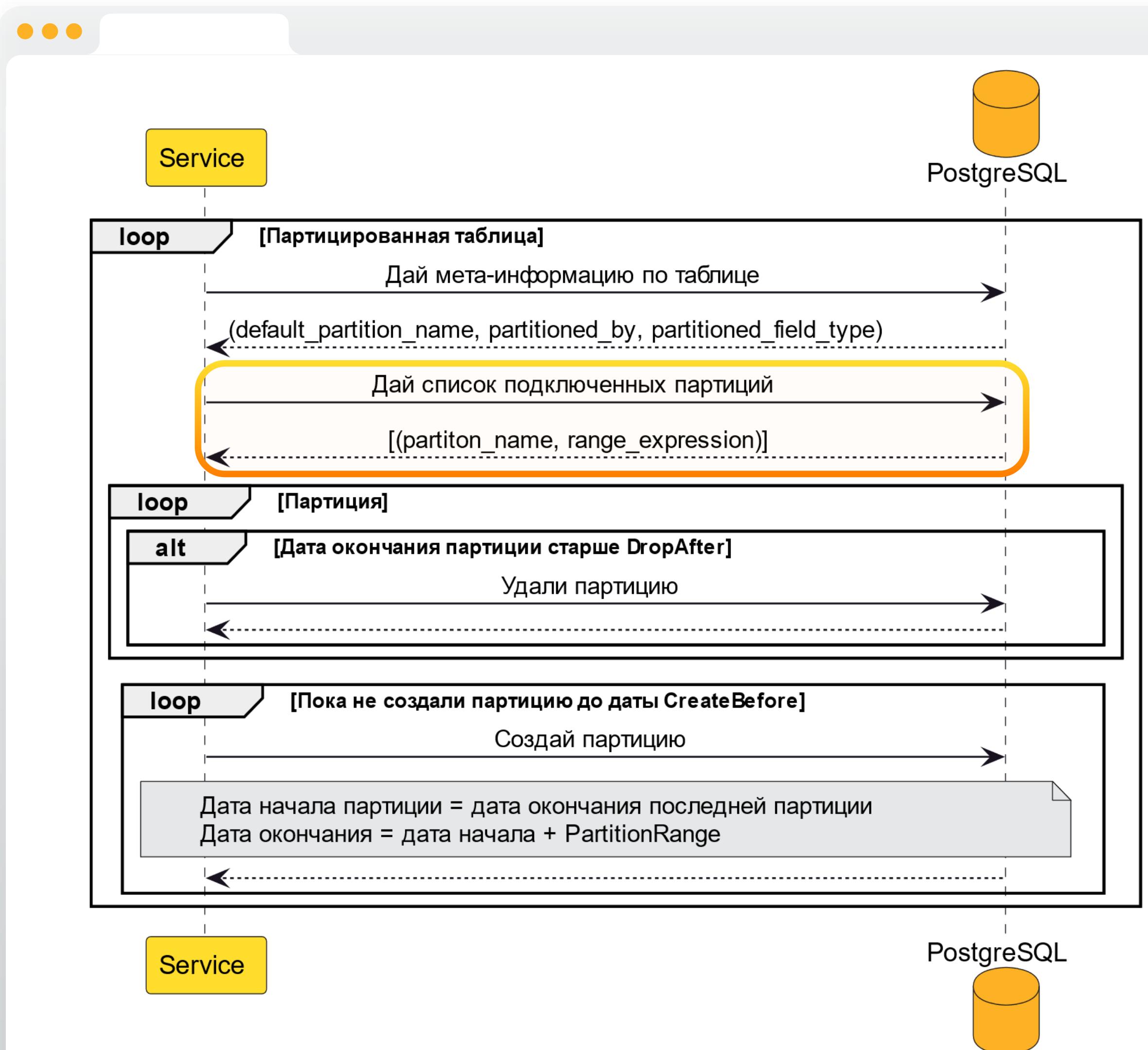
Service



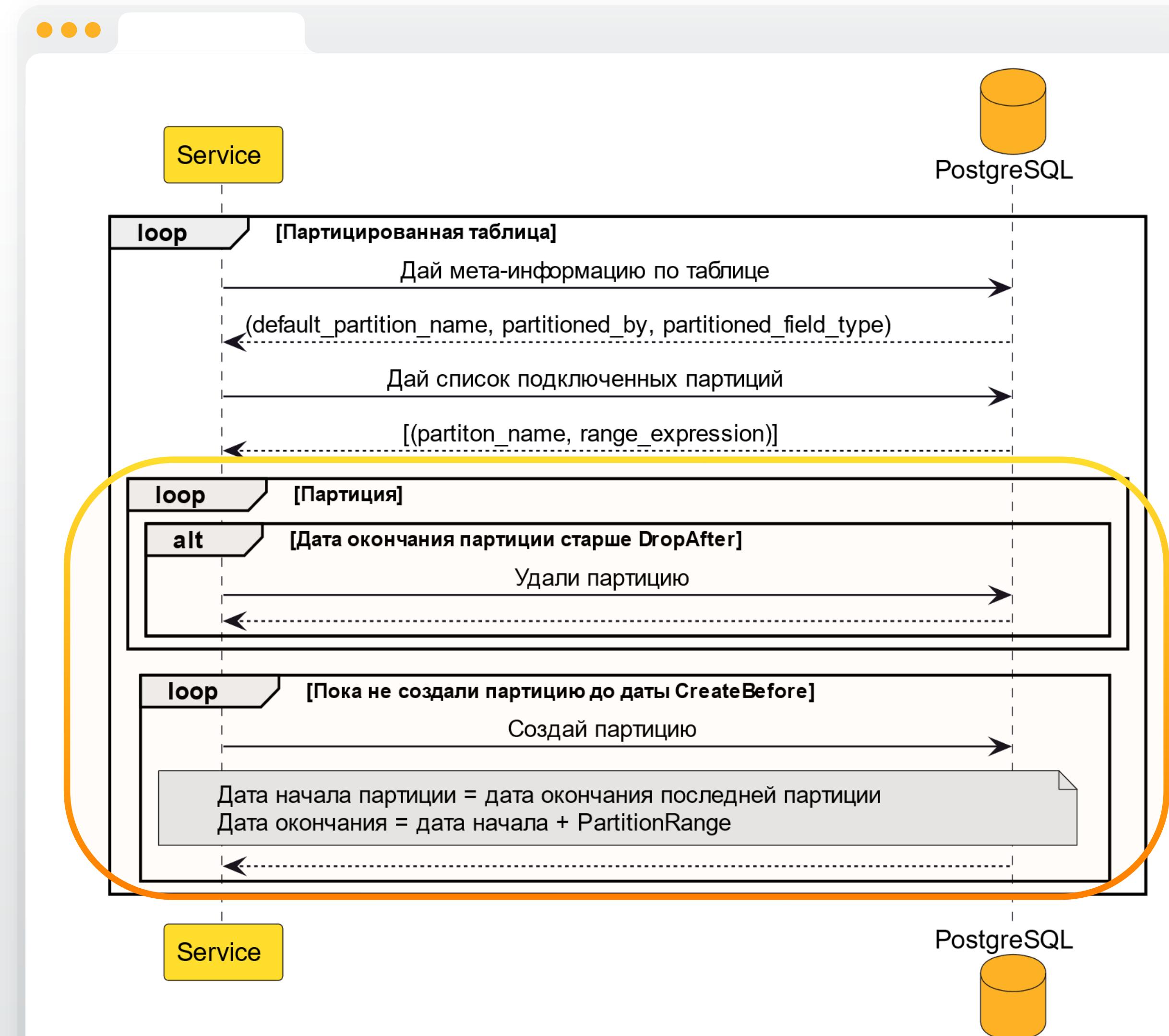
Алгоритм работы



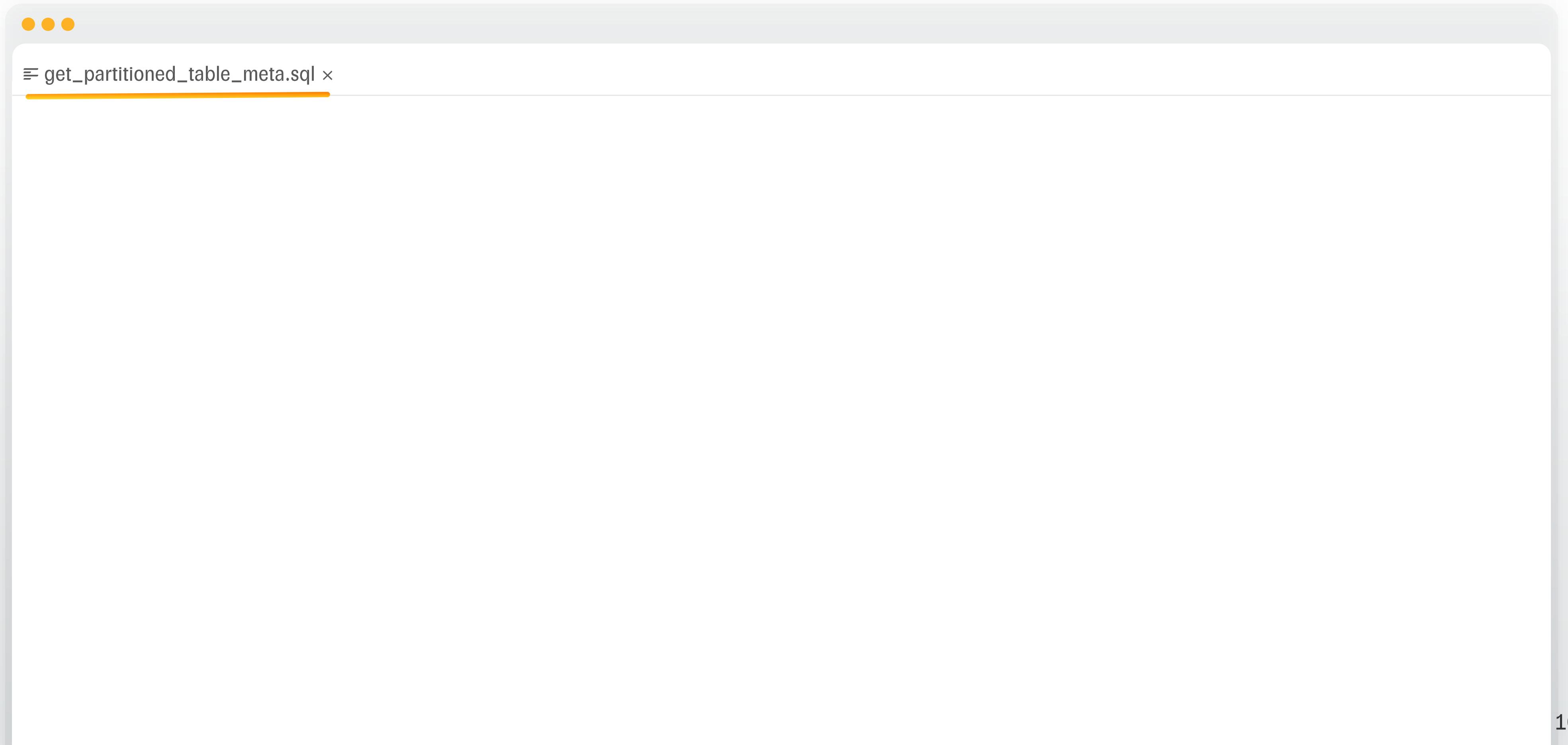
Алгоритм работы



Алгоритм работы



Запрос для получения меты



A screenshot of a code editor window titled "get_partitioned_table_meta.sql". The file is currently empty, indicated by the three yellow dots at the top of the editor area.

```
☰ get_partitioned_table_meta.sql ×
```

Запрос для получения меты

```
•••
≡ get_partitioned_table_meta.sql ×

SELECT
    defpart.relname AS default_partition_name,
    pa.attname AS partitioned_by,
    attr_type.typname AS partitioned_by_field_type
FROM pg_partitioned_table pt
JOIN pg_catalog.pg_class parent_class ON pt.partrelid = parent_class.oid
JOIN pg_catalog.pg_namespace parent_nsp ON parent_class.relnamespace = parent_nsp.oid
JOIN pg_catalog.pg_attribute pa ON pt.partrelid = pa.attrelid AND pa.attnum = ANY (pt.partatrs)
JOIN pg_catalog.pg_type attr_type ON pa.atttypid = attr_type.oid
LEFT JOIN pg_catalog.pg_class defpart ON pt.partdefid = defpart.oid
WHERE
    parent_class.relname = 'mappings'
    AND parent_nsp.nspname = 'public'
    AND pt.partstrat = 'r'
```

Запрос для получения меты

```
• • •  
≡ get_partitioned_table_meta.sql ×  
  
SELECT  
    defpart.relname AS default_partition_name,  
    pa.attname AS partitioned_by,  
    attr_type.typname AS partitioned_by_field_type  
FROM pg_partitioned_table pt  
JOIN pg_catalog.pg_class parent_class ON pt.partrelid = parent_class.oid  
JOIN pg_catalog.pg_namespace parent_nsp ON parent_class.relnamespace = parent_nsp.oid  
JOIN pg_catalog.pg_attribute pa ON pt.partrelid = pa.attrelid AND pa.attnum = ANY (pt.partatrs)  
JOIN pg_catalog.pg_type attr_type ON pa.atttypid = attr_type.oid  
LEFT JOIN pg_catalog.pg_class defpart ON pt.partdefid = defpart.oid  
WHERE  
    parent_class.relname = 'mappings'  
    AND parent_nsp.nspname = 'public'  
    AND pt.partstrat = 'r'
```

Запрос для полученияパーティций

```
• • •
≡ get_partitions.sql ×

SELECT
    child_class.relname AS partition_name,
    pe.part_expr AS range_expression
FROM pg_catalog.pg_partitioned_table pt
JOIN pg_catalog.pg_class parent_class ON pt.partrelid = parent_class.oid
JOIN pg_catalog.pg_namespace parent_nsp ON parent_class.relnamespace = parent_nsp.oid
JOIN pg_catalog.pg_inherits i ON pt.partrelid = i.inhparent
JOIN pg_catalog.pg_class child_class ON i.inherelid = child_class.oid
LEFT JOIN LATERAL (
    SELECT pg_catalog.pg_get_expr(child_class.relpartbound, child_class.oid) AS part_expr
) AS pe ON TRUE
WHERE
    parent_class.relname = 'mappings'
    AND parent_nsp.nspname = 'public'
    AND pt.partdefid != child_class.oid
```

Запрос для полученияパーティций

```
•••
get_partitions.sql ×

SELECT
    child_class.relname AS partition_name,
    pe.part_expr AS range_expression
FROM pg_catalog.pg_partitioned_table pt
JOIN pg_catalog.pg_class parent_class ON pt.partrelid = parent_class.oid
JOIN pg_catalog.pg_namespace parent_nsp ON parent_class.relnamespace = parent_nsp.oid
JOIN pg_catalog.pg_inherits i ON pt.partrelid = i.inhparent
JOIN pg_catalog.pg_class child_class ON i.inherelid = child_class.oid
LEFT JOIN LATERAL (
    SELECT pg_catalog.pg_get_expr(child_class.relpartbound, child_class.oid) AS part_expr
) AS pe ON TRUE
WHERE
    parent_class.relname = 'mappings'
    AND parent_nsp.nspname = 'public'
    AND pt.partdefid != child_class.oid
```



Итого

Проблемы

- ➡ Процедуры сложно переиспользовать в других сервисах.
- ➡ pg_cron устанавливается только в одну базу данных на одном сервере PostgreSQL.
- ➡ Отсутствует мониторинг. Если партиция не создалась, то об этом узнаем не сразу.

Проблемы

- ➡ ~~Процедуры сложно переиспользовать в других сервисах.~~
- ➡ pg_cron устанавливается только в одну базу данных на одном сервере PostgreSQL.
- ➡ Отсутствует мониторинг. Если партиция не создалась, то об этом узнаем не сразу.

Проблемы

- ~~Процедуры сложно переиспользовать в других сервисах.~~
- ~~рр_соп устанавливается только в одну базу данных на одном сервере PostgreSQL.~~
- Отсутствует мониторинг. Если партиция не создалась, то об этом узнаем не сразу.

Проблемы

- ~~Процедуры сложно переиспользовать в других сервисах.~~
- ~~рр_соп устанавливается только в одну базу данных на одном сервере PostgreSQL.~~
- ~~Отсутствует мониторинг. Если партиция не создалась, то об этом узнаем не сразу.~~

Как подобрать размер партиции?

Объем данных

Меньше – лучше

Идеально, когда партиция полностью
помещается в буфер сервера

Объем данных

Количество партиций

VS

Меньше – лучше

Идеально, когда партиция полностью помещается в буфер сервера

Больше – хуже

Если есть запросы, которые ходят сразу во множество партиций, то партиций должно быть меньше

Партиционированные таблицы и EF Core

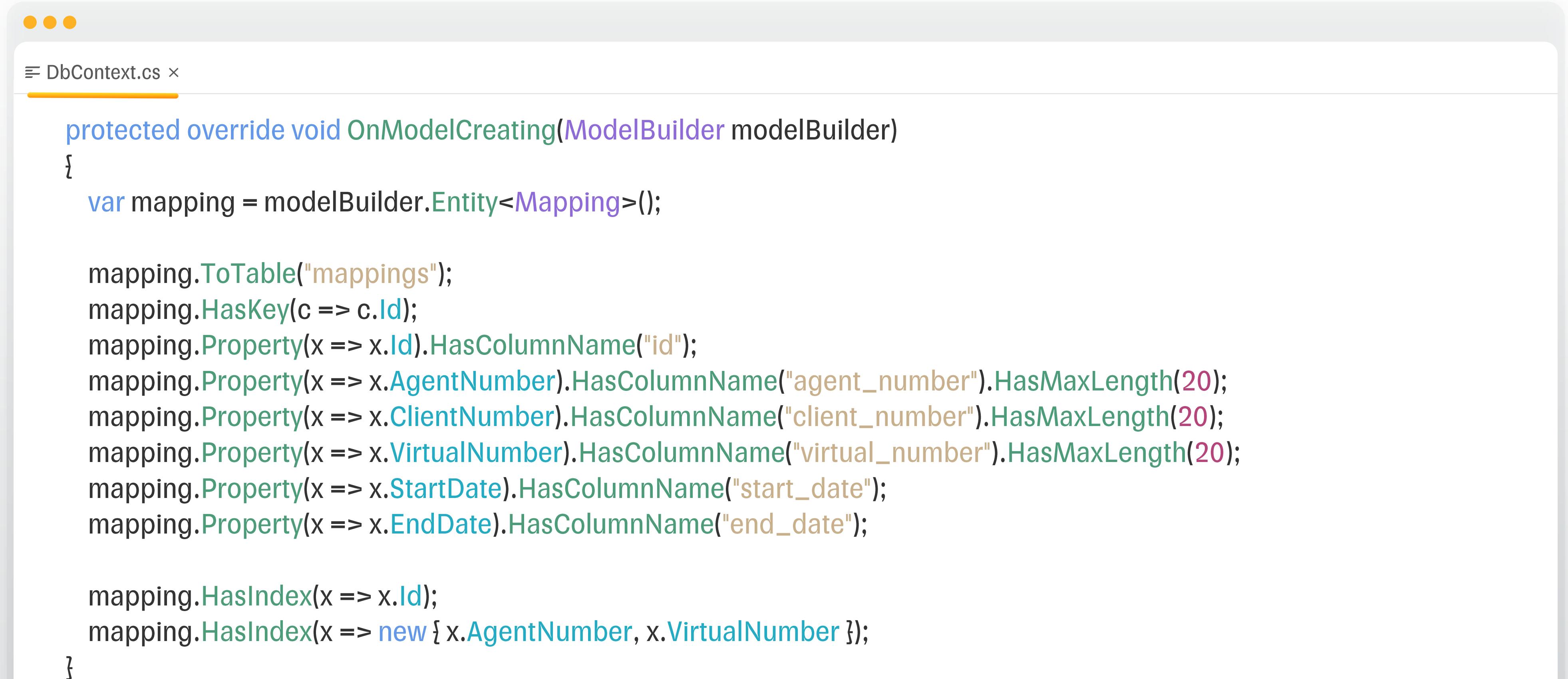
Модель



The screenshot shows a code editor window with a tab labeled "Mappings.cs". The code defines a public class named "Mapping" with several properties:

```
public class Mapping
{
    public Guid Id { get; init; }
    public required string AgentNumber { get; set; }
    public required string ClientNumber { get; set; }
    public required string VirtualNumber { get; init; }
    public DateTimeOffset StartDate { get; init; }
    public DateTimeOffset? EndDate { get; set; }
}
```

Конфигурация модели



The screenshot shows a code editor window with a tab labeled "DbContext.cs". The code is written in C# and defines a method for configuring a database model. The code uses Entity Framework's fluent API to map properties to columns in a table named "mappings". It specifies column names and maximum lengths for various properties like AgentNumber, ClientNumber, and VirtualNumber. It also creates indexes on the Id and a composite key consisting of AgentNumber and VirtualNumber.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    var mapping = modelBuilder.Entity<Mapping>();

    mapping.ToTable("mappings");
    mapping.HasKey(c => c.Id);
    mapping.Property(x => x.Id).HasColumnName("id");
    mapping.Property(x => x.AgentNumber).HasColumnName("agent_number").HasMaxLength(20);
    mapping.Property(x => x.ClientNumber).HasColumnName("client_number").HasMaxLength(20);
    mapping.Property(x => x.VirtualNumber).HasColumnName("virtual_number").HasMaxLength(20);
    mapping.Property(x => x.StartDate).HasColumnName("start_date");
    mapping.Property(x => x.EndDate).HasColumnName("end_date");

    mapping.HasIndex(x => x.Id);
    mapping.HasIndex(x => new { x.AgentNumber, x.VirtualNumber });
}
```

Конфигурация модели

```
...  
DbContext.cs x  
  
protected override void OnModelCreating(ModelBuilder modelBuilder)  
{  
    var mapping = modelBuilder.Entity<Mapping>();  
  
    mapping.ToTable("mappings");  
    mapping.HasKey(c => c.Id);  
    mapping.Property(x => x.Id).HasColumnName("id");  
    mapping.Property(x => x.AgentNumber).HasColumnName("agent_number").HasMaxLength(20);  
    mapping.Property(x => x.ClientNumber).HasColumnName("client_number").HasMaxLength(20);  
    mapping.Property(x => x.VirtualNumber).HasColumnName("virtual_number").HasMaxLength(20);  
    mapping.Property(x => x.StartDate).HasColumnName("start_date");  
    mapping.Property(x => x.EndDate).HasColumnName("end_date");  
  
    mapping.HasIndex(x => x.Id);  
    mapping.HasIndex(x => new { x.AgentNumber, x.VirtualNumber });  
}
```

Конфигурация модели

```
...  
DbContext.cs x  
  
protected override void OnModelCreating(ModelBuilder modelBuilder)  
{  
    var mapping = modelBuilder.Entity<Mapping>();  
  
    mapping.ToTable("mappings");  
    mapping.HasKey();  
    mapping.Property(x => x.Id).HasColumnName("id");  
    mapping.Property(x => x.AgentNumber).HasColumnName("agent_number").HasMaxLength(20);  
    mapping.Property(x => x.ClientNumber).HasColumnName("client_number").HasMaxLength(20);  
    mapping.Property(x => x.VirtualNumber).HasColumnName("virtual_number").HasMaxLength(20);  
    mapping.Property(x => x.StartDate).HasColumnName("start_date");  
    mapping.Property(x => x.EndDate).HasColumnName("end_date");  
  
    mapping.HasIndex(x => x.Id);  
    mapping.HasIndex(x => new { x.AgentNumber, x.VirtualNumber });  
}
```

Операции над моделью

В EF модель без первичного ключа
не может отслеживать изменения.

Select



Insert



Update



Delete



Операции над моделью

В EF модель без первичного ключа
не может отслеживать изменения.

Select



Insert



Update



Delete



Операции над моделью

В EF модель без первичного ключа
не может отслеживать изменения.

Select



Insert



Update



Delete



Операции над моделью

В EF модель без первичного ключа
не может отслеживать изменения.

Select



Insert



Update



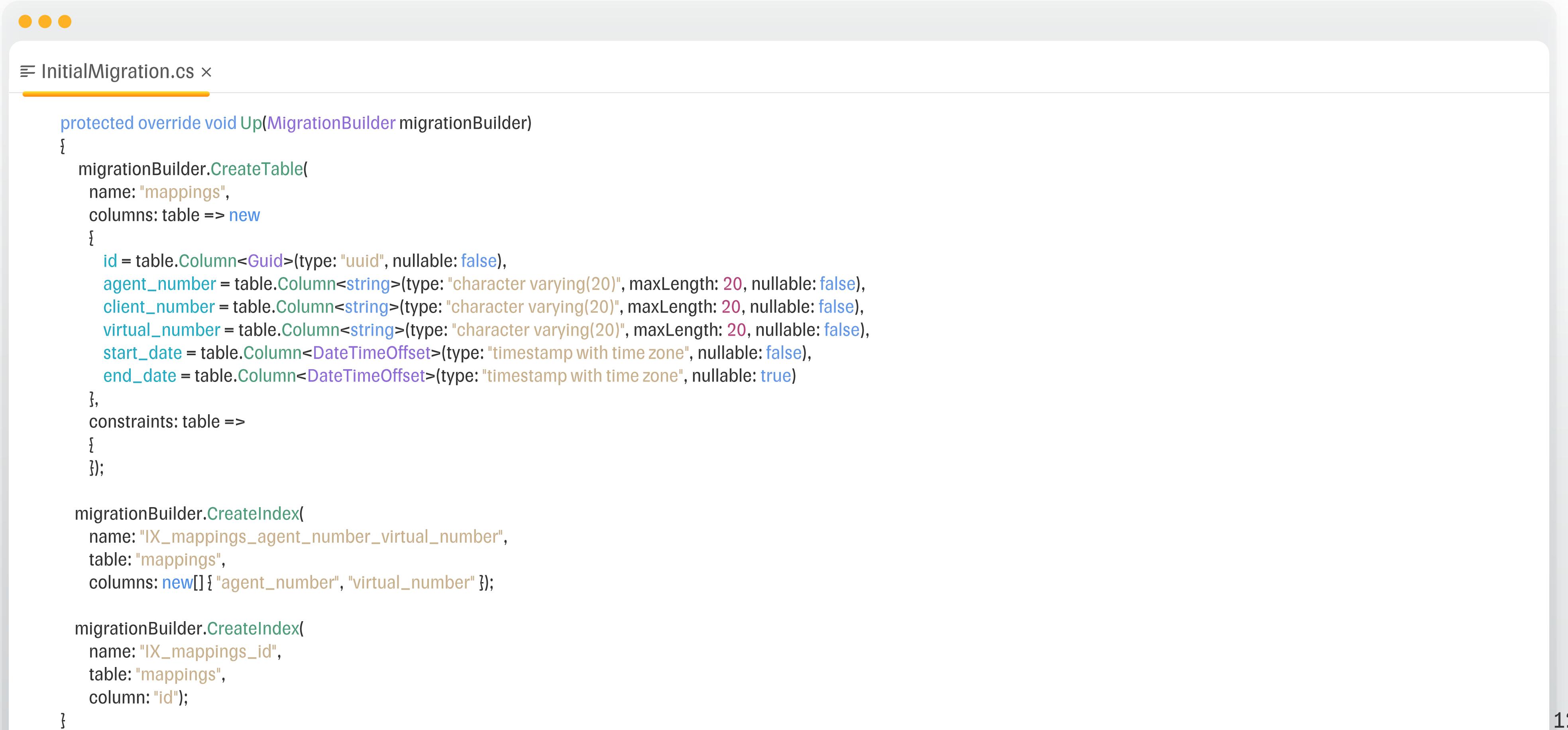
Delete



Конфигурация модели

```
...  
DbContext.cs x  
  
protected override void OnModelCreating(ModelBuilder modelBuilder)  
{  
    var mapping = modelBuilder.Entity<Mapping>();  
  
    mapping.ToTable("mappings");  
    mapping.HasKey(c => c.Id);  
    mapping.Property(x => x.Id).HasColumnName("id");  
    mapping.Property(x => x.AgentNumber).HasColumnName("agent_number").HasMaxLength(20);  
    mapping.Property(x => x.ClientNumber).HasColumnName("client_number").HasMaxLength(20);  
    mapping.Property(x => x.VirtualNumber).HasColumnName("virtual_number").HasMaxLength(20);  
    mapping.Property(x => x.StartDate).HasColumnName("start_date");  
    mapping.Property(x => x.EndDate).HasColumnName("end_date");  
  
    mapping.HasIndex(x => x.Id);  
    mapping.HasIndex(x => new { x.AgentNumber, x.VirtualNumber });  
}
```

Миграция



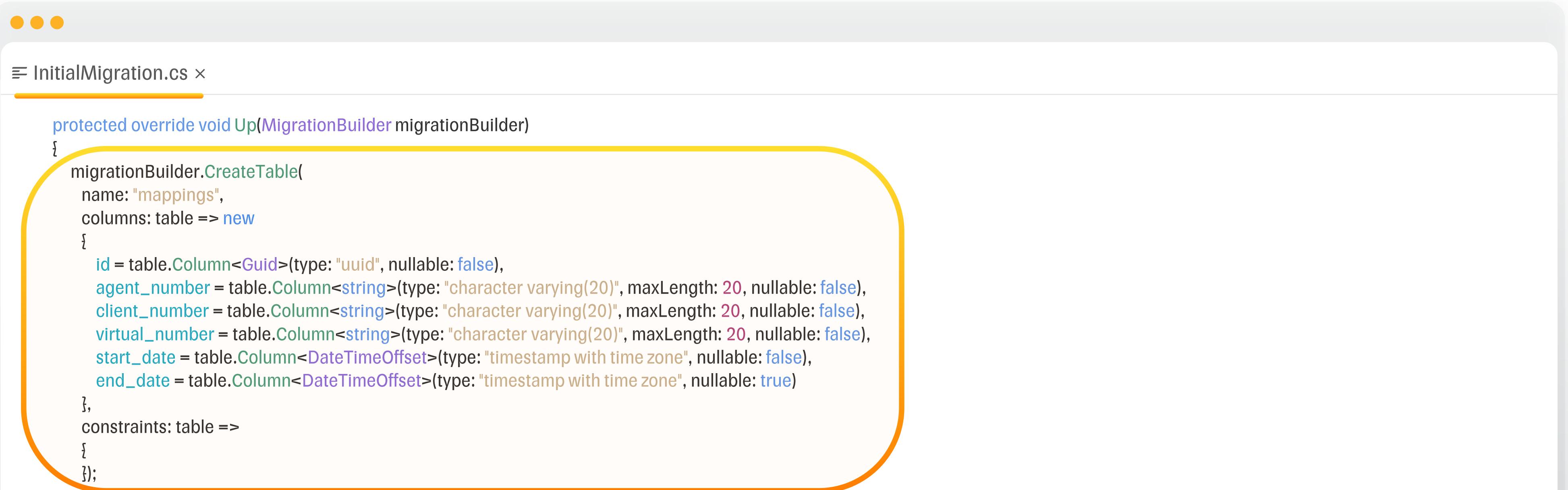
The screenshot shows a code editor window with a tab labeled "InitialMigration.cs". The code is written in C# and defines a migration class. The class has a protected override void Up method that creates a table named "mappings" with several columns: id (uuid, nullable false), agent_number (string, max length 20, nullable false), client_number (string, max length 20, nullable false), virtual_number (string, max length 20, nullable false), start_date (DateTimeOffset, nullable false), and end_date (DateTimeOffset, nullable true). It also creates two indexes: IX_mappings_agent_number_virtual_number on columns agent_number and virtual_number, and IX_mappings_id on column id.

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "mappings",
        columns: table => new
        {
            id = table.Column<Guid>(type: "uuid", nullable: false),
            agent_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            client_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            virtual_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            start_date = table.Column<DateTimeOffset>(type: "timestamp with time zone", nullable: false),
            end_date = table.Column<DateTimeOffset>(type: "timestamp with time zone", nullable: true)
        },
        constraints: table =>
        {
        });
}

migrationBuilder.CreateIndex(
    name: "IX_mappings_agent_number_virtual_number",
    table: "mappings",
    columns: new[] { "agent_number", "virtual_number" });

migrationBuilder.CreateIndex(
    name: "IX_mappings_id",
    table: "mappings",
    column: "id");
```

Миграция



```
InitialMigration.cs

protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "mappings",
        columns: table => new
        {
            id = table.Column<Guid>(type: "uuid", nullable: false),
            agent_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            client_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            virtual_number = table.Column<string>(type: "character varying(20)", maxLength: 20, nullable: false),
            start_date = table.Column<DateTimeOffset>(type: "timestamp with time zone", nullable: false),
            end_date = table.Column<DateTimeOffset>(type: "timestamp with time zone", nullable: true)
        },
        constraints: table =>
        {
        });
}

migrationBuilder.CreateIndex(
    name: "IX_mappings_agent_number_virtual_number",
    table: "mappings",
    columns: new[] { "agent_number", "virtual_number" });

migrationBuilder.CreateIndex(
    name: "IX_mappings_id",
    table: "mappings",
    column: "id");
```

Миграция

```
InitialMigration.cs

protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.Sql("""
        CREATE TABLE mappings (
            id UUID NOT NULL,
            agent_number VARCHAR(20) NOT NULL,
            client_number VARCHAR(20) NOT NULL,
            virtual_number VARCHAR(20) NOT NULL,
            start_date TIMESTAMPTZ NOT NULL,
            end_date TIMESTAMPTZ
        ) PARTITION BY RANGE (end_date)
        """);
}

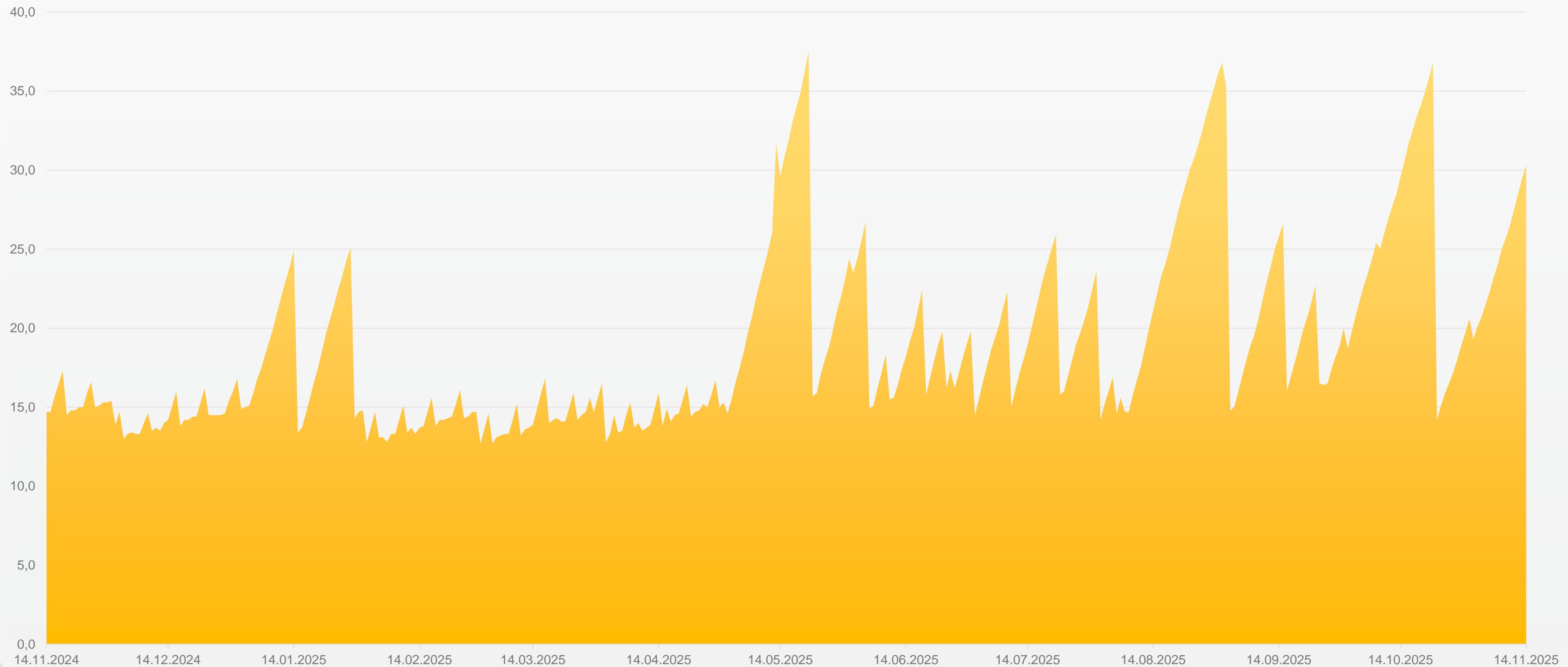
migrationBuilder.CreateIndex(
    name: "IX_mappings_agent_number_virtual_number",
    table: "mappings",
    columns: new[] { "agent_number", "virtual_number" });

migrationBuilder.CreateIndex(
    name: "IX_mappings_id",
    table: "mappings",
    column: "id");

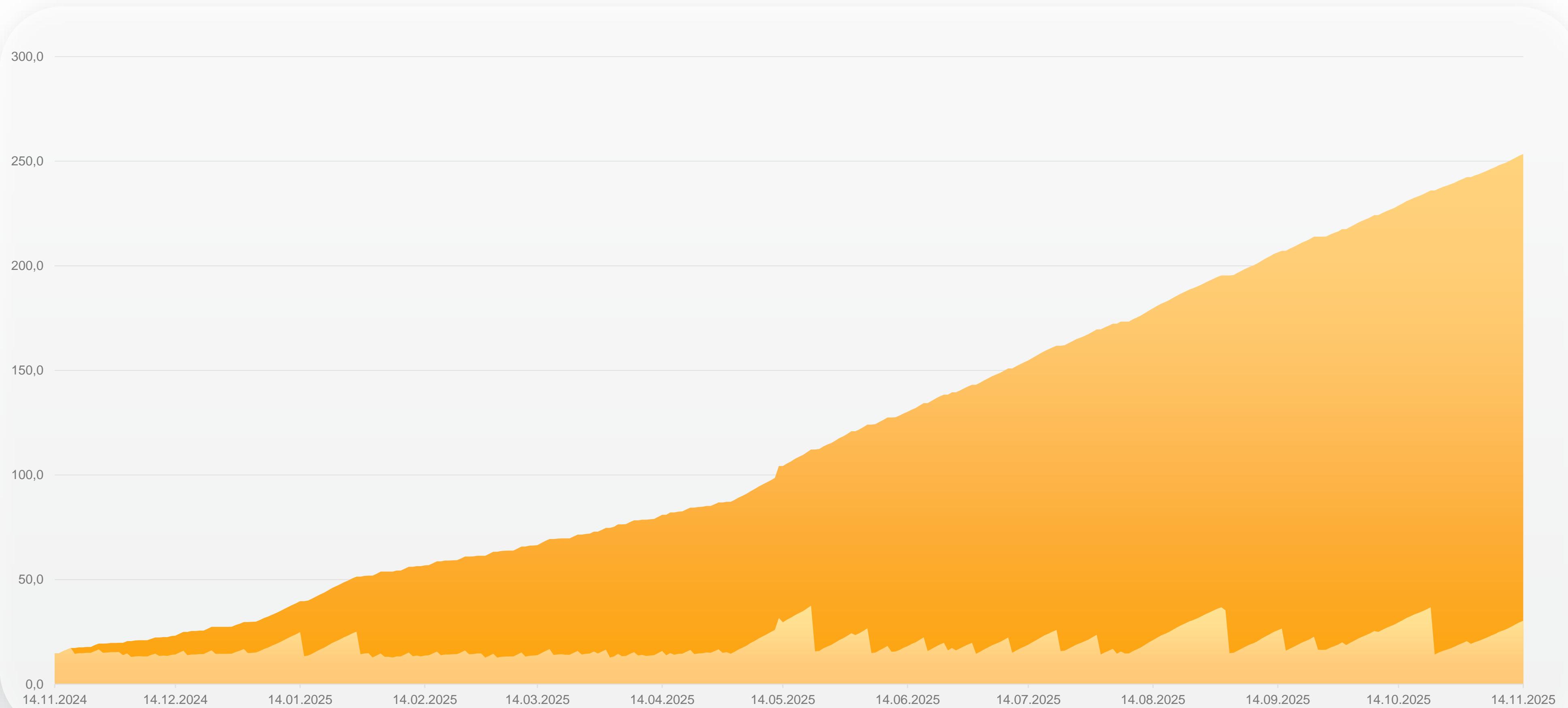
migrationBuilder.Sql("CREATE TABLE mapping_defaults PARTITION OF mappings DEFAULT");
}
```

Результаты на графиках

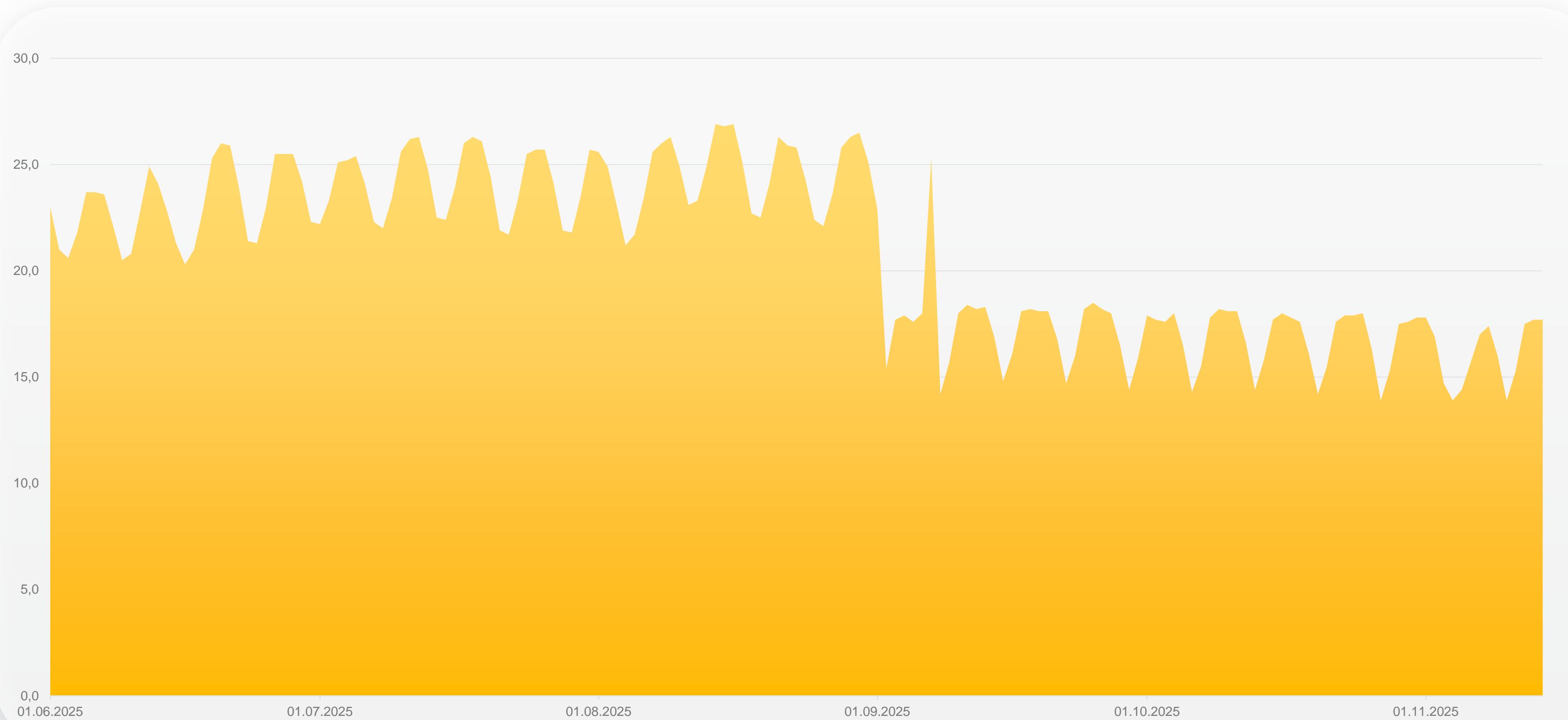
Размер БД связок



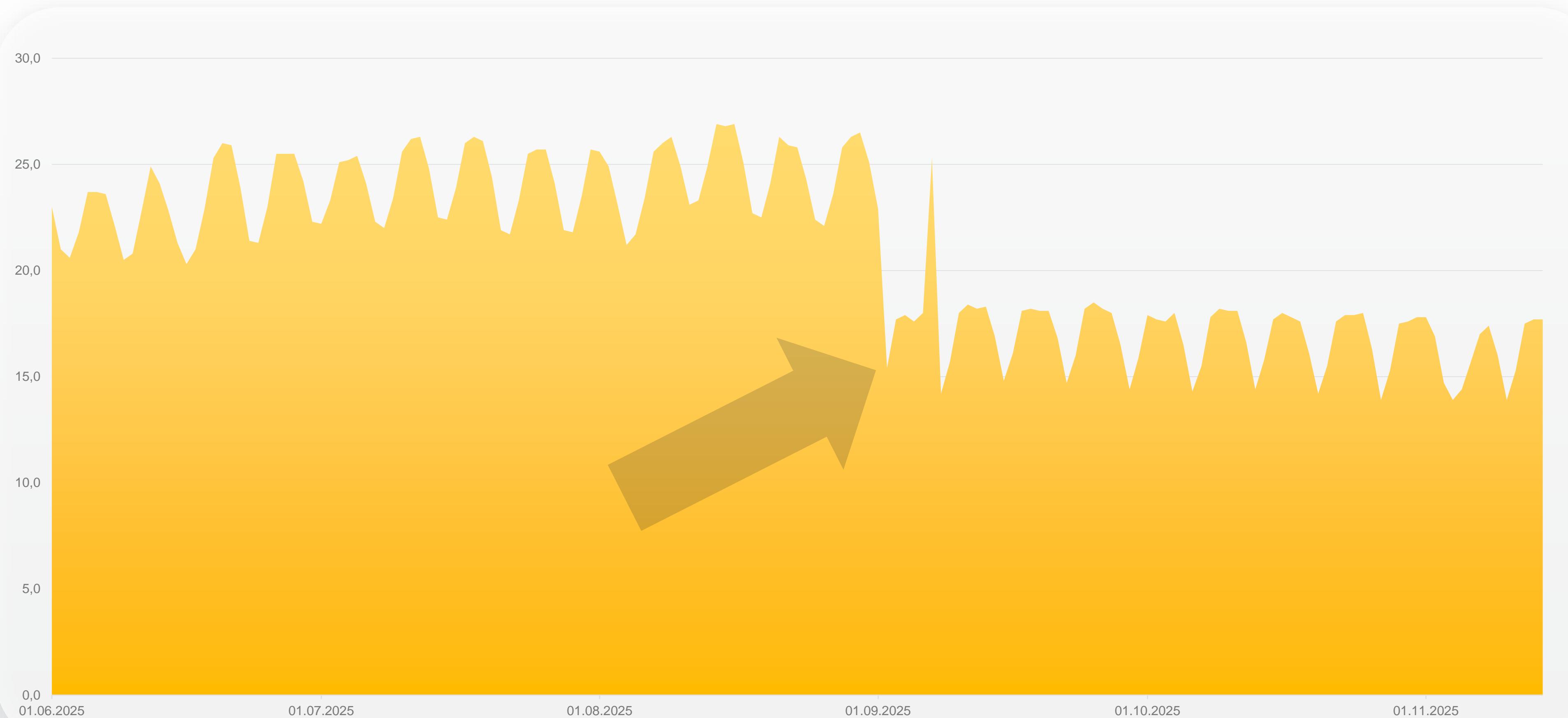
Размер БД связок



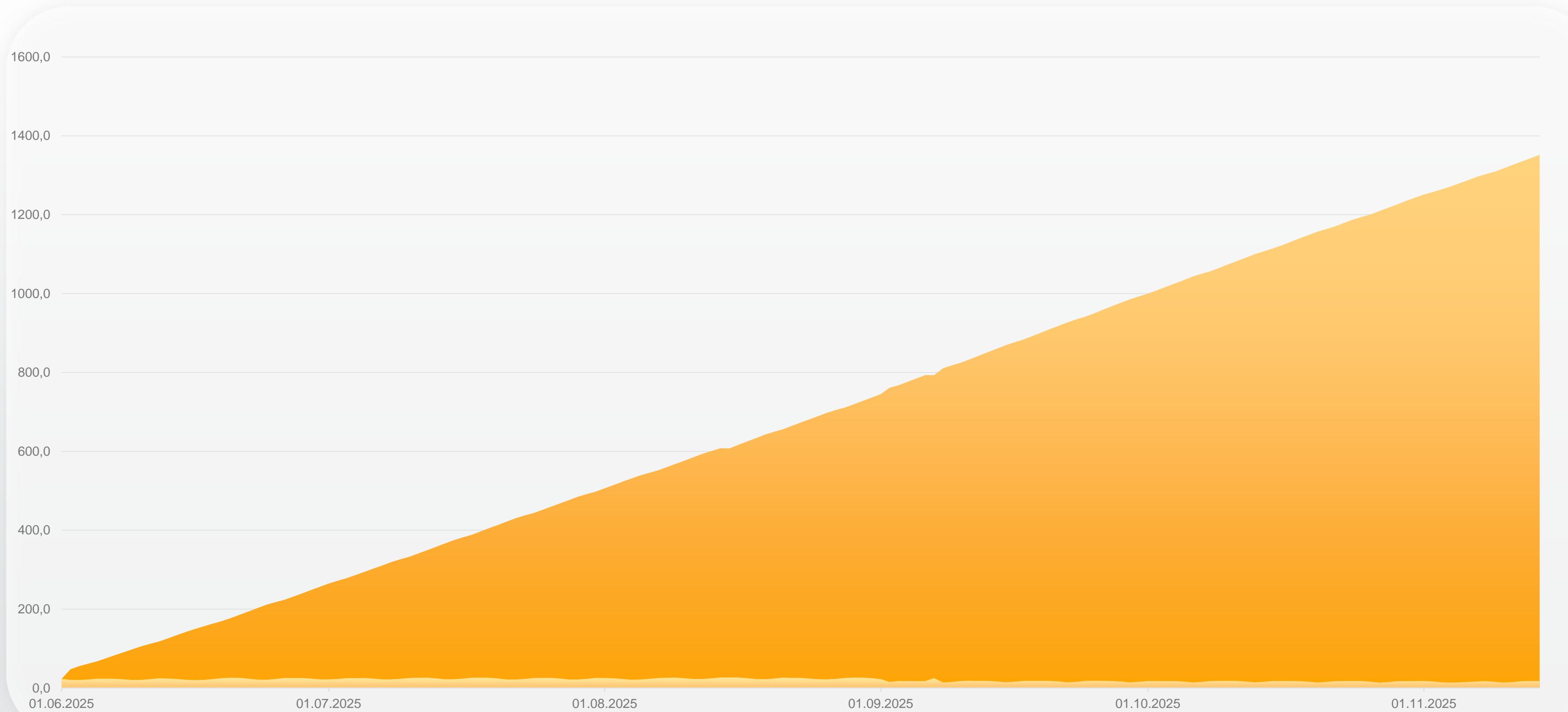
Размер Бд состояния звонков



Размер Бд состояния звонков



Размер Бд состояния звонков



Спасибо за внимание!

Ваши вопросы



Npgsql

LINQ

C#

PostgreSQL

DbContext

.NET

EF Core

SQL

Development

Proces

Продам

Гараж

