EDB POSTGRES AI

Deep dive into the PostgreSQL Frontend/Backend protocol

Xavier Fischer
PGConf Belgium 2026

©EDB 2026 — ALL RIGHTS RESERVED.

Who is using **psql**?

©EDB 2026 — ALL RIGHTS RESERVED.

How on Earth is this possible?

you type a query text, and…

```
dvdrental=# select * from actor limit 10;
 actor_id | first_name |  last_name    |        last_update
----------+------------+---------------+----------------------
        1 | Penelope   | Guiness       | 2013-05-26 14:47:57.62
        2 | Nick       | Wahlberg      | 2013-05-26
        3 | Ed         | Chase         | 2013-05-26
        4 | Jennifer   | Davis         | 2013-05-26
        5 | Johnny     | Lollobrigida  | 2013-05-26
        6 | Bette      | Nicholson     | 2013-05-26
```

… you get data back

©EDB 2026 — ALL RIGHTS RESERVED.

psql does NOT know beforehand

- if the query syntax is OK
- how many columns the results will have
- the name, type for each column value

```
dvdrental=# select * from actor limit 10;
 actor_id | first_name |  last_name    |      last_update
----------+------------+---------------+-----------------------
        1 | Penelope   | Guiness       | 2013-05-26 14:47:57.62
        2 | Nick       | Wahlberg      | 2013-05-26 14:47:57.62
        3 | Ed         | Chase         | 2013-05-26 14:47:57.62
        4 | Jennifer   | Davis         | 2013-05-26 14:47:57.62
        5 | Johnny     | Lollobrigida  | 2013-05-26 14:47:57.62
        6 | Bette      | Nicholson     | 2013-05-26 14:47:57.62
```

This is handled by libpq (the “**connector**”) after exchanging several messages with PostgreSQL

©EDB 2026 — ALL RIGHTS RESERVED.

Who am I

Xavier Fischer

- Principal Software Engineer at EDB since 2023
- Working on EDB .NET Connector
- 20+ years of experience in .NET / database development
- Hailing in Aix-en-Provence, France

©EDB 2026 — ALL RIGHTS RESERVED.

Who am I

Xavier Fischer

- Principal Software Engineer at EDB sin
- Working on EDB .NET Connector
- 20+ years of experience in .NET / da
- Hailing in Aix-en-Provence, France

EDB POSTGRES AI

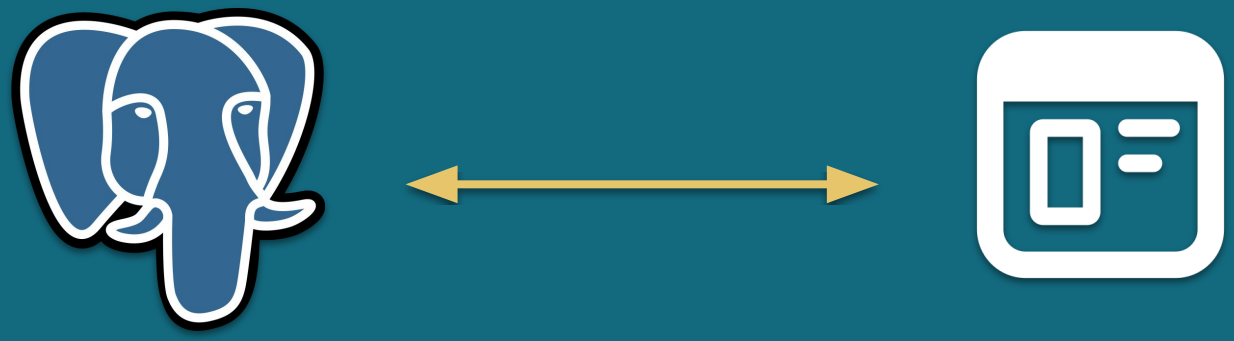
©EDB 2026 — ALL RIGHTS RESERVED.

What is a **connector**?

©EDB 2026 — ALL RIGHTS RESERVED.

it's a **database driver**

available in **your programming language**
will **translate** calls to database language using
the **wire protocol**
aka the Frontend/Backend protocol



EDB POSTGRES AI

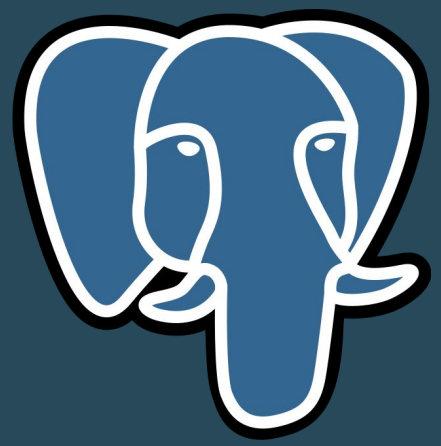
©EDB 2026 — ALL RIGHTS RESERVED.

what’s the **wire protocol**?

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

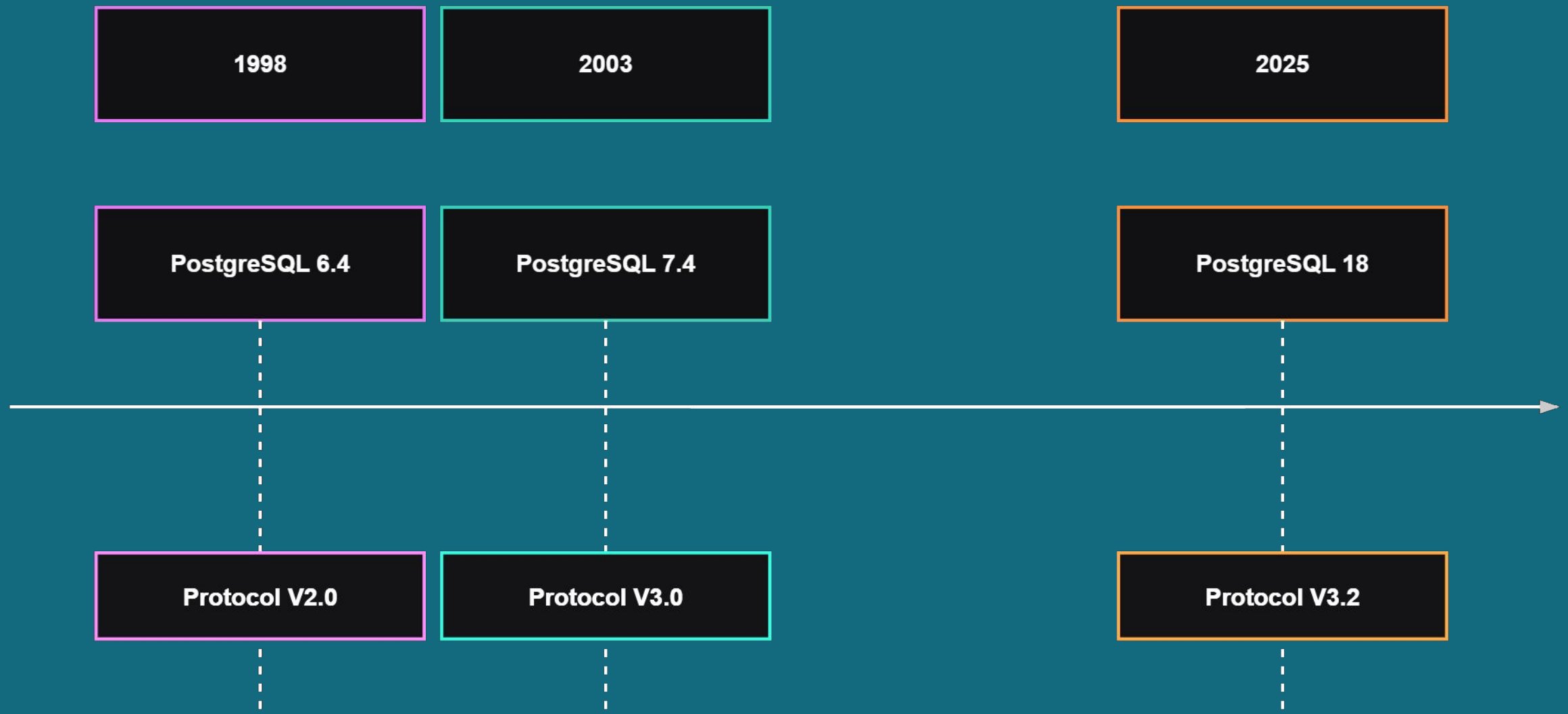
It’s the PostgreSQL **message-based** protocol between **frontends and backends** (clients, servers, poolers, tools, …) over TCP/IP or Unix sockets



EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

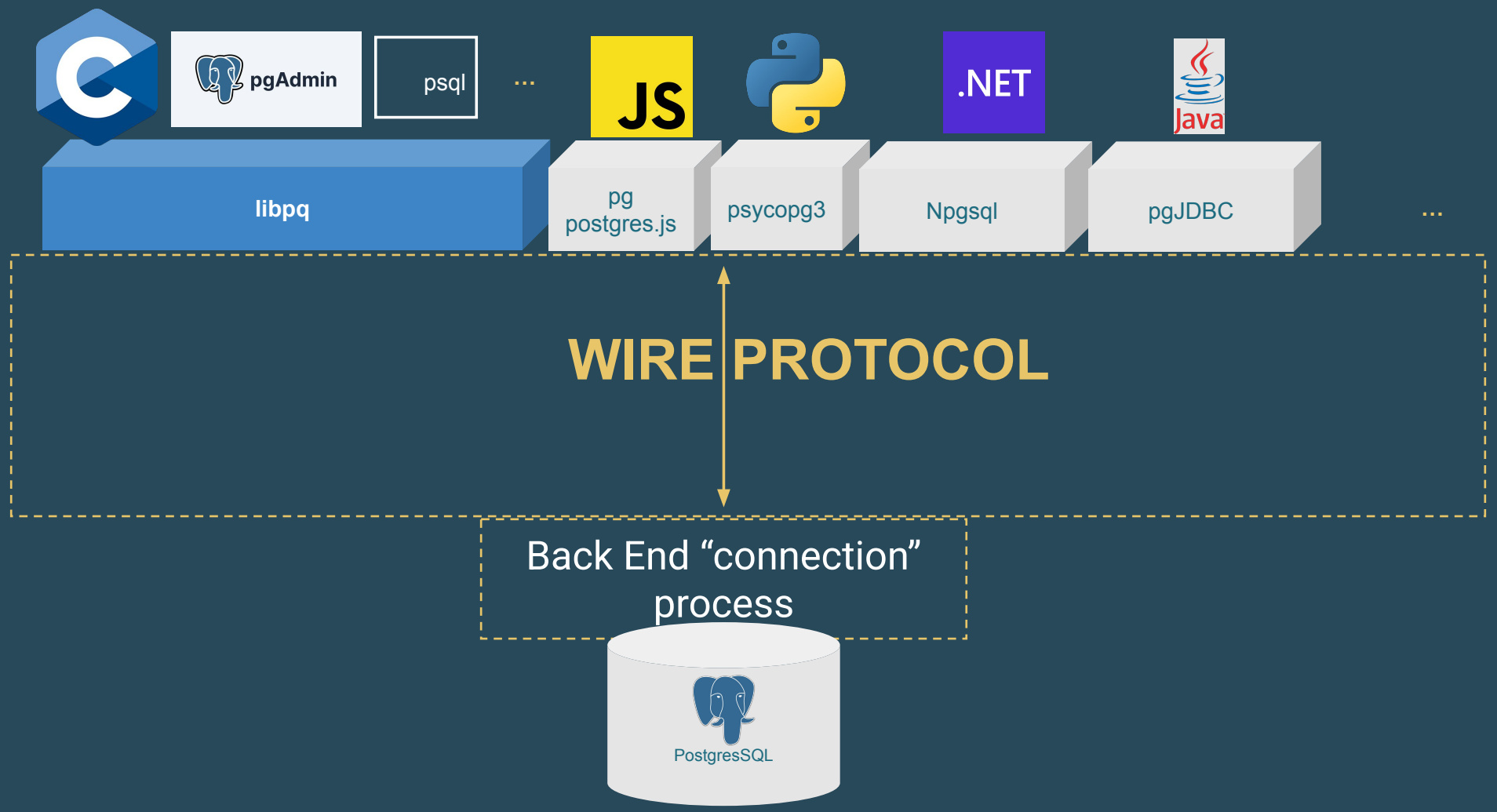
Protocol history



EDB POSTGRES AI

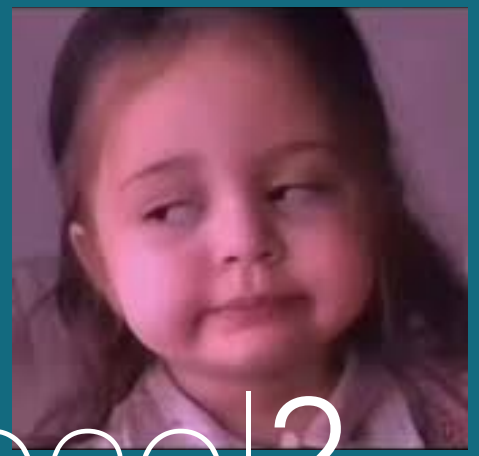
©EDB 2026 — ALL RIGHTS RESERVED.

Postgres Connector landscape



©EDB 2026 — ALL RIGHTS RESERVED.

ok
but
what **is** the wire protocol?

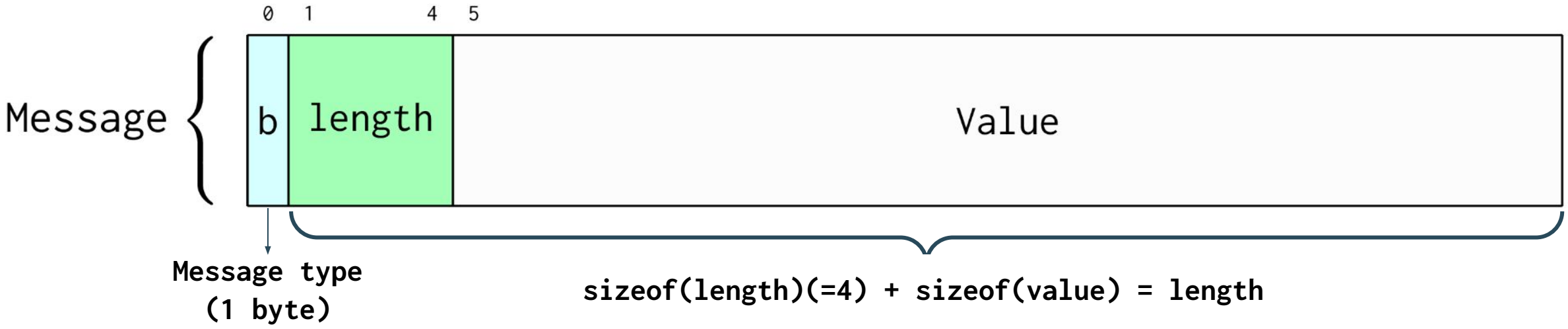


EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

Protocol message format 101

Messages are usually Type-Length-Value (TLV)



=> If a new protocol is released, unknown messages can be skipped easily

Protocol conversation 101

TCP or Unix socket connection

Encryption channel

Direct

Password

GSSAPI

...

Simple Query
Extended Query

Replication

Copy

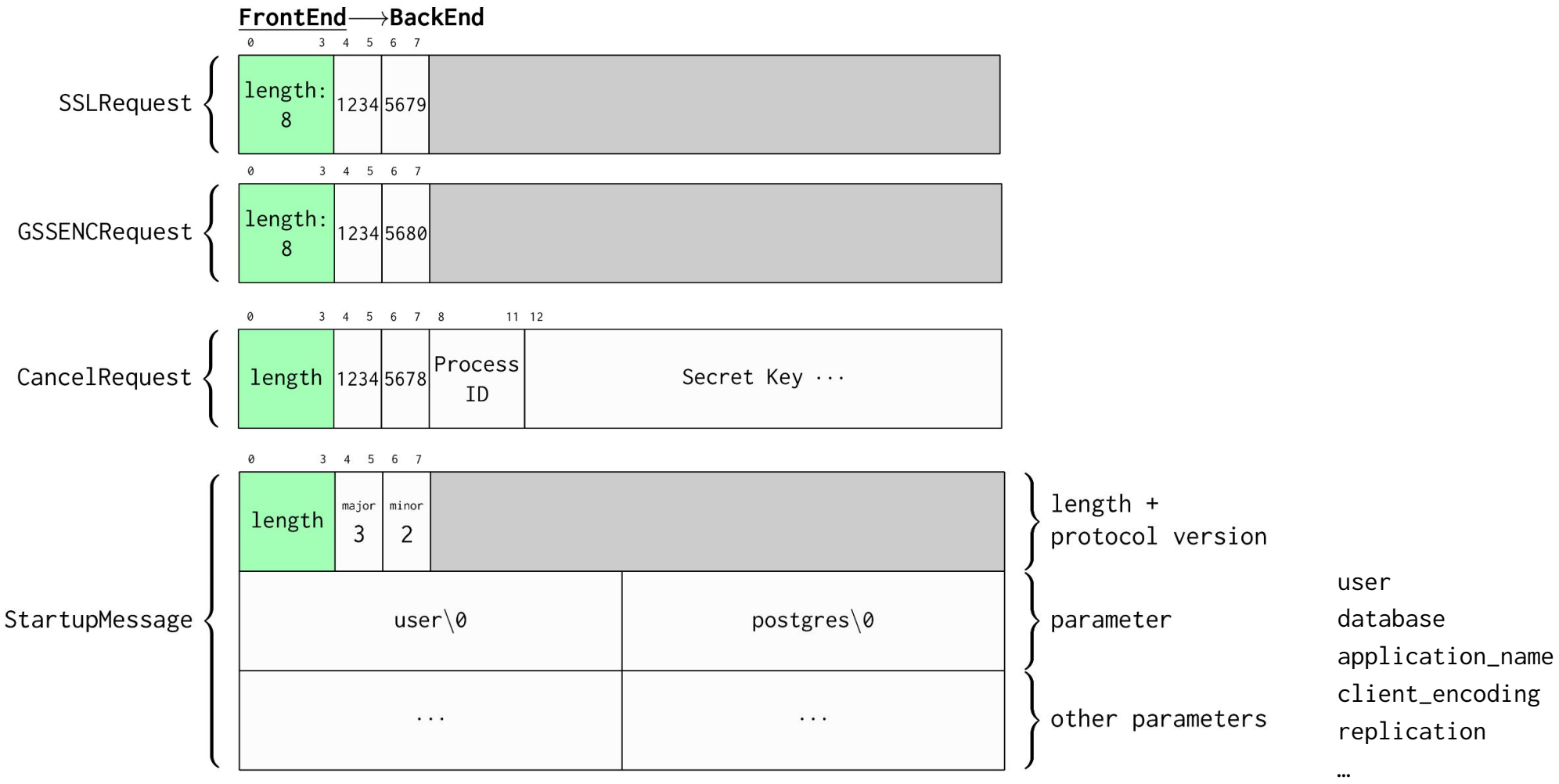
Listen/Notify

Free resources

©EDB 2026 — ALL RIGHTS RESERVED.

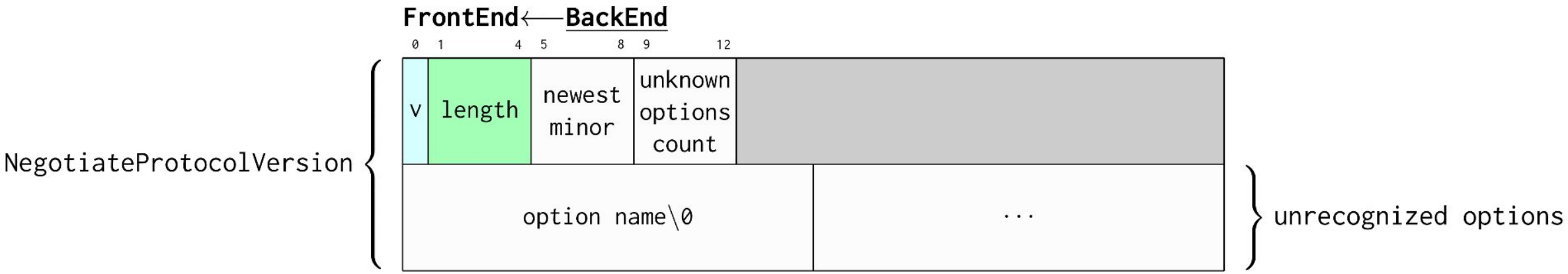
Startup Message

First message sent from the frontend is one of those



Startup Message

When backend doesn't support minor protocol version



Frontend should close the connection if it does not support this minor version.
With PG18 and V3.2 this could happen if frontend expects V3.2 and server only has V3.0

The frontend can usually continue if it can live without the unsupported options or stick to server version

Authentication

Method picked in pg_hba.conf, backend sends the Authentication* “challenge” message

- Messages exchanged vary for each method
- Final step: backend sends `Error` or `AuthenticationOk`

PG18 introduced OAUTH, deprecated MD5

SCRAM and OAUTH are recommended since PG18

you're (nearly) in!

©EDB 2026 — ALL RIGHTS RESERVED.

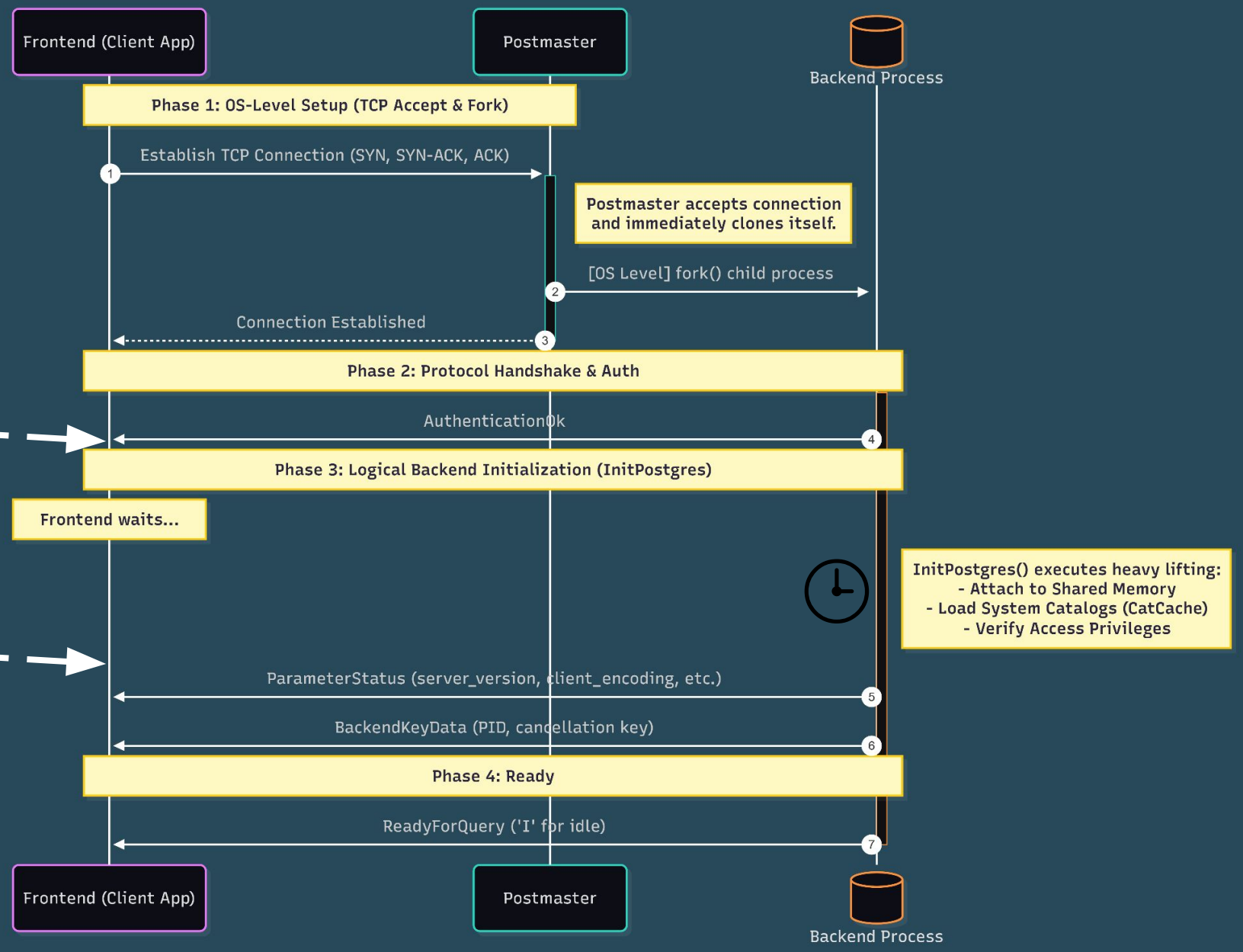
how is the **backend** process **created**?

©EDB 2026 — ALL RIGHTS RESERVED.

Backend spawn

You are here

We are going there



Server parameters

- ParameterStatus (a lot)
- Backend Key Data: PID + key

(needed by CancelRequest on a fresh new connection)

- **ReadyForQuery (🎉)**

FrontEnd⟵BackEnd

AuthenticationOK	R	length: 8	type: 0	
ParameterStatus	S	length: 23	in_hot_standby	off
ParameterStatus	S	length: 25	integer_datetimes	on
ParameterStatus	S	length: 26	TimeZone	Europe/Paris
ParameterStatus	S	length: 27	IntervalStyle	postgres
ParameterStatus	S	length: 32	search_path	"$user", public
ParameterStatus	S	length: 20	is_superuser	on
ParameterStatus	S	length: 26	application_name	psql
ParameterStatus	S	length: 38	default_transaction _read_only	off
ParameterStatus	S	length: 26	scram_iterations	4096
ParameterStatus	S	length: 23	DateStyle	ISO, MDY
ParameterStatus	S	length: 35	standard_conforming_string	on
ParameterStatus	S	length: 35	session_authorization	postgres
ParameterStatus	S	length: 28	client_encoding	WIN1252
ParameterStatus	S	length: 24	server_version	18.3
ParameterStatus	S	length: 25	server_encoding	UTF8
BackendKeyData	K	length: 12	PID 19480	Secret 6796488
ReadyForQuery	Z	length: 5	Idle	

Simple query explained

```
dvdrental=# select * from actor limit 10;
 actor_id | first_name |  last_name    |      last_update
----------+------------+---------------+-----------------------
        1 | Penelope   | Guiness       | 2013-05-26 14:47:57.62
        2 | Nick       | Wahlberg      | 2013-05-26 14:47:57.62
        3 | Ed         | Chase         | 2013-05-26 14:47:57.62
        4 | Jennifer   | Davis         | 2013-05-26 14:47:57.62
        5 | Johnny     | Lollobrigida  | 2013-05-26 14:47:57.62
        6 | Bette      | Nicholson     | 2013-05-26 14:47:57.62
```


©EDB 2026 — ALL RIGHTS RESERVED.

Simple query explained

```
# select * from actor limit 10;
```

psql client

Postgres backend

Q (Query) : select * from actor limit 10;

libpq knows: 1. query has no errors
2. what columns/types to expect

T (RowDescription)
Information about the incoming rows:
- column names,
- types,
- formats (binary/text)

R (DataRow): all values for a single row
R (DataRow)
R (DataRow)
R (DataRow)
R (DataRow)

libpq buffers all rows

C (CommandComplete) : I'm done with this batch

Z (ReadyForQuery) : I'm done with all batches and ready for subsequent work

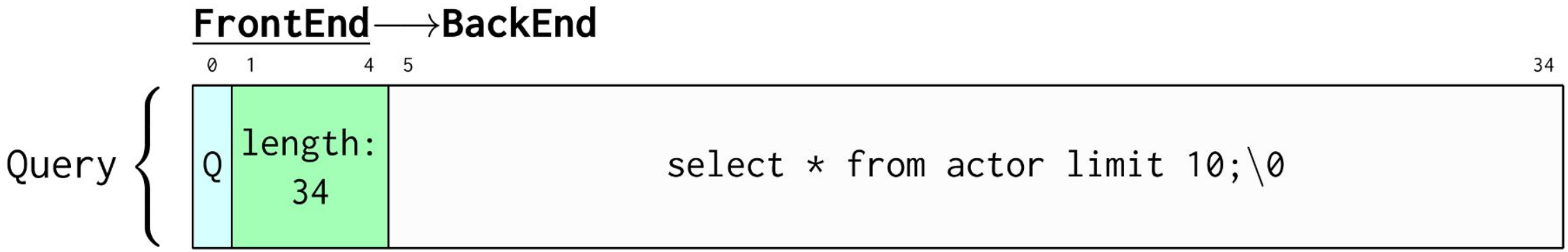
actor_id	first_name	last_name	last_update
1	Penelope	Guiness	2013-05-26 14:47:57.62
2	Nick	Wahlberg	2013-05-26 14:47:57.62
3	Ed	Chase	2013-05-26 14:47:57.62
4	Jennifer	Davis	2013-05-26 14:47:57.62
5	Johnny	Lollobrigida	2013-05-26 14:47:57.62
6	Bette	Nicholson	2013-05-26 14:47:57.62
7	Grace	Mostel	2013-05-26 14:47:57.62
8	Matthew	Johansson	2013-05-26 14:47:57.62
9	Joe	Swank	2013-05-26 14:47:57.62
10	Christian	Gable	2013-05-26 14:47:57.62

(10 lignes)

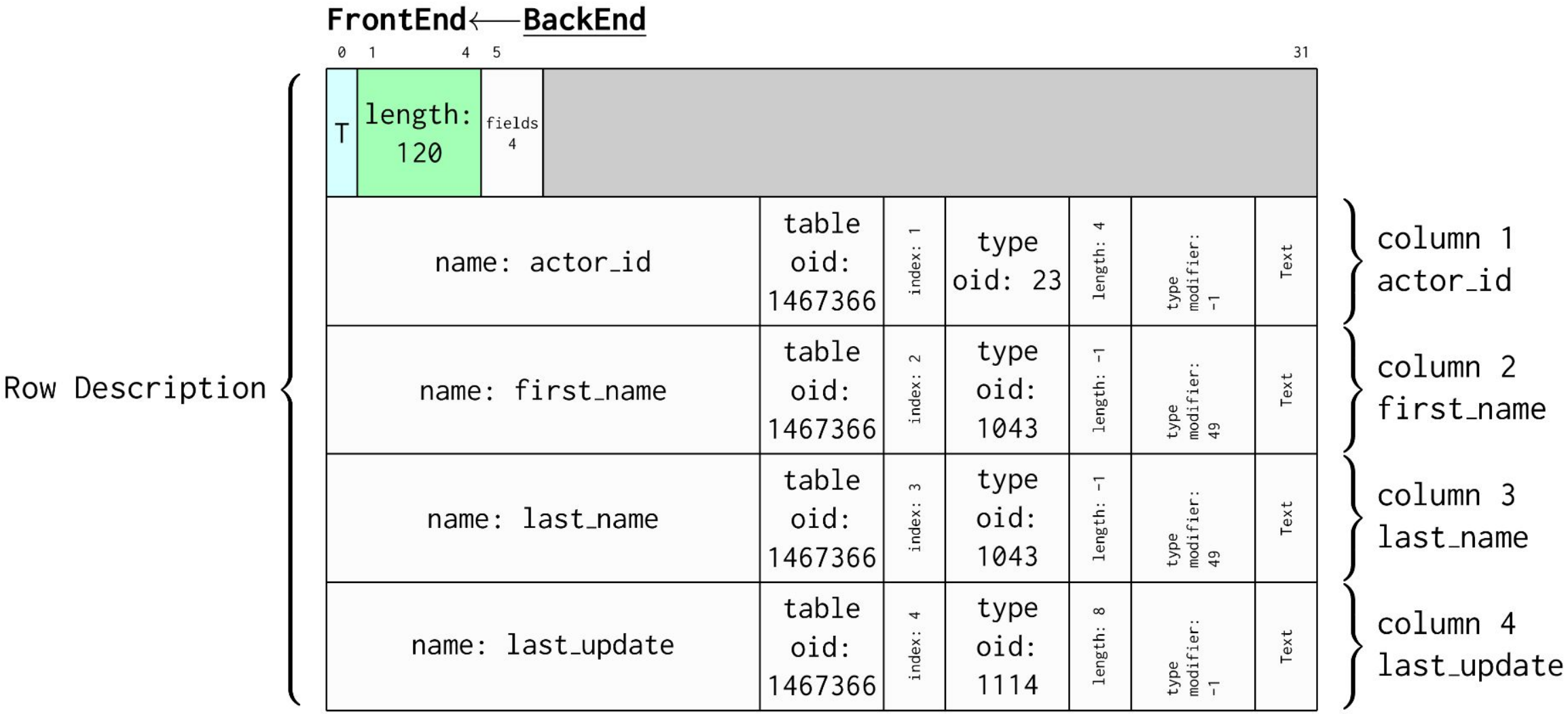
EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

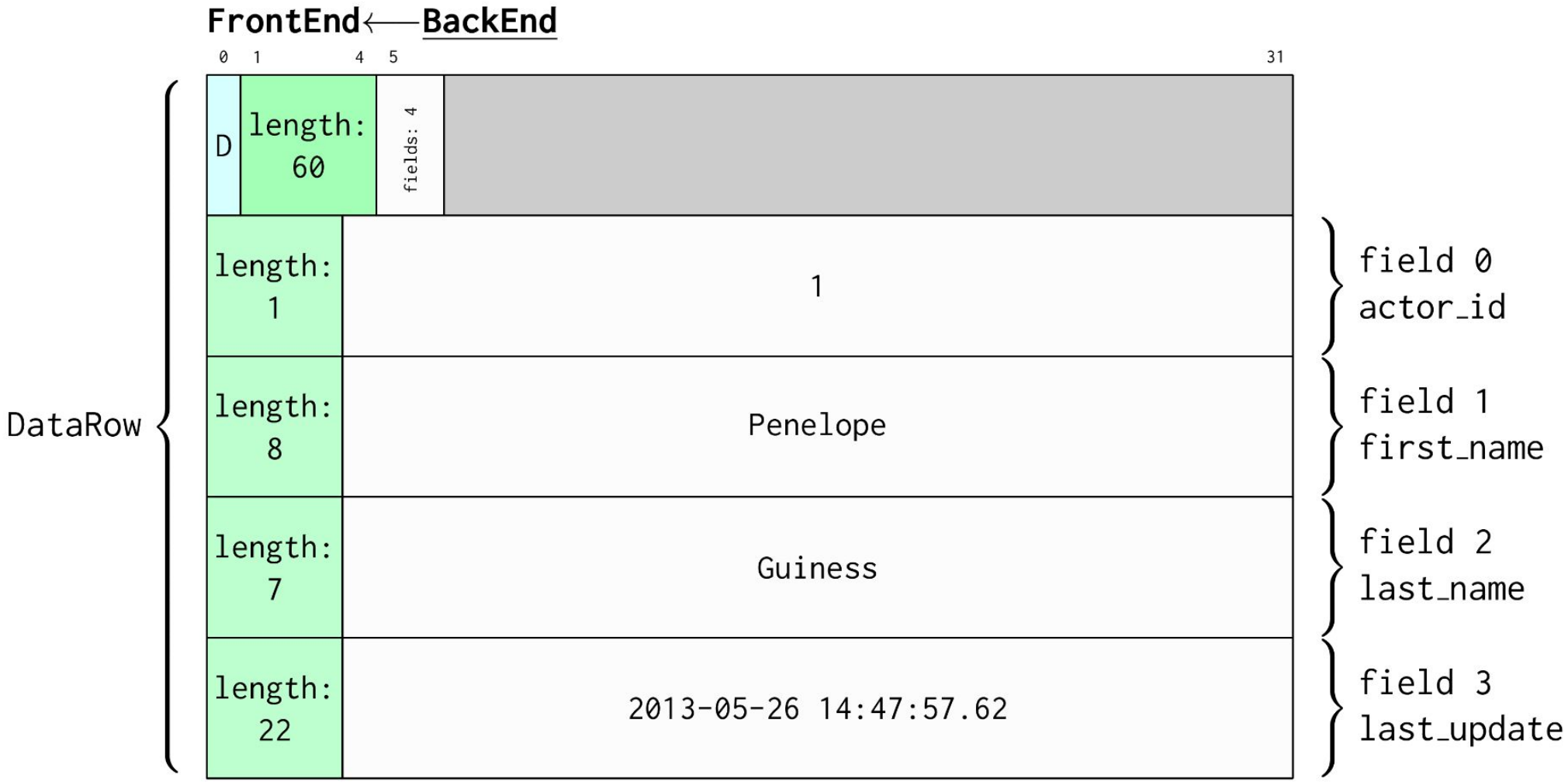
Simple query explained: >Q



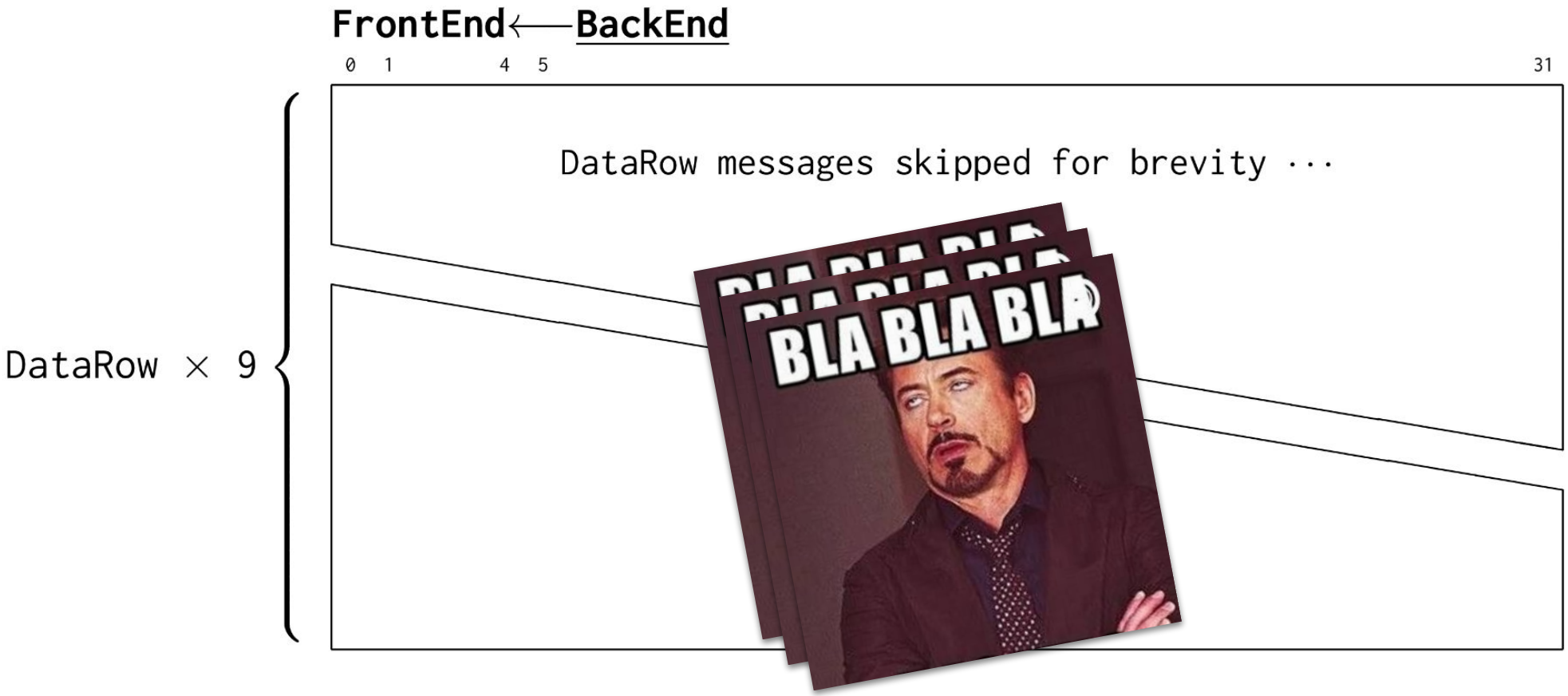
Simple query explained: >Q <T



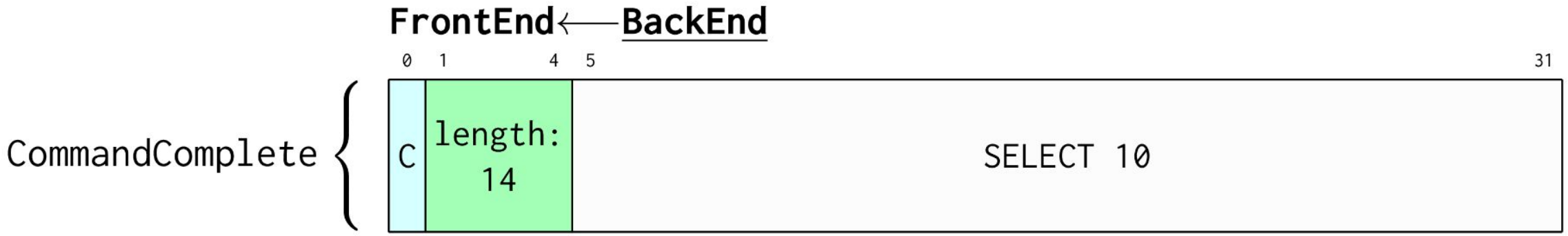
Simple query explained: >Q <T/D



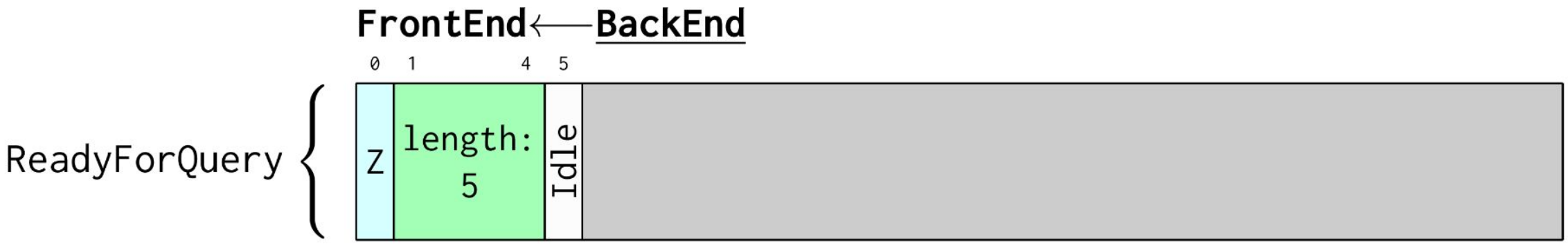
Simple query explained: >Q <T/D/……



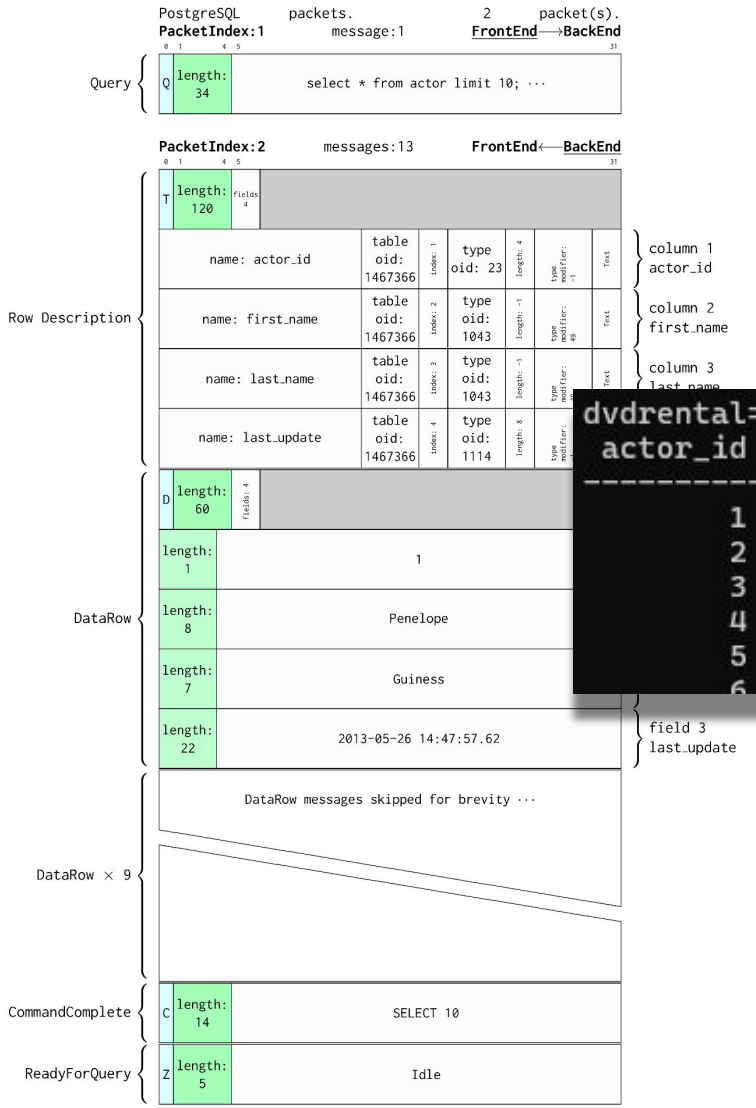
Simple query explained: >Q <T/D/D/D/D/D/D/D/D/D/D/C



Simple query explained: >Q <T/D/D/D/D/D/D/D/D/D/D/C/Z



Simple query explained: >Q <T/D/D/D/D/D/D/D/D/D/D/C/Z




```
dvdrental=# select * from actor limit 10;
 actor_id | first_name | last_name    |      last_update
----------+------------+--------------+-----------------------
        1 | Penelope   | Guiness      | 2013-05-26 14:47:57.62
        2 | Nick       | Wahlberg     | 2013-05-26 14:47:57.62
        3 | Ed         | Chase        | 2013-05-26 14:47:57.62
        4 | Jennifer   | Davis        | 2013-05-26 14:47:57.62
        5 | Johnny     | Lollobrigida | 2013-05-26 14:47:57.62
        6 | Bette      | Nicholson    | 2013-05-26 14:47:57.62
```

extended query

©EDB 2026 — ALL RIGHTS RESERVED.

Extended query: simple example

psql client

```
SELECT film_id, title, length FROM film WHERE rating = $1 AND length < $2
\bind 'PG-13' 80 \g
```

Postgres backend

P (parse) : query (unnamed statement)

B (bind) : “here are the parameters sent as text, I want data as text”

D (describe portal) : “tell me what is the shape of the data to expect”

E (execute)

S (sync) : I’m done

1 (parse complete): parsed OK

query parsed. Statement ready on BackEnd

2 (bind complete): params OK

parameters OK. planning done, portal ready on BackEnd
Custom plan generated (5-attempts before comparing with a generic plan)

T (RowDescription) result columns: names, types, formats all text)

R (DataRow)s or n (NoData)

C (CommandComplete) : I’m done with this batch

Z (ReadyForQuery) : I’m done with all batches and ready for subsequent work

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

Statements / portals

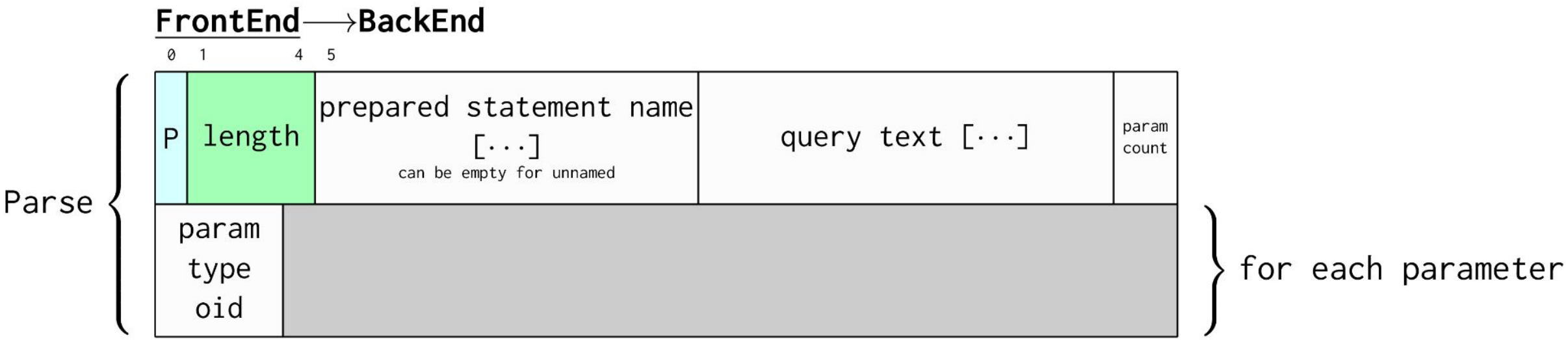
- **Statements** are a parameterized query “template” than can accept parameters. Created with a Parse message.
- **Portals** are created with Bind. It’s statements with parameters, planned and ready to be executed.

Parse → statement → Bind → portal → Execute

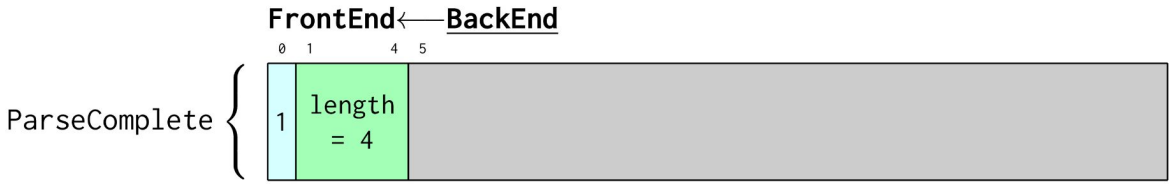
EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

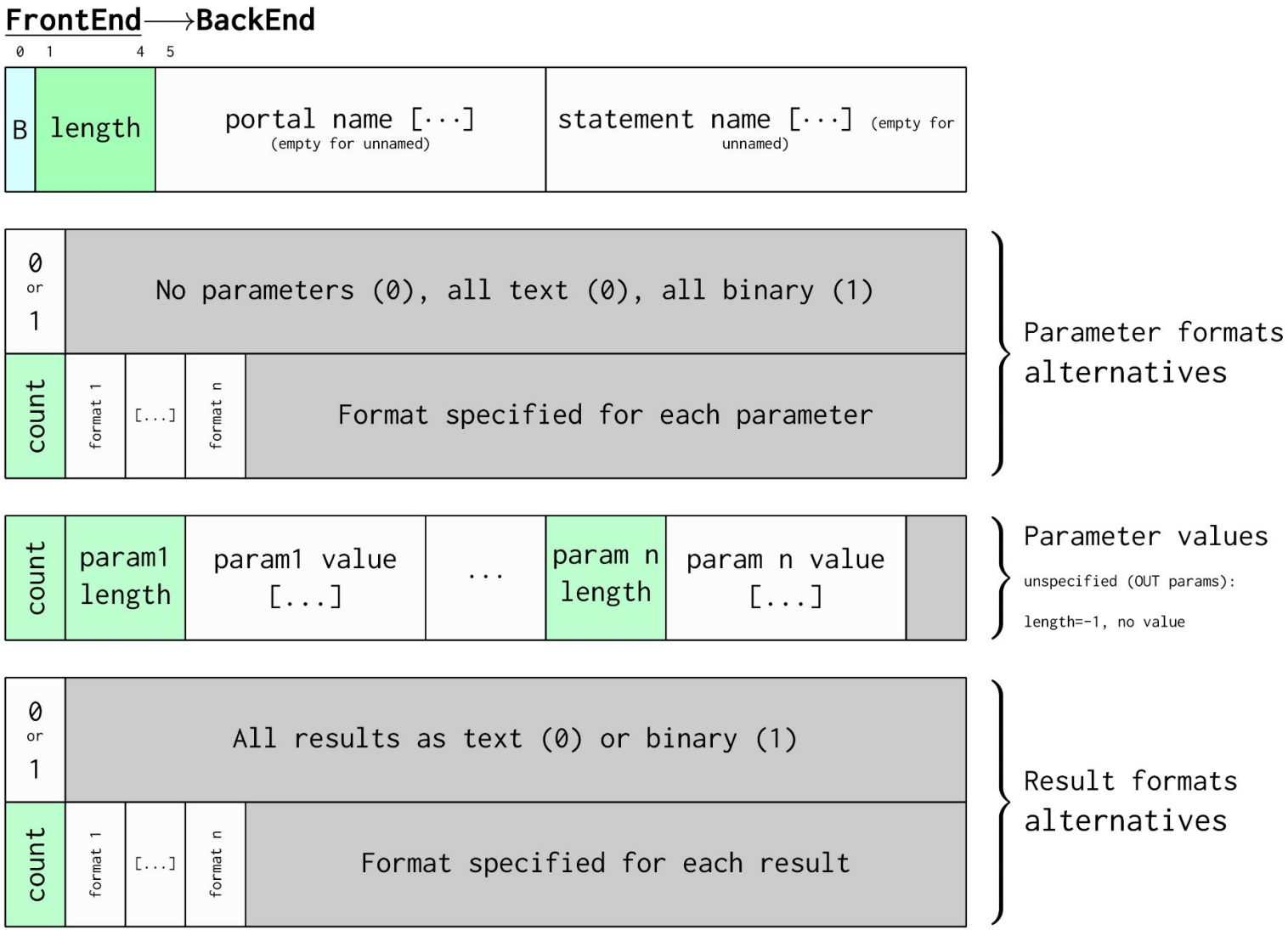
Extended query explained: Parse



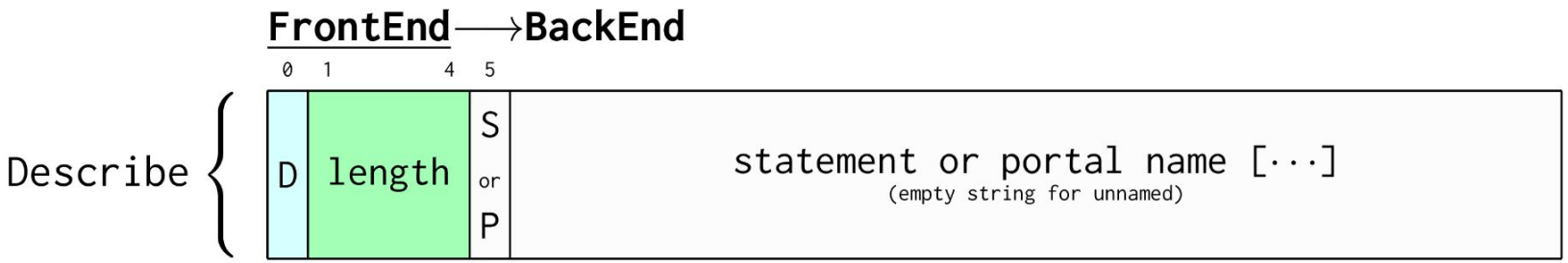
=> Parse complete expected



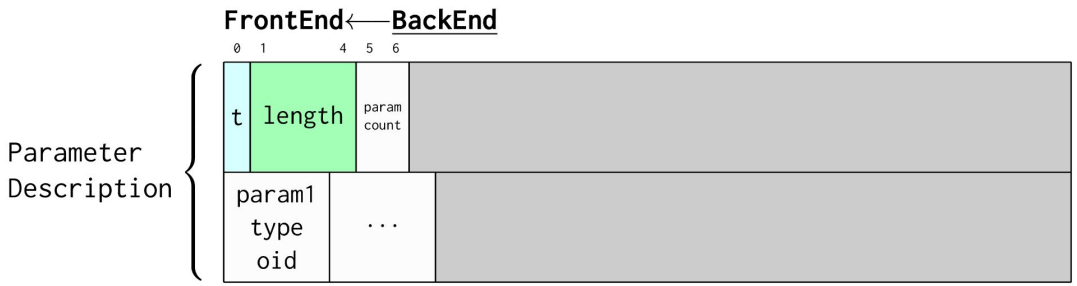
Extended query explained: `Bind`



Extended query explained: `Describe (statement/portal)`



=> Statement variant returns ParameterDescription, usually called before Bind



=> both variants return RowDescription

Extended query : prepared statement

Named statement
(in psql since PG18)

psql client

Postgres backend

```
SELECT film_id, title, length FROM film WHERE rating
= $1 AND length < $2
\parse film_search
```

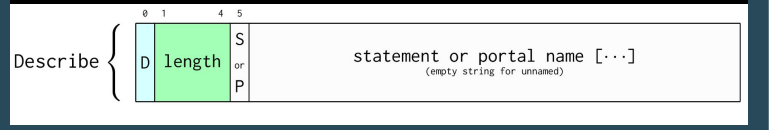
P (parse) : query (‘film_search’ named statement)

S (sync)

1 (parse complete)

Z (ReadyForQuery)

```
\bind_named film_search 'PG-13' 80
EXECUTE film_search;
```



B (Bind) film_search / D (Describe) portal / E (execute) / S (sync)

2 (BindComplete) / T (RowDescription) / R (DataRow[s]) / C (CommandComplete) / Z (ReadyForQuery)

```
\bind_named film_search 'PG-13' 120
EXECUTE film_search;
```

B (Bind) film_search / D (Describe) portal / E (execute) / S (sync)

2 (BindComplete) / T (RowDescription) / R (DataRow[s]) / C (CommandComplete) / Z (ReadyForQuery)

...

```
\close_prepared film_search
```

C (Close) film_search / S (Sync)

3 (CloseComplete) / Z (ReadyForQuery)

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

Extended query : cursors/portals

Producer/Consumer with slow consumer

psql client

Postgres backend

P (parse) ‘film_search’ / D (Describe) S ‘film_search’ / S (Sync)

1 (ParseComplete) / t (ParameterDescription) / D (RowDescription) / Z

B (Bind) values + film_search + **film_portal** / E (execute) film_portal 100 rows / S (sync)

2 (BindComplete) / R (100 DataRow[s]) / s (PortalSuspended) / Z (ReadyForQuery)

Slow processing…

E (execute) film_portal 100 rows / S (sync)

R (100 DataRow[s]) / s (PortalSuspended) / Z (ReadyForQuery)

…

…

R (20 DataRow[s]) / C (CommandComplete)

C (Close) film_portal / S (sync)

3 (CloseComplete) / Z (ReadyForQuery)

220 rows in total

©EDB 2026 — ALL RIGHTS RESERVED.

Extended query / wrap up

Pros

- Prevents SQL injection
- Binary data support
- Plan reuse with prepared statements
- Many ways to improve performance

Cons

- Slower for one-off queries
- Single statement only

©EDB 2026 — ALL RIGHTS RESERVED.

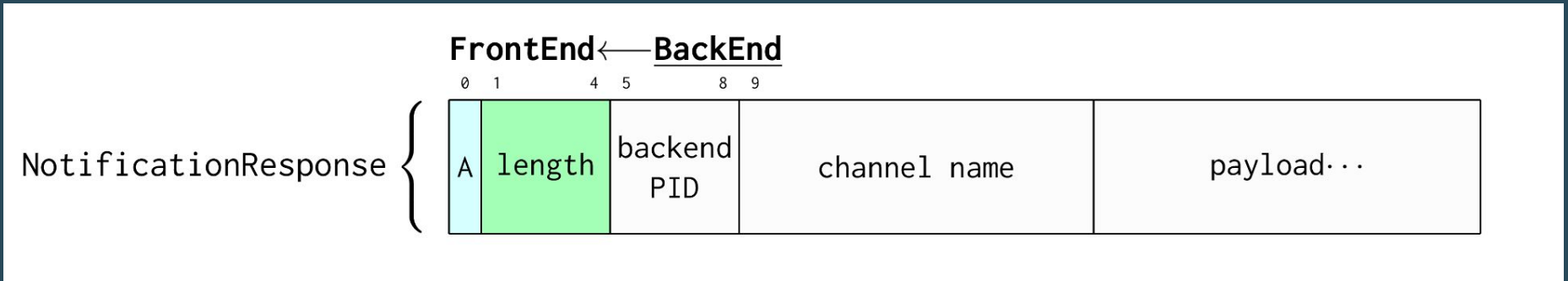
server-pushed messages

©EDB 2026 — ALL RIGHTS RESERVED.

Listen / notify

Subscription model

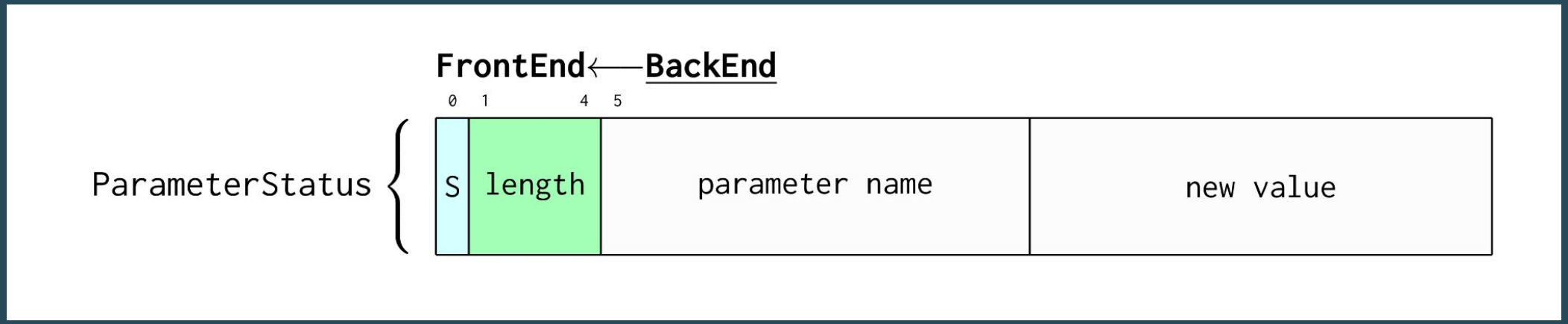
- Any frontend subscribes via simple query *LISTEN <channel name>*
- Any backend can *NOTIFY <channel name> payload*
- Listeners receive a NotificationResponse message when connection is idle



©EDB 2026 — ALL RIGHTS RESERVED.

GUC changes

- Notifies frontends when any GUC sent after Startup changes
- Backend sends *ParameterStatus* message before the next *ReadyForQuery* for each changed GUC
- Needed for drivers to interpret wire data correctly and consistently



©EDB 2026 — ALL RIGHTS RESERVED.

copy

©EDB 2026 — ALL RIGHTS RESERVED.

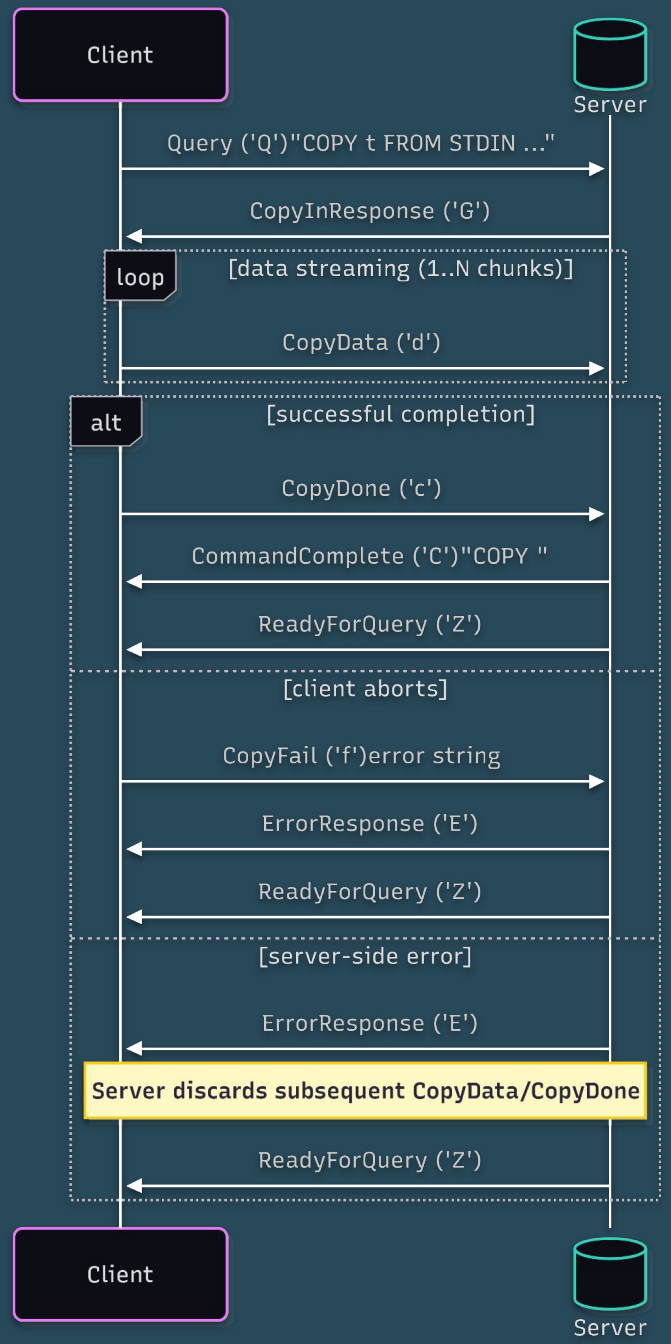
Bulk import/export

Copy is a sub protocol

Formats

- Text (default)
- CSV
- Binary (10-100x faster than INSERTs)

Used by shp2pgsql, pg_dump, pglogical



EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

replication

©EDB 2026 — ALL RIGHTS RESERVED.

Replication

Initiated with the **`replication`** parameter in the Startup packet.

It's a special mode of the protocol with its own grammar

(IDENTIFY_SYSTEM, START_REPLICATION…)

- **`replication=true`** physical replication
- **`replication=<database>`** logical replication

©EDB 2026 — ALL RIGHTS RESERVED.

Replication

Physical replication works only with same:

- server major version
- Same on-disk binary format

Logical replication is “cross platform”

- Does not replicate DDL (functions, procedures, catalog)

©EDB 2026 — ALL RIGHTS RESERVED.

Replication

Great talks from PG Conf EU 2023

“The journey towards active-active replication in PostgreSQL” by Jonathan S. Katz

https://www.postgresql.eu/events/pgconfeu2023/schedule/session/4783-the-journey-towards-active-active-replication-in-postgresql/

©EDB 2026 — ALL RIGHTS RESERVED.

performance

©EDB 2026 — ALL RIGHTS RESERVED.

Performance considerations : connections

Connections are expensive (5 to 10MB, 5-50ms to open)

- use pooling (implemented in some drivers, or with pgBouncer, pgPool, ...)

©EDB 2026 — ALL RIGHTS RESERVED.

Performance considerations : queries

Repetitive parameterized queries benefit from prepared statements

- use binary format
- 5-exections_trap: make sure data is well distributed with param variation; Postgres may switch to a generic plan whose selectivity assumptions are wrong for skewed data

©EDB 2026 — ALL RIGHTS RESERVED.

Performance considerations : quick-wins

Avoid N+1 problem (fetch N rows, then perform a query per row)

- Postgres will do it better than you!

Avoid unnecessary roundtrips

- Use batches (INSERT, arrays)

Trust Postgres, trust the drivers, measure, and read the docs!

©EDB 2026 — ALL RIGHTS RESERVED.

diagnosing

©EDB 2026 — ALL RIGHTS RESERVED.

libpq has PQtrace()

If you use/wrap libpq, you can activate protocol tracing (improved in PG14)

```
PQtrace(conn, trace_file);
    void PQtrace(PGconn *conn, FILE *debug_port)
    === in fe-trace.c ===
    Describe
```

```
F       33      Query   "select * from actor limit 10"
B       120     RowDescription  4 "actor_id" 1467366 1 23 4 -1 0 "first_name" 1467366 2 1043 65535 49 0 "last_name"
1467366 3 1043 65535 49 0 "last_update" 1467366 4 1114 8 -1 0
B       60      DataRow 4 1 '1' 8 'Penelope' 7 'Guiness' 22 '2013-05-26 14:47:57.62'
B       57      DataRow 4 1 '2' 4 'Nick' 8 'Wahlberg' 22 '2013-05-26 14:47:57.62'
B       52      DataRow 4 1 '3' 2 'Ed' 5 'Chase' 22 '2013-05-26 14:47:57.62'
B       58      DataRow 4 1 '4' 8 'Jennifer' 5 'Davis' 22 '2013-05-26 14:47:57.62'
B       63      DataRow 4 1 '5' 6 'Johnny' 12 'Lollobrigida' 22 '2013-05-26 14:47:57.62'
B       59      DataRow 4 1 '6' 5 'Bette' 9 'Nicholson' 22 '2013-05-26 14:47:57.62'
B       56      DataRow 4 1 '7' 5 'Grace' 6 'Mostel' 22 '2013-05-26 14:47:57.62'
B       61      DataRow 4 1 '8' 7 'Matthew' 9 'Johansson' 22 '2013-05-26 14:47:57.62'
B       53      DataRow 4 1 '9' 3 'Joe' 5 'Swank' 22 '2013-05-26 14:47:57.62'
B       60      DataRow 4 2 '10' 9 'Christian' 5 'Gable' 22 '2013-05-26 14:47:57.62'
B       14      CommandComplete         "SELECT 10"
B       5       ReadyForQuery   I
F       59      Parse   "actor_by_id" "select * from actor where actor_id  = $1" 0
F       4       Sync
B       4       ParseComplete
B       5       ReadyForQuery   I
F       30      Bind    "" "actor_by_id" 0 1 1 '1' 1 1
F       6       Describe        P ""
F       9       Execute "" 0
F       4       Sync
B       4       BindComplete
B       120     RowDescription  4 "actor_id" 1467366 1 23 4 -1 1 "first_name" 1467366 2 1043 65535 49 1 "last_name"
1467366 3 1043 65535 49 1 "last_update" 1467366 4 1114 8 -1 1
B       49      DataRow 4 4 '\x00\x00\x00\x01' 8 'Penelope' 7 'Guiness' 8 '\x00\x01\x80\x9e\xe2\xf57 '
```

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

diagnosing issues with wireshark

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

wireshark allows to trace network messages

wireshark dissectors allows presenting the messages in a **understandable way** for a given **protocol**

EDB POSTGRES AI

©EDB 2026 — ALL RIGHTS RESERVED.

The Wireshark Network Analyzer

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

Welcome to Wireshark

Capture

...using this filter: Enter a capture filter ...

8 interfaces shown, 4 hidden

vEthernet (Default Switch)

Wi-Fi

Adapter for loopback traffic capture

Local Area Connection* 12

Local Area Connection* 10

Local Area Connection* 9

Local Area Connection* 13

Local Area Connection* 3

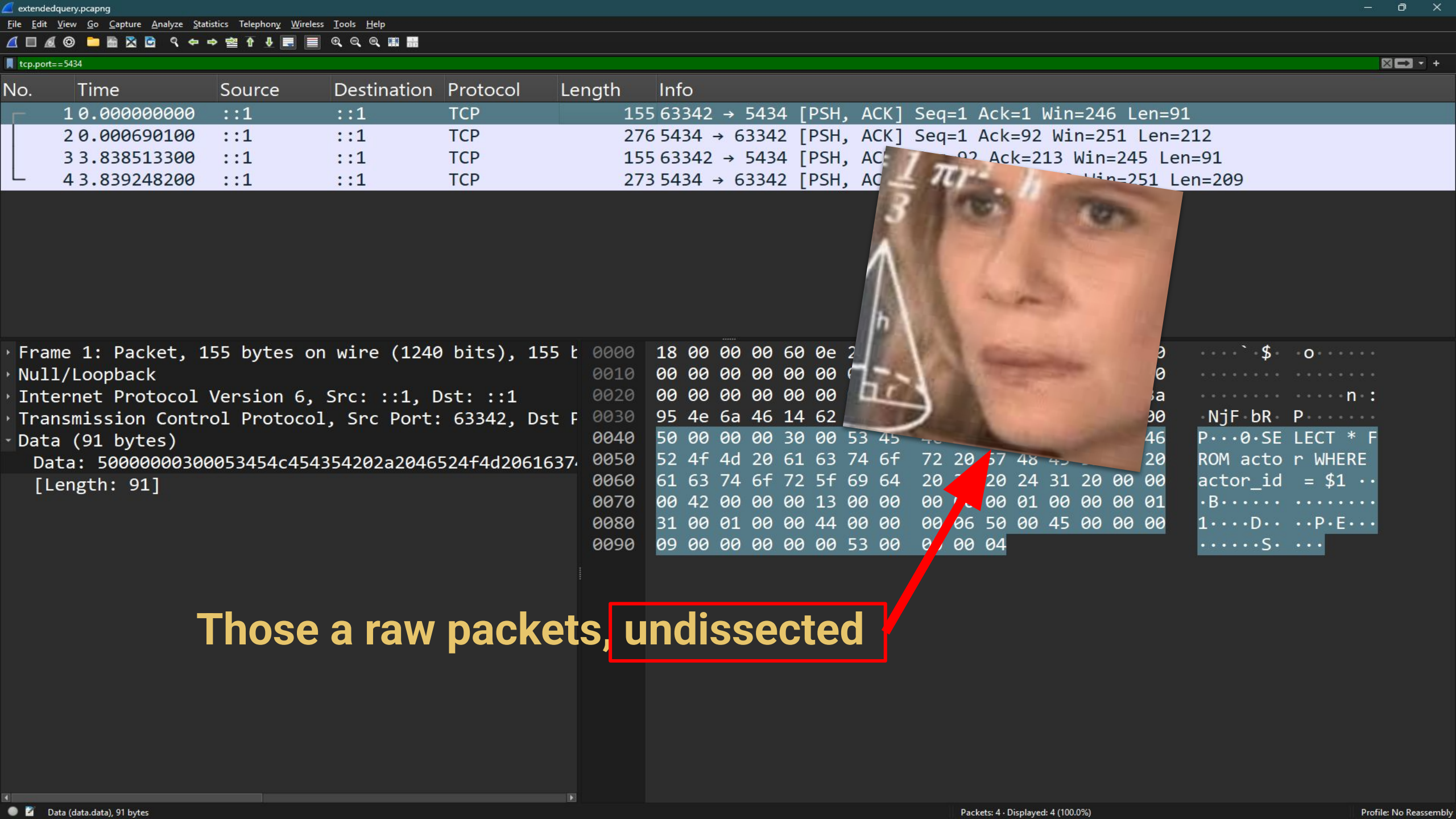
Choose a network interface to capture

Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

You are running Wireshark 4.6.4 (v4.6.4-0-g93282876538d). You receive automatic updates.

Ready to load or capture | No Packets | Profile: PG AND PGSQLEDB

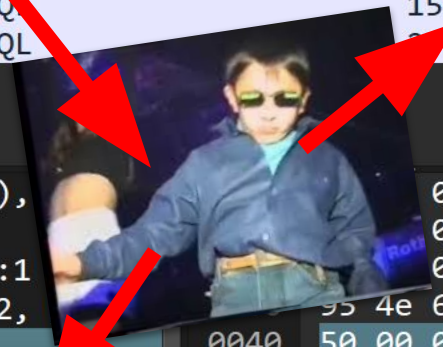

extendedquery.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==5434
No. Time Source Destination Protocol Length Info
1 0.000000000 ::1 ::1 TCP 155 63342 → 5434 [PSH, ACK] Seq=1 Ack=1 Win=246 Len=91
2 0.000690100 ::1 ::1 TCP 276 5434 → 63342 [PSH, ACK] Seq=1 Ack=92 Win=251 Len=212
3 3.838513300 ::1 ::1 TCP 155 63342 → 5434 [PSH, AC
4 3.839248200 ::1 ::1 TCP 273 5434 → 63342 [PSH, AC
Frame 1: Packet, 155 bytes on wire (1240 bits), 155 b
Null/Loopback
Internet Protocol Version 6, Src: ::1, Dst: ::1
Transmission Control Protocol, Src Port: 63342, Dst P
Data (91 bytes)
Data: 50000000300053454c454354202a2046524f4d2061637
[Length: 91]
Those a raw packets, undissected
Data (data.data), 91 bytes
Packets: 4 · Displayed: 4 (100.0%)
Profile: No Reassembly

extendedquery.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

pgsql

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000…	63342	5434	PGSQL	155	>P/B/D/E/S
2	0.000…	5434	63342	PGSQL	276	<1/2/T/D/C/Z
3	3.838…	63342	5434	PGSQ	155	>P/B/D/E/S
4	3.839…	5434	63342	PGSQL	3	<1/2/T/D/C/Z

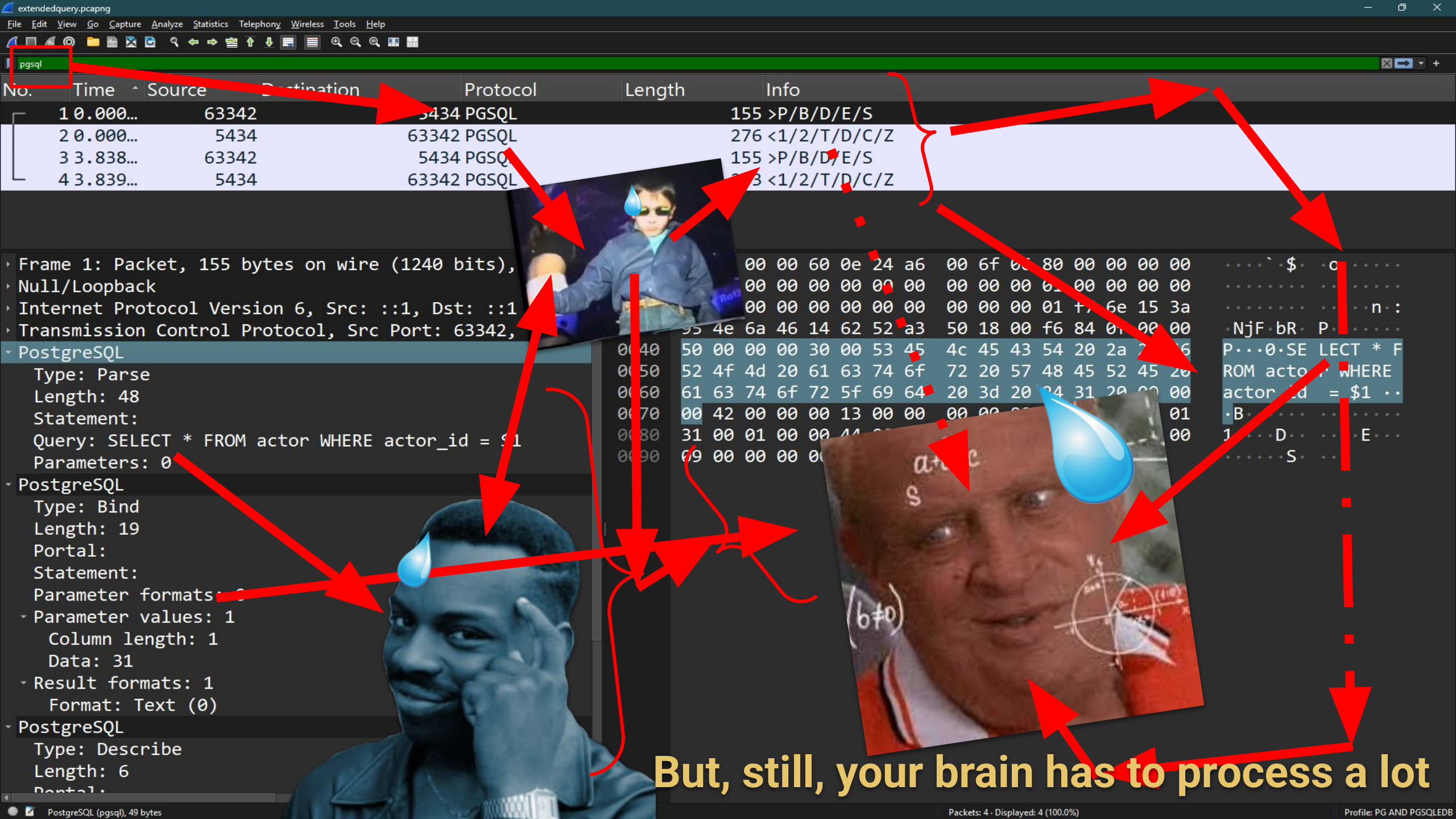


```
Frame 1: Packet, 155 bytes on wire (1240 bits),
Null/Loopback
Internet Protocol Version 6, Src: ::1, Dst: ::1
Transmission Control Protocol, Src Port: 63342,
PostgreSQL
  Type: Parse
  Length: 48
  Statement:
  Query: SELECT * FROM actor WHERE actor_id = $1
  Parameters: 0
PostgreSQL
  Type: Bind
  Length: 19
  Portal:
  Statement:
  Parameter formats: 0
  Parameter values: 1
    Column length: 1
    Data: 31
  Result formats: 1
    Format: Text (0)
PostgreSQL
  Type: Describe
  Length: 6
  Portal:
```

```
                00 00 60 0e 24 a6  00 6f 06 80 00 00 00 00   ····`·$· ·o····· 
                00 00 00 00 00 00  00 00 00 01 00 00 00 00   ···· ···· ····· 
                00 00 00 00 00 00  00 00 00 01 f7 6e 15 3a   ···· ···· ····n·: 
                95 4e 6a 46 14 62  52 a3 50 18 00 f6 84 0f   ·NjF·bR· P····· 
0040            50 00 00 00 30 00  53 45 4c 45 43 54 20 2a   P···0·SE LECT * 
0050            52 4f 4d 20 61 63  74 6f 72 20 57 48 45 52   ROM acto r WHER 
0060            61 63 74 6f 72 5f  69 64 20 3d 20 24 31 20   actor_id = $1 · 
0070            00 42 00 00 00 13  00 00 00 00 00 01 00 00   ·B····· ····· 
0080            31 00 01 00 00 44  00 00 00 06 50 00 45 00   1····D·· ··P·E· 
0090            09 00 00 00 00 00  53 00 00 00 04            ····· S··· 
```

Wireshark has a built in PostgreSQL dissector

PostgreSQL (pgsql), 49 bytes | Packets: 4 · Displayed: 4 (100.0%) | Profile: PG AND PGSQLEDB

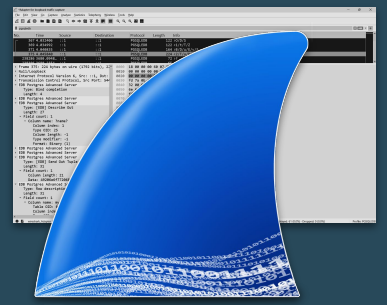

extendedquery.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
pgsql
No. Time Source Destination Protocol Length Info
1 0.000… 63342 5434 PGSQL 155 >P/B/D/E/S
2 0.000… 5434 63342 PGSQL 276 <1/2/T/D/C/Z
3 3.838… 63342 5434 PGSQL 155 >P/B/D/E/S
4 3.839… 5434 63342 PGSQL 3 <1/2/T/D/C/Z
Frame 1: Packet, 155 bytes on wire (1240 bits),
Null/Loopback
Internet Protocol Version 6, Src: ::1, Dst: ::1
Transmission Control Protocol, Src Port: 63342,
PostgreSQL
Type: Parse
Length: 48
Statement:
Query: SELECT * FROM actor WHERE actor_id = $1
Parameters: 0
PostgreSQL
Type: Bind
Length: 19
Portal:
Statement:
Parameter formats: 0
Parameter values: 1
Column length: 1
Data: 31
Result formats: 1
Format: Text (0)
PostgreSQL
Type: Describe
Length: 6
PostgreSQL (pgsql), 49 bytes
Packets: 4 · Displayed: 4 (100.0%)
Profile: PG AND PGSQLEDB
But, still, your brain has to process a lot

here comes

pg_protoexport

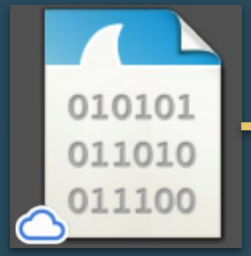
an open source tool do **document** the **protocol**

©EDB 2026 — ALL RIGHTS RESERVED.

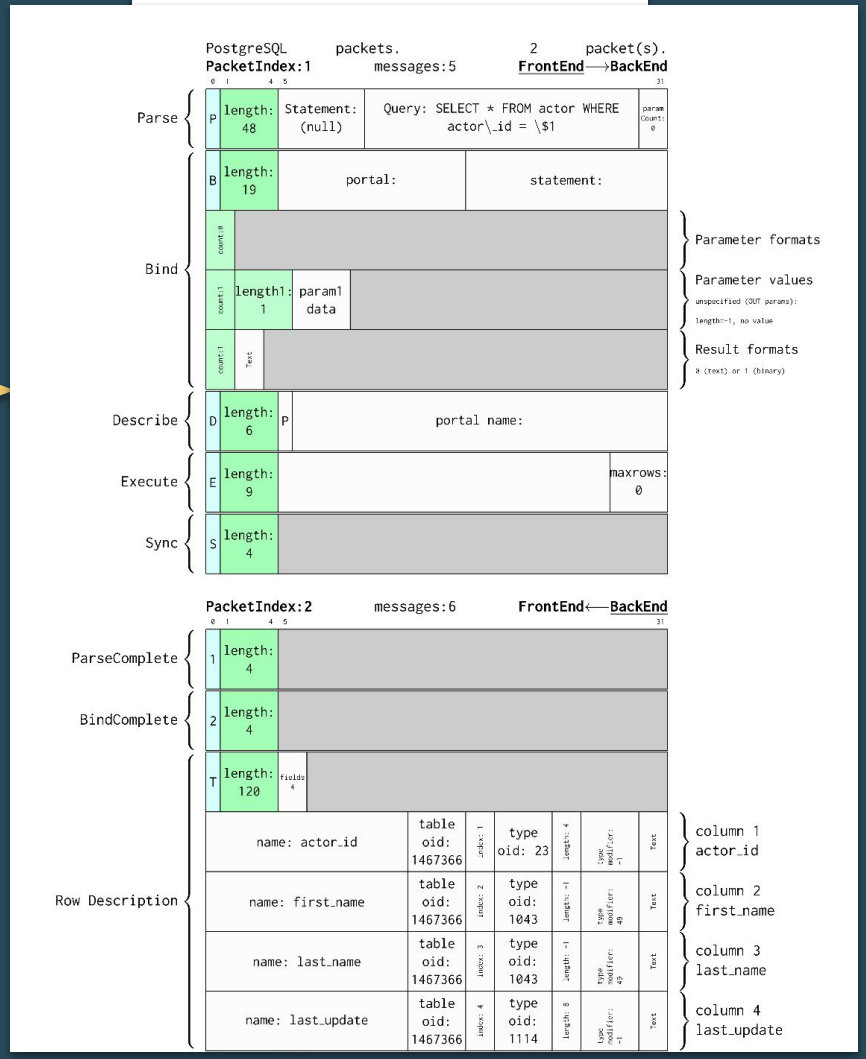


Wireshark

capture file



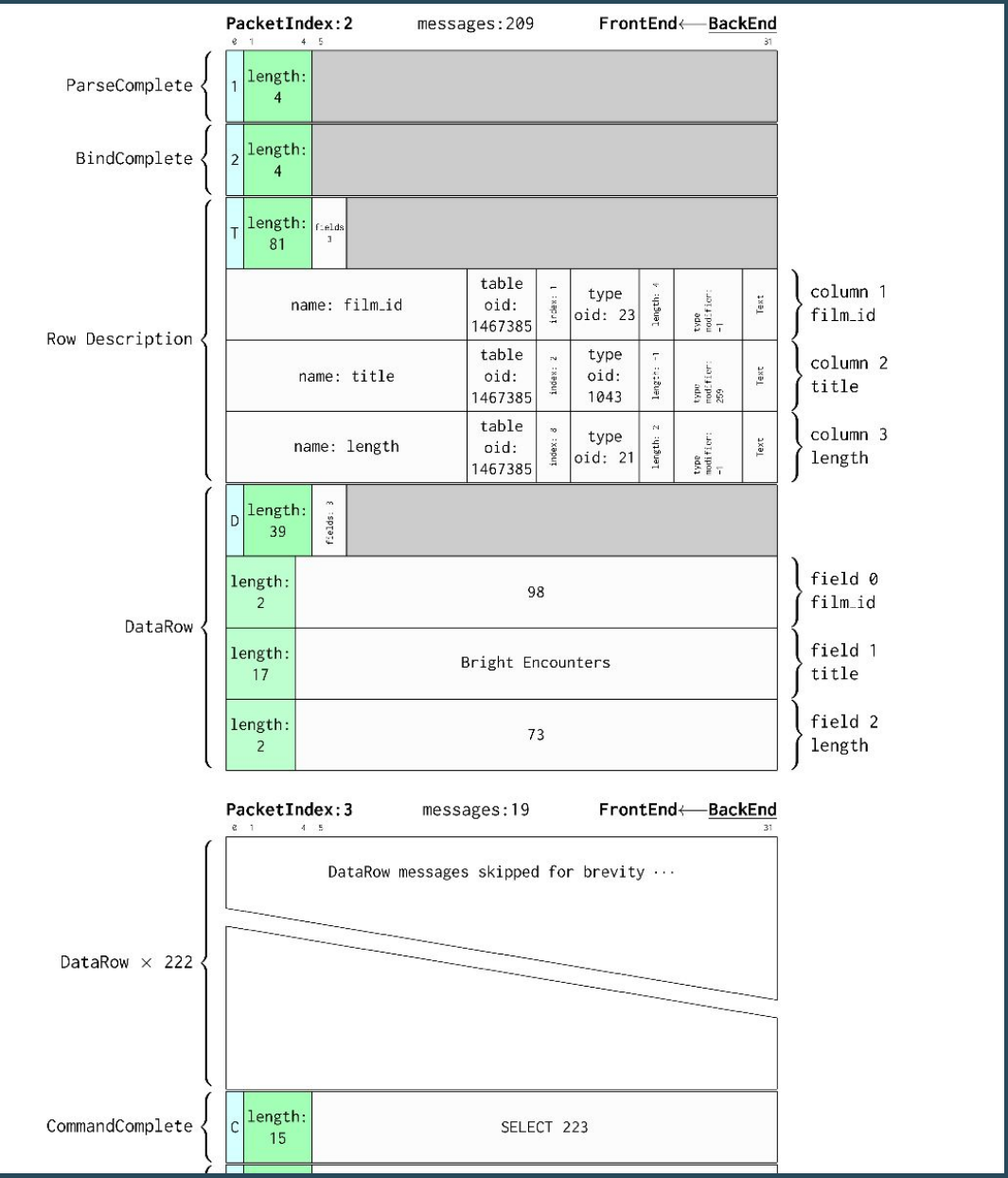
pg_protoexport



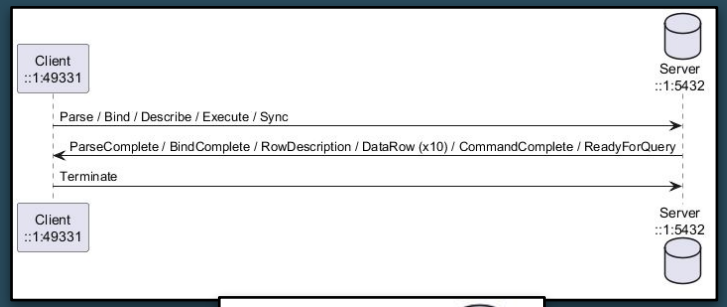
EDB POSTGRES AI

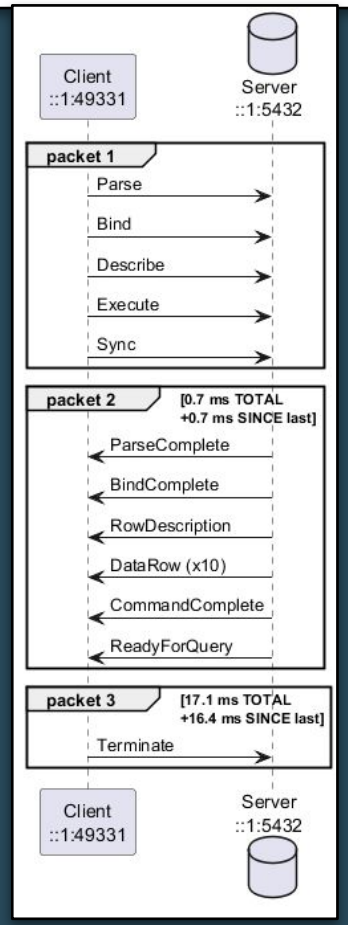
©EDB 2026 — ALL RIGHTS RESERVED.

LaTeX



Mermaid



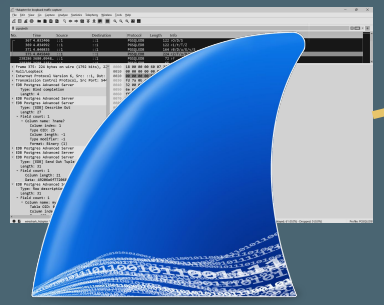


Text

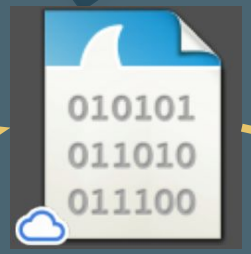
```
1  Packet (F)
2  2025-03-13T06:36:30.1436940Z   F   83  Parse   statement=, query=select oid, typname, typtype from
   pg_type where typtype <> $1 limit 10;, parameter oids=[25]
3  2025-03-13T06:36:30.1436940Z   F   19  Bind    : formats: [], values len: [1], results format: [1]
4  2025-03-13T06:36:30.1436940Z   F   6   Describe    Portal: ''
5  2025-03-13T06:36:30.1436940Z   F   9   Execute portal='', maxrows=0
6  2025-03-13T06:36:30.1436940Z   F   4   Sync    SyncMessage
7  Packet (B)
8  2025-03-13T06:36:30.1444240Z   B   4   ParseComplete   ParseCompleteMessage
9  2025-03-13T06:36:30.1444240Z   B   4   BindComplete
10 2025-03-13T06:36:30.1444240Z   B   80  RowDescription  [oid: 1247, 1, 26, 4, -1, 1, typname: 1247, 2, 19,
   64, -1, 1, typtype: 1247, 7, 18, 1, -1, 1]
11 2025-03-13T06:36:30.1444240Z   B   30  DataRow [oid:00000047, typname:pg_type, typtype:c]
12 2025-03-13T06:36:30.1444240Z   B   35  DataRow [oid:0000004b, typname:pg_attribute, typtype:c]
13 2025-03-13T06:36:30.1444240Z   B   30  DataRow [oid:00000051, typname:pg_proc, typtype:c]
14 2025-03-13T06:36:30.1444240Z   B   31  DataRow [oid:00000053, typname:pg_class, typtype:c]
15 2025-03-13T06:36:30.1444240Z   B   37  DataRow [oid:00000020, typname:pg_ddl_command, typtype:p]
16 2025-03-13T06:36:30.1444240Z   B   30  DataRow [oid:000002c1, typname:unknown, typtype:p]
17 2025-03-13T06:36:30.1444240Z   B   32  DataRow [oid:00000f40, typname:int4range, typtype:r]
18 2025-03-13T06:36:30.1444240Z   B   31  DataRow [oid:00000f42, typname:numrange, typtype:r]
19 2025-03-13T06:36:30.1444240Z   B   30  DataRow [oid:00000f44, typname:tsrange, typtype:r]
20 2025-03-13T06:36:30.1444240Z   B   32  DataRow [oid:00000f46, typname:tstzrange, typtype:r]
21 2025-03-13T06:36:30.1444240Z   B   14  CommandComplete SELECT 10
22 2025-03-13T06:36:30.1444240Z   B   5   ReadyForQuery   Idle
23 Packet (F)
24 2025-03-13T06:36:30.1607910Z   F   4   Terminate   TerminateMessage
25
```

whatever

(like, even PostgreSQL)



Wireshark



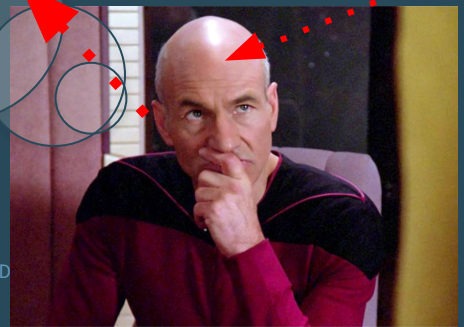
capture file

pg_protoexport

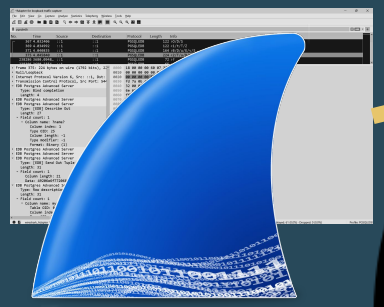
psql

whatever

(like, even PostgreSQL or spreadsheet)



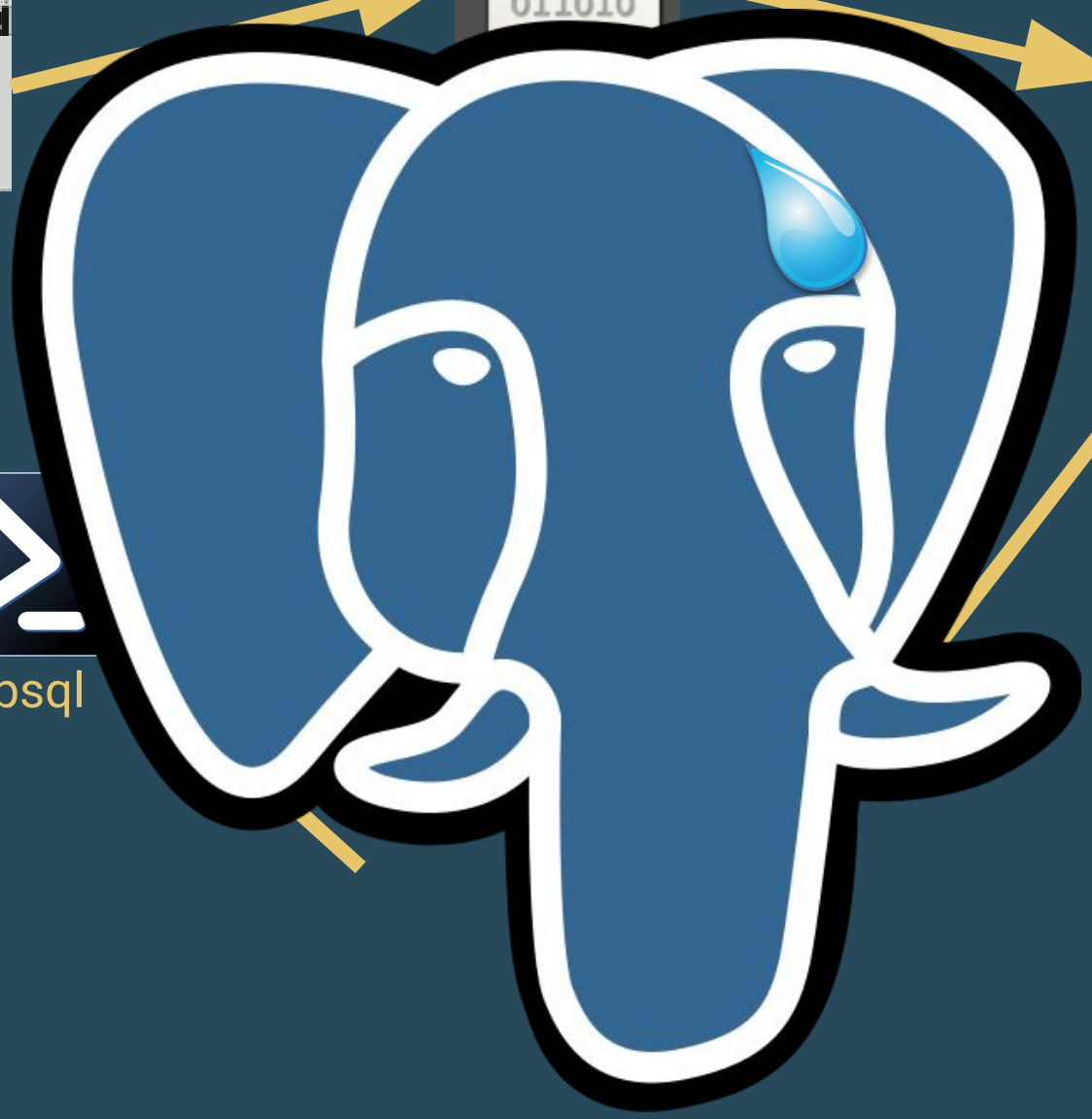
©ED



Wireshark

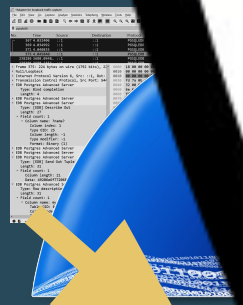
pg_protoexport

psql



whatever

PostgreSQL



EDB POSTGRES AI

pg_protoexport

(TBA)

Thank you!

Bedankt!

xavier.fischer@enterprisedb.com

fischerxavier

@edbpostgres.bsky.social

©EDB 2026 — ALL RIGHTS RESERVED.

EDB POSTGRES AI