Golang 的熱門 PostgreSQL Library 比較

透過觀察封包比較實作上的差異

Ruian Huang @Golang Taipei 2020/06/23

Ruian Huang

Dcard 工程師

- 交大資工畢業
- 偶爾寫點文章
  https://medium.com/@ruian

包含 PG Wire Protocol 與
Query Optimizer 相關文章

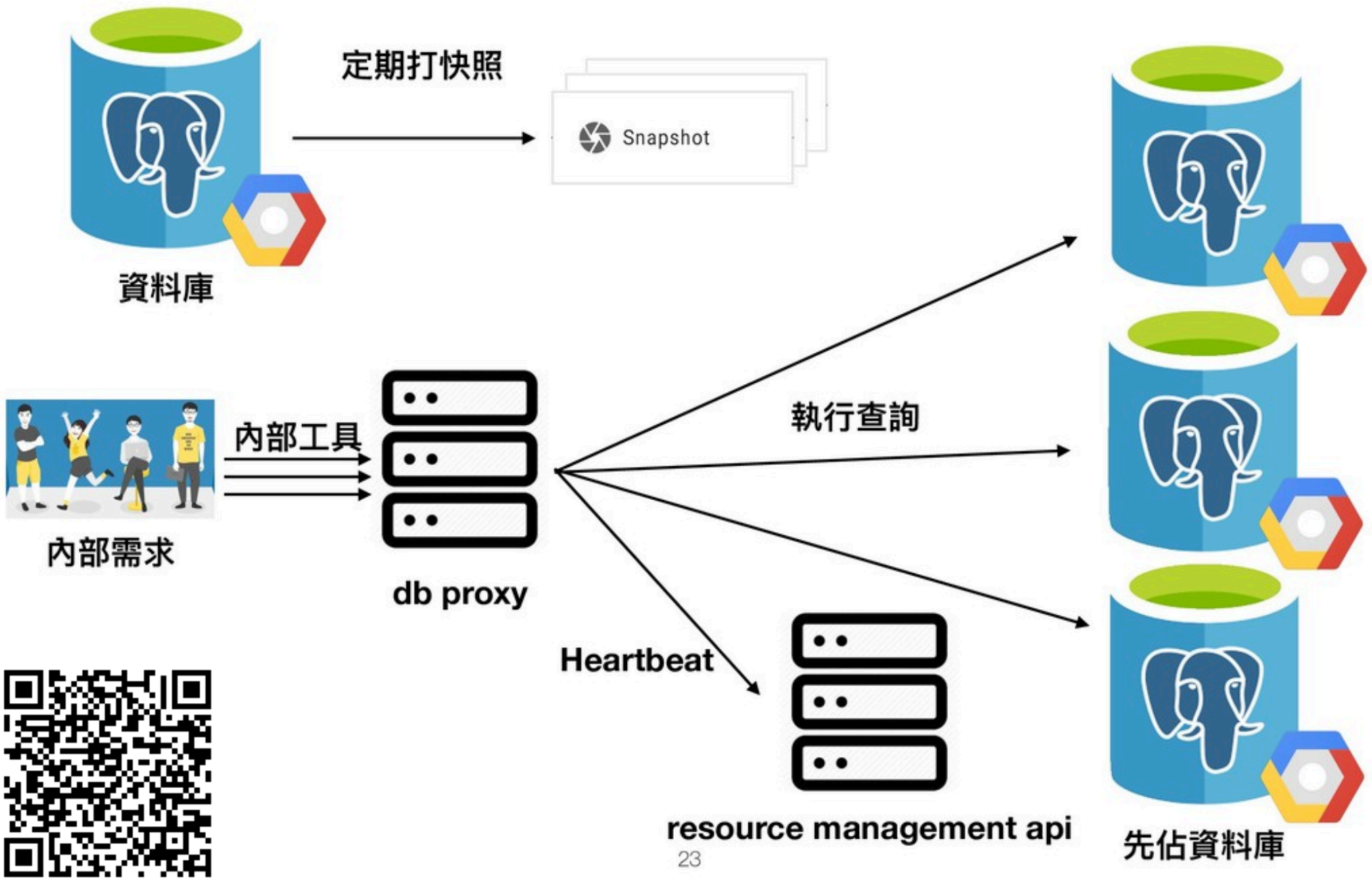
回顧去年同樣於 Golang Taipei 分享的主題

在 GCP 上，對 OLTP 資料庫擴展分析型查詢

Ruian Huang @ Dcard Plaza / Golang Taipei Gathering #42. 2019-06-18

1

回顧去年同樣於 Golang Taipei 分享的主題



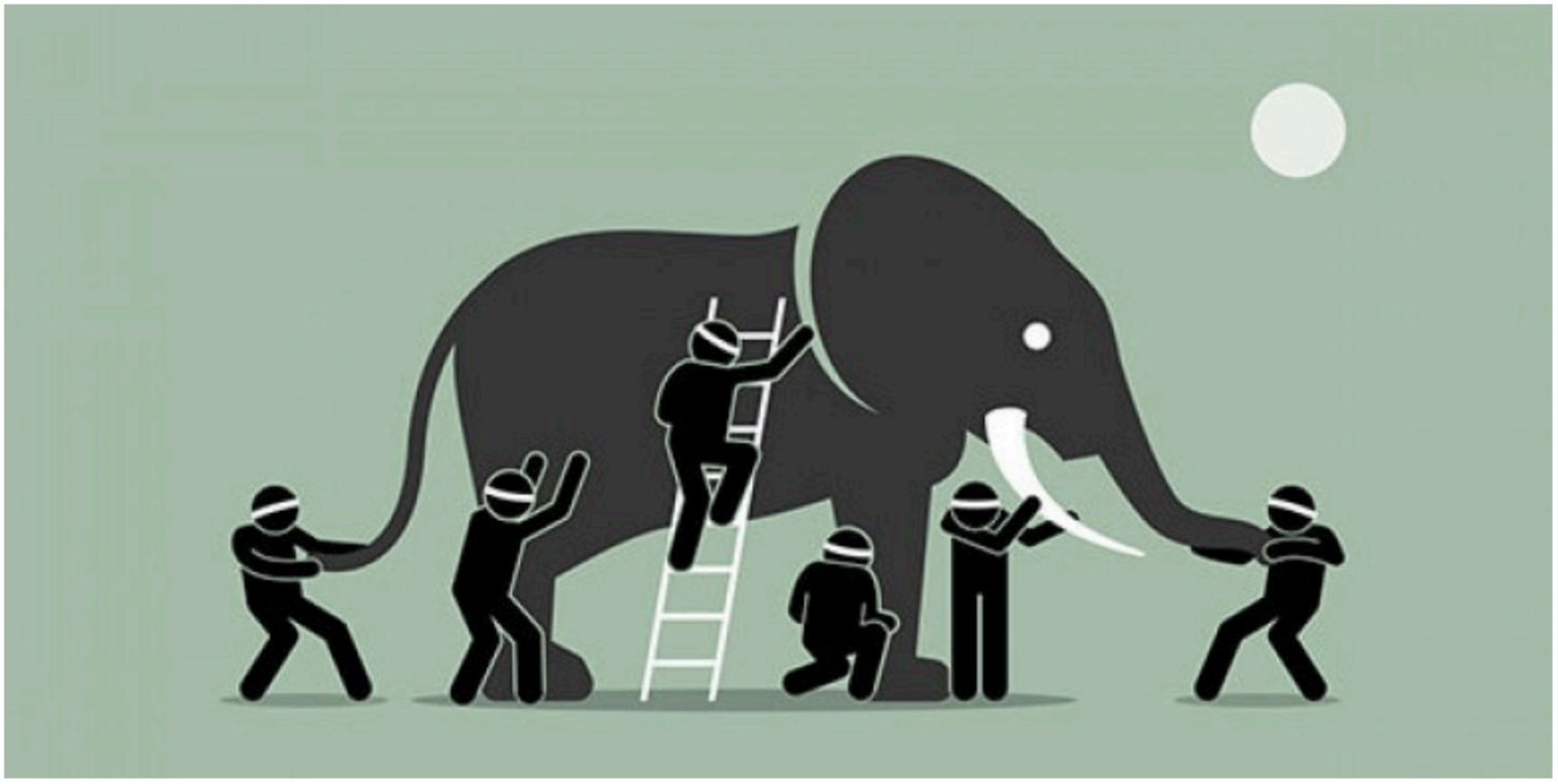
這次分享的內容主要來自 Medium 上的文章，
改用 Wireshark 展示，並補充 Connection Pool 細節

PostgreSQL 使用 Extended Query Protocol 避免頻寬與效能浪費

你的 lib 能啟用 Binary Representation 傳遞參數與結果嗎？

Rui An Huang

Mar 16 · 10 min read



這次比較的 lib

- lib/pq v1.7.0
- Gorm (同 lib/pq)
- go-pg/v10
- pgx/v4

我們希望這些 lib 能幫我們做什麼？怎麼做的？

- 傳送 SQL 讀取結果 -> 用什麼形式？
- 預防 SQL Injection -> 怎麼預防的？

使用範例: lib/pq

```
import (
        "database/sql"

        _ "github.com/lib/pq"
)

func main() {
        connStr := "user=pqgotest dbname=pqgotest sslmode=verify-full"
        db, err := sql.Open("postgres", connStr)
        if err != nil {
                log.Fatal(err)
        }

        age := 21
        rows, err := db.Query("SELECT name FROM users WHERE age = $1", age)
        …
}
```

使用範例: gorm

```
db.Exec("DROP TABLE users;")
db.Exec("UPDATE orders SET shipped_at=? WHERE id IN (?)", time.Now(), []int64{11,22,33})

// Scan
type Result struct {
  Name string
  Age  int
}

var result Result
db.Raw("SELECT name, age FROM users WHERE name = ?", 3).Scan(&result)
```

使用範例: go-pg

```
func Example_placeholders() {
    var num int

    // Simple params.
    _, err := db.Query(pg.Scan(&num), "SELECT ?", 42)
    if err != nil {
        panic(err)
    }
    fmt.Println("simple:", num)

    // Indexed params.
    _, err = db.Query(pg.Scan(&num), "SELECT ?0 + ?0", 1)
    if err != nil {
        panic(err)
    }
    fmt.Println("indexed:", num)
```

使用範例: pgx

```
package main

import (
        "context"
        "fmt"
        "os"

        "github.com/jackc/pgx/v4"
)

func main() {
        conn, err := pgx.Connect(context.Background(), os.Getenv("DATABASE_URL"))
        if err != nil {
                fmt.Fprintf(os.Stderr, "Unable to connect to database: %v\n", err)
                os.Exit(1)
        }
        defer conn.Close(context.Background())

        var name string
        var weight int64
        err = conn.QueryRow(context.Background(), "select name, weight from widgets where id=$1", 42).Scan(&name, &weight)
        if err != nil {
                fmt.Fprintf(os.Stderr, "QueryRow failed: %v\n", err)
                os.Exit(1)
        }

        fmt.Println(name, weight)
}
```

我們希望這些 lib 能幫我們做什麼？怎麼做的？

- 傳送 SQL 讀取結果 -> 用什麼形式？
  - Printable SQL Representation?
- 預防 SQL Injection -> 怎麼預防的？
  - Parameter Placeholder?

Download Wireshark

The current stable release of Wireshark is 3.2.4. It supersedes all previous releases.

Stable Release (3.2.4)

Windows Installer (64-bit)

Windows Installer (32-bit)

Windows PortableApps® (32-bit)

macOS Intel 64-bit .dmg

Source Code

Old Stable Release (3.0.11)

Documentation

範例 Users 表

先插入一個有 200 bytes 的 secret 的 foo user 然後用不同 lib 讀出來看看

```
7   CREATE TABLE users (
8       name   VARCHAR,
9       secret BYTEA
10  );
11
12  INSERT INTO users (name, secret) VALUES ('foo', '\xc9ed8f9e75970bdc8e656cb69
```

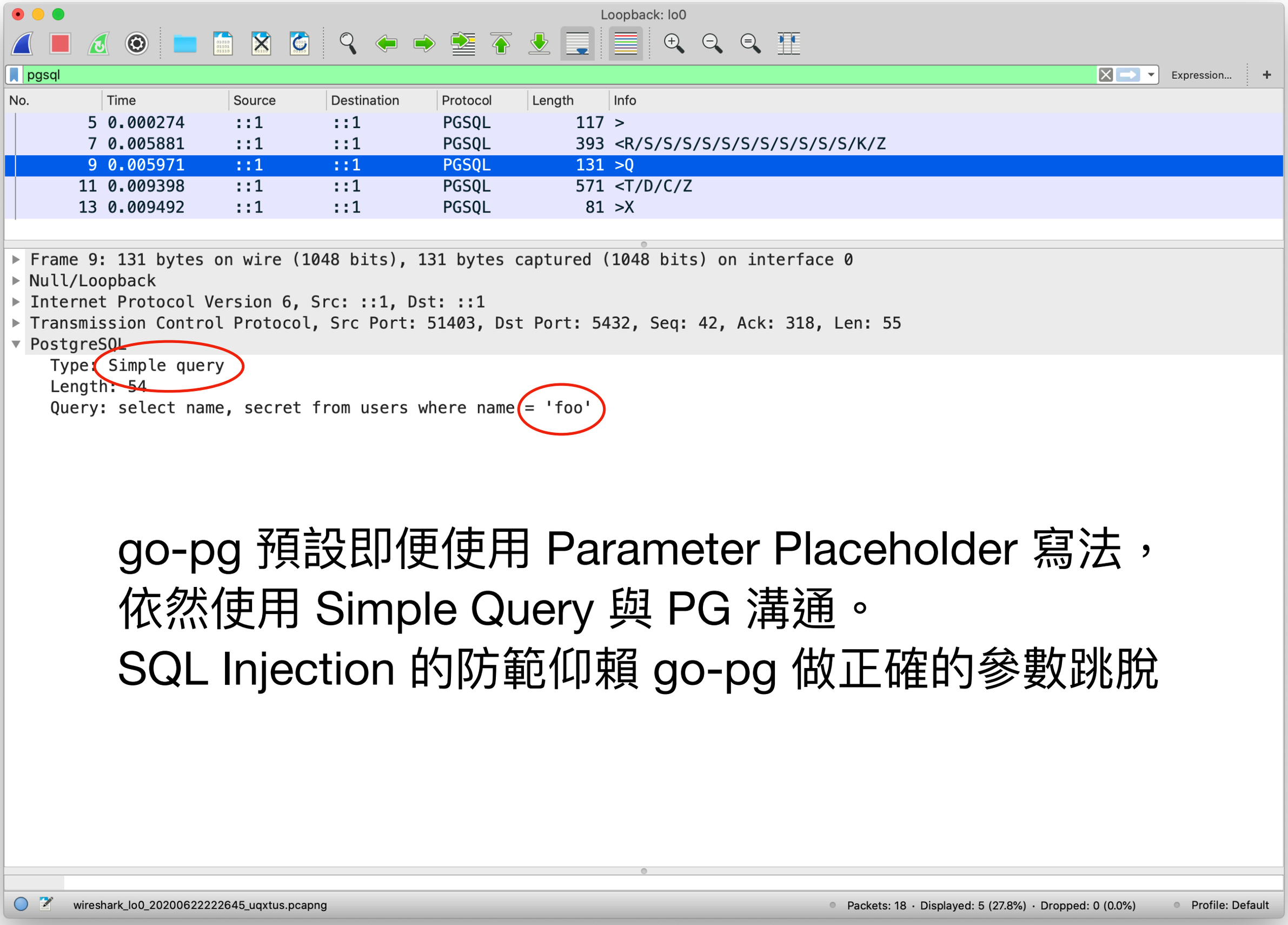
go-pg

```go
package main

import (
    "github.com/go-pg/pg/v10"
)

func main() {
    opt, _ := pg.ParseURL( sURL: "postgres://postgres@localhost:5432/postgres?sslmode=disable")
    db := pg.Connect(opt)
    defer db.Close()

    _, _ = db.Exec( query: `select name, secret from users where name = ?`,  params...: "foo")
}
```



go-pg 預設即便使用 Parameter Placeholder 寫法，
依然使用 Simple Query 與 PG 溝通。
SQL Injection 的防範仰賴 go-pg 做正確的參數跳脫

Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000274	::1	::1	PGSQL	117	>
7	0.005881	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
9	0.005971	::1	::1	PGSQL	131	>Q
11	0.009398	::1	::1	PGSQL	571	<T/D/C/Z
13	0.009492	::1	::1	PGSQL	81	>X

```
▶ Frame 11: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 5432, Dst Port: 51403, Seq: 318, Ack: 97, Len: 495
▼ PostgreSQL
    Type: Row description
    Length: 54
  ▼ Field count: 2
    ▶ Column name: name
    ▶ Column name: secret
▼ PostgreSQL
    Type: Data row
    Length: 419
  ▼ Field count: 2
      Column length: 3
      Data: 666f6f
      Column length: 402
      Data: 5c786339656438663965373539373062646338653635366…
▼ PostgreSQL
    Type: Command completion
    Length: 13
    Tag: SELECT 1
▼ PostgreSQL
    Type: Ready for query
    Length: 5
    Status: Idle (73)
```

並且在 Simple Query 下，回傳 bytea 以 Hex 表示體積變成原本的兩倍

The length of a parameter value, in bytes. -1 means NULL. (pgsql.val.length), 4 bytes

Packets: 18 · Displayed: 5 (27.8%) · Dropped: 0 (0.0%)

Profile: Default

lib/pq

```go
package main

import (
    "database/sql"

    _ "github.com/lib/pq"
)

func main() {
    stdConn, _ := sql.Open(
        driverName: "postgres",
        dataSourceName: `postgres://postgres@localhost:5432/postgres?sslmode=disable`,
    )
    defer stdConn.Close()

    _, _ = stdConn.Query( query: `select name, secret from users where name = $1`,  args...: "foo")
}
```

Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
19	0.609292	::1	::1	PGSQL	178	>
21	0.615767	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
23	0.615918	::1	::1	PGSQL	143	>P/D/S
25	0.617799	::1	::1	PGSQL	153	<1/t/T/Z
27	0.617869	::1	::1	PGSQL	115	>B/E/S
29	0.619514	::1	::1	PGSQL	319	<2/D/C/Z

```
▶ Frame 23: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 50878, Dst Port: 5432, Seq: 103, Ack: 318, Len: 67
▼ PostgreSQL
    Type: Parse
    Length: 54
    Statement:
    Query: select name, secret from users where name = $1
    Parameters: 0
▼ PostgreSQL
    Type: Describe
    Length: 6
    Statement:
▼ PostgreSQL
    Type: Sync
    Length: 4
```

wireshark_lo0_20200622214840_nxtswN.pcapng

Packets: 34 · Displayed: 6 (17.6%) · Dropped: 0 (0.0%)

Profile: Default

Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
19	0.609292	::1	::1	PGSQL	178	>
21	0.615767	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
23	0.615918	::1	::1	PGSQL	143	>P/D/S
25	0.617799	::1	::1	PGSQL	153	<1/t/T/Z
27	0.617869	::1	::1	PGSQL	115	>B/E/S
29	0.619514	::1	::1	PGSQL	319	<2/D/C/Z

```
▶ Frame 25: 153 bytes on wire (1224 bits), 153 bytes captured (1224 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 5432, Dst Port: 50878, Seq: 318, Ack: 170, Len: 77
▼ PostgreSQL
    Type: Parse completion
    Length: 4
▼ PostgreSQL
    Type: Parameter description
    Length: 10
  ▼ Parameters: 1
      Type OID: 25
▼ PostgreSQL
    Type: Row description
    Length: 54
  ▼ Field count: 2
    ▶ Column name: name
    ▼ Column name: secret
        Table OID: 16392
        Column index: 2
        Type OID: 17
        Column length: -1
        Type modifier: -1
        Format: Text (0)
▼ PostgreSQL
    Type: Ready for query
    Length: 5
    Status: Idle (73)
```

type Oid

```
type Oid uint32
```

Oid is a Postgres Object ID.

```
const (
    T_bool              Oid = 16
    T_bytea             Oid = 17
    T_char              Oid = 18
    T_name              Oid = 19
    T_int8              Oid = 20
    T_int2              Oid = 21
    T_int2vector        Oid = 22
```

The object identifier of a type. (pgsql.oid.type), 4 bytes

Packets: 34 · Displayed: 6 (17.6%) · Dropped: 0 (0.0%)

Profile: Default

Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
19	0.609292	::1	::1	PGSQL	178	>
21	0.615767	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
23	0.615918	::1	::1	PGSQL	143	>P/D/S
25	0.617799	::1	::1	PGSQL	153	<1/t/T/Z
27	0.617869	::1	::1	PGSQL	115	>B/E/S
29	0.619514	::1	::1	PGSQL	319	<2/D/C/Z

```
▶ Frame 27: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 50878, Dst Port: 5432, Seq: 170, Ack: 395, Len: 39
▼ PostgreSQL
    Type: Bind
    Length: 23
    Portal:
    Statement:
    Parameter formats: 0
  ▼ Parameter values: 1
      Column length: 3
      Data: 666f6f
  ▼ Result formats: 2
      Format: Text (0)
      Format: Binary (1)
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
▼ PostgreSQL
    Type: Sync
    Length: 4
```

lib/pq 則使用 Extended Query Protocol 避免 SQL Injection，並且指定 bytea 使用 Binary Format 回傳

wireshark_lo0_20200622214840_nxtswN.pcapng

Packets: 34 · Displayed: 6 (17.6%) · Dropped: 0 (0.0%)

Profile: Default

Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
19	0.609292	::1	::1	PGSQL	178	>
21	0.615767	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
23	0.615918	::1	::1	PGSQL	143	>P/D/S
25	0.617799	::1	::1	PGSQL	153	<1/t/T/Z
27	0.617869	::1	::1	PGSQL	115	>B/E/S
29	0.619514	::1	::1	PGSQL	319	<2/D/C/Z

```
▶ Frame 29: 319 bytes on wire (2552 bits), 319 bytes captured (2552 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 5432, Dst Port: 50878, Seq: 395, Ack: 209, Len: 243
▼ PostgreSQL
    Type: Bind completion
    Length: 4
▼ PostgreSQL
    Type: Data row
    Length: 217
  ▼ Field count: 2
      Column length: 3
      Data: 666f6f
      Column length: 200
      Data: c9ed8f9e75970bdc8e656cb6978df8d3c190408effab3a39…
▼ PostgreSQL
    Type: Command completion
    Length: 13
    Tag: SELECT 1
▼ PostgreSQL
    Type: Ready for query
    Length: 5
    Status: Idle (73)
```

bytea 使用 Binary Format 回傳沒有浪費頻寬

wireshark_lo0_20200622214840_nxtswN.pcapng

Packets: 34 · Displayed: 6 (17.6%) · Dropped: 0 (0.0%)

Profile: Default

52.1. Overview 關於 PG 的 Protocol 細節可以於手冊 52 章節找到

The protocol has separate phases for startup and normal operation. In the startup phase, the frontend opens a connection to the server and authenticates itself to the satisfaction of the server. (This might involve a single message, or multiple messages depending on the authentication method being used.) If all goes well, the server then sends status information to the frontend, and finally enters normal operation. Except for the initial startup-request message, this part of the protocol is driven by the server.

During normal operation, the frontend sends queries and other commands to the backend, and the backend sends back query results and other responses. There are a few cases (such as `NOTIFY`) wherein the backend will send unsolicited messages, but for the most part this portion of a session is driven by frontend requests.

Termination of the session is normally by frontend choice, but can be forced by the backend in certain cases. In any case, when the backend closes the connection, it will roll back any open (incomplete) transaction before exiting.

Extended Query Protocol 的好處

- 避免 SQL Injection
- 重複利用 Parse 結果？
- Binary Format 節省頻寬？

PostgreSQL Binary Format vs Text Format

	Binary Format Size	Text Format Size	Text Example
UUID	16 bytes	36 bytes	d4d1d263-4f5c-49bb-ae36-1e26d4adf44a
Timestampz	8 bytes	~ 30 bytes	2020-06-22 17:16:28.87282+00
Bigint	8 bytes	1 ~ 19 bytes	9223372036854775807
Bytea	Variable	Hex 2x size	\x1feb6644e0

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	重複使用 Parse
lib/pq	V			
gorm	V			
go-pg	X			
pgx	V			

lib/pq

```
11  func main() {
12      db, _ := sql.Open(
13          driverName: "postgres",
14          dataSourceName: "postgres://postgres@localhost:5432/postgres?sslmode=disable",
15      )
16      defer db.Close()
17
18      stmt, _ := db.Prepare( query: `SELECT $1::bytea, $2::timestamptz`)
19      defer stmt.Close()
20
21      v := make([]byte, 1000)
22      _, _ = rand.Read(v)
23
24      _, _ = stmt.Exec(v, time.Now())
25  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000160	::1	::1	PGSQL	178	>
7	0.008716	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
9	0.008858	::1	::1	PGSQL	132	>P/D/S
11	0.010356	::1	::1	PGSQL	163	<1/t/T/Z
13	0.010727	::1	::1	PGSQL	3003	>B/E/S
15	0.013346	::1	::1	PGSQL	1145	<2/D/C/Z
17	0.013402	::1	::1	PGSQL	84	>C
18	0.013408	::1	::1	PGSQL	81	>S
21	0.014453	::1	::1	PGSQL	87	<3/Z
23	0.014491	::1	::1	PGSQL	81	>X

```
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 52189, Dst Port: 5432, Seq: 159, Ack: 405, Len: 2927
▼ PostgreSQL
    Type: Bind
    Length: 2911
    Portal:
    Statement: 1
    Parameter formats: 0
  ▼ Parameter values: 2
      Column length: 2854
      Data: 5c303235[illegible]c3333365c5c685c3332355c3234335c3337…
      Column length: 32
      Data: 323032302d30362d32322032333a32383a32302e35353030…
  ▼ Result formats: 2
      Format: Binary (1)
      Format: Text (0)
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
▼ PostgreSQL
    Type: Sync
    Length: 4
```

lib/pq 在使用 Prepare 時，預設使用 Text 傳參數
回傳的 Bytea 使用 Binary Format，
回傳的 Timestampz 使用 Text Format

Loopback: lo0: <live capture in progress>

Packets: 316 · Displayed: 10 (3.2%)

Profile: Default

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000160	::1	::1	PGSQL	178	>
7	0.008716	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
9	0.008858	::1	::1	PGSQL	132	>P/D/S
11	0.010356	::1	::1	PGSQL	163	<1/t/T/Z
13	0.010727	::1	::1	PGSQL	3003	>B/E/S
15	0.013346	::1	::1	PGSQL	1145	<2/D/C/Z
17	0.013402	::1	::1	PGSQL	84	>C
18	0.013408	::1	::1	PGSQL	81	>S
21	0.014453	::1	::1	PGSQL	87	<3/Z
23	0.014491	::1	::1	PGSQL	81	>X

```
▶ Frame 15: 1145 bytes on wire (9160 bits), 1145 bytes captured (9160 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 5432, Dst Port: 52189, Seq: 405, Ack: 3086, Len: 1069
▼ PostgreSQL
    Type: Bind completion
    Length: 4
▼ PostgreSQL
    Type: Data row
    Length: 1043
  ▼ Field count: 2
      Column length: 1000
      Data: 153127de5c68d5a3fa4cae9b4fd896d360ff12cb0f42fa2c…
      Column length: 29
      Data: 323032302d30362d32322031353a32383a32302e35353030…
▼ PostgreSQL
    Type: Command completion
    Length: 13
    Tag: SELECT 1
▼ PostgreSQL
    Type: Ready for query
    Length: 5
    Status: Idle (73)
```

Loopback: lo0: <live capture in progress>

Packets: 340 · Displayed: 10 (2.9%)

Profile: Default

lib/pq + binary_parameters=yes

```
11  func main() {
12      db, _ := sql.Open(
13          driverName: "postgres",
14          dataSourceName: "postgres://postgres@localhost:5432/postgres?sslmode=disable&binary_parameters=yes",
15      )
16      defer db.Close()
17
18      stmt, _ := db.Prepare( query: `SELECT $1::bytea, $2::timestamptz`)
19      defer stmt.Close()
20
21      v := make([]byte, 1000)
22      _, _ = rand.Read(v)
23
24      _, _ = stmt.Exec(v, time.Now())
25  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
25	7.171310	::1	::1	PGSQL	178	>
27	7.175820	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
29	7.175954	::1	::1	PGSQL	132	>P/D/S
31	7.177534	::1	::1	PGSQL	163	<1/t/T/Z
33	7.177842	::1	::1	PGSQL	1152	>B/E/S
35	7.179132	::1	::1	PGSQL	1144	<2/D/C/Z
37	7.179182	::1	::1	PGSQL	84	>C
38	7.179188	::1	::1	PGSQL	81	>S
41	7.180189	::1	::1	PGSQL	87	<3/Z
43	7.180229	::1	::1	PGSQL	81	>X

```
▶ Frame 33: 1152 bytes on wire (9216 bits), 1152 bytes captured (9216 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 52320, Dst Port: 5432, Seq: 159, Ack: 405, Len: 1076
▼ PostgreSQL
    Type: Bind
    Length: 1060
    Portal:
    Statement: 1
  ▼ Parameter formats: 2
      Format: Binary (1)
      Format: Text (0)
  ▼ Parameter values: 2
      Column length: 1000
      Data: 989955045041a723921d782855b1ccb17512e1962b82be18…
      Column length: 31
      Data: 323032302d30362d32322032333a33383a33392e37373131…
  ▼ Result formats: 2
      Format: Binary (1)
      Format: Text (0)
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
```

Prepare 在開啟 binary_parameter 參數後，
Bytea 參數也會使用 Binary Format 傳遞給 PG

Loopback: lo0: <live capture in progress>

Packets: 66 · Displayed: 10 (15.2%)

Profile: Default

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	重複使用 Parse
lib/pq + Parepare	-	X	Bytea Only	
lib/pq + Parepare binary_parameters=yes	-	Bytea Only	Bytea Only	
gorm(lib/pq)	V			
gorm(lib/pq) + binary_parameters=yes	V			
go-pg	X			
pgx	V			

gorm (與 lib/pq 不使用顯性 Prepare 呼叫相同)

```go
func main() {
    db, _ := gorm.Open(
        dialect: "postgres",
        args...: "postgres://postgres@localhost:5432/postgres?sslmode=disable",
    )
    defer db.Close()

    v := make([]byte, 1000)
    _, _ = rand.Read(v)

    db.Exec( sql: `SELECT $1::bytea, $2::timestamptz`, v, time.Now())
}
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	0.007018	::1	::1	PGSQL	83	>Q
14	0.009045	::1	::1	PGSQL	87	<I/Z
19	0.009233	::1	::1	PGSQL	130	>P/D/S
23	0.010620	::1	::1	PGSQL	163	<1/t/T/Z
25	0.010933	::1	::1	PGSQL	3069	>B/E/S
27	0.012383	::1	::1	PGSQL	1145	<2/D/C/Z
29	0.012450	::1	::1	PGSQL	81	>X

```
▶ Frame 25: 3069 bytes on wire (24552 bits), 3069 bytes captured (24552 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 52586, Dst Port: 5432, Seq: 164, Ack: 416, Len: 2993
▼ PostgreSQL
    Type: Bind
    Length: 2977
    Portal:
    Statement:
    Parameter formats: 0
  ▼ Parameter values: 2
      Column length: 2921
      Data: 5c3231332c5c33303351544b5c3231335c3331327e5c3030…
      Column length: 32
      Data: 323032302d30362d32322032333a35363a35342e39323533…
  ▼ Result formats: 2
      Format: Binary (1)
      Format: Text (0)
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
▼ PostgreSQL
    Type: Sync
    Length: 4
```

Loopback: lo0: <live capture in progress>

Packets: 54 · Displayed: 9 (16.7%)

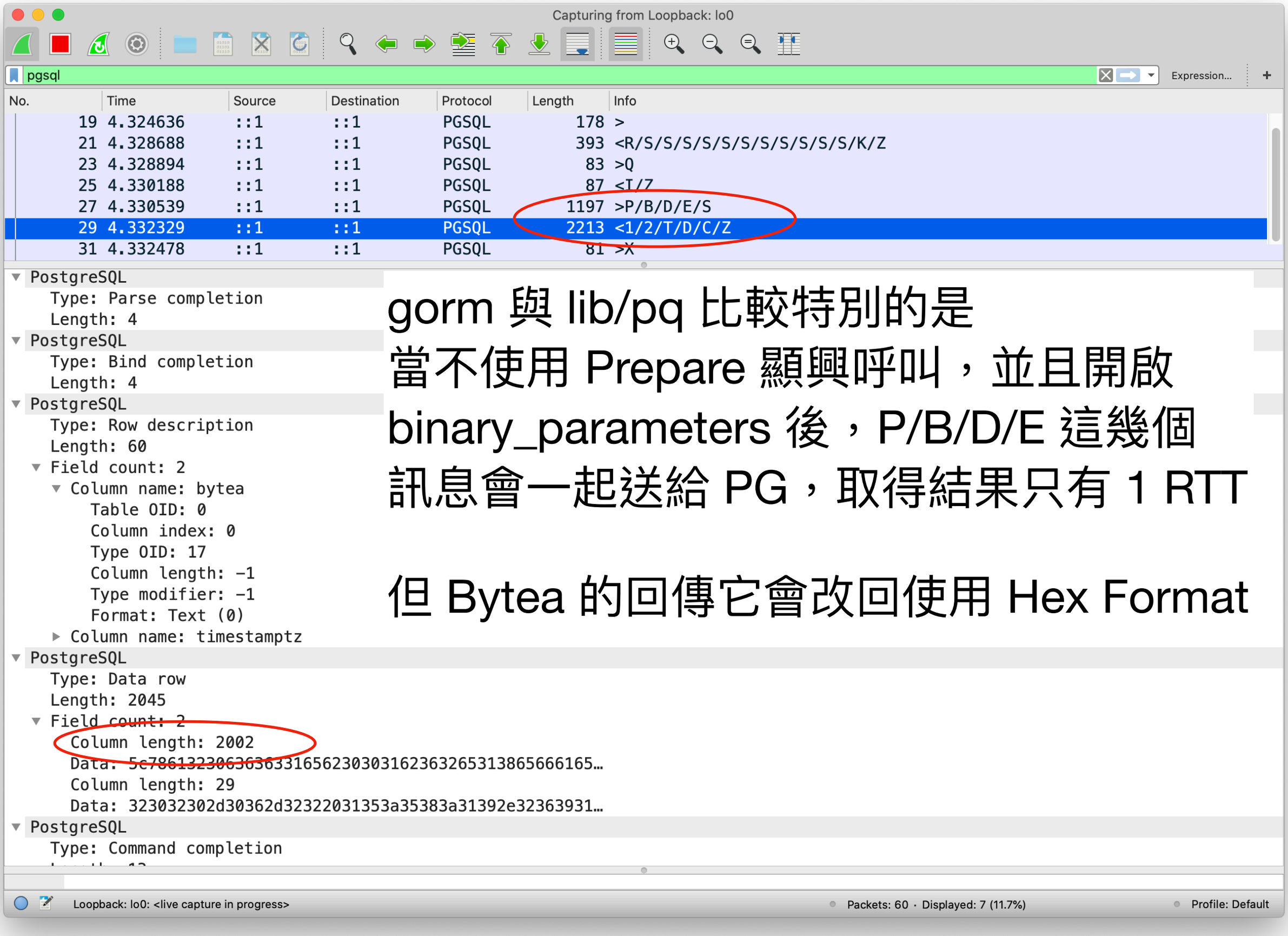
Profile: Default

gorm + binary_parameters=yes

```go
func main() {
    db, _ := gorm.Open(
        dialect: "postgres",
        args...: "postgres://postgres@localhost:5432/postgres?sslmode=disable&binary_parameters=yes",
    )
    defer db.Close()

    v := make([]byte, 1000)
    _, _ = rand.Read(v)

    db.Exec( sql: `SELECT $1::bytea, $2::timestamptz`, v, time.Now())
}
```



gorm 與 lib/pq 比較特別的是
當不使用 Prepare 顯興呼叫，並且開啟 binary_parameters 後，P/B/D/E 這幾個訊息會一起送給 PG，取得結果只有 1 RTT

但 Bytea 的回傳它會改回使用 Hex Format

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT	重複使用 Parse
lib/pq + Parepare	-	X	Bytea Only	2	
lib/pq + Parepare binary_parameters= yes	-	Bytea Only	Bytea Only	2	
gorm(lib/pq)	V	X	Bytea Only	2	
gorm(lib/pq) + binary_parameters= yes	V	Bytea Only	X	1	
go-pg	X				
pgx	V				

Single round-trip parameterized queries without binary_parameters #636

New issue

Closed **cskr** opened this issue on 9 Jul 2017 · 8 comments

cskr commented on 9 Jul 2017 • edited

When `binary_parameters` is not enabled, all parameterized queries take 2 round-trips for execution. Given the PostgreSQL protocol does not require this, single-roundtrip execution must be the default for all unprepared queries, irrespective of whether `binary_parameters` is enabled.

johto commented on 9 Jul 2017 Contributor

> Given the PostgreSQL protocol does not require this, single-roundtrip execution must be the default for all unprepared queries, irrespective of whether binary_parameters is enabled.

It's really not that simple. The root of the problem is that strings and byte slices are guaranteed to be interchangeable. This decision was made a long time ago, and you could argue that it wasn't maybe the right decision, but it's done, and breaking that contract will break a significant number of applications.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked pull requests
Successfully merging a pull request may close this issue.

lib / pq

Watch 155 Star 6k Fork 746

Code Issues 193 Pull requests 76 Actions Projects Wiki Security Insights

binary one way or the other, but not both #632

New issue

Open **rothskeller** opened this issue on 4 Jul 2017 · 0 comments

rothskeller commented on 4 Jul 2017

When `binary_parameters` is disabled, byte arrays are sent to the server in text format (hex encoded), but byte arrays retrieved from the server (via a prepared statement) are retrieved in binary format. When `binary_parameters` is enabled, byte arrays are sent to the server in binary format, but they are retrieved from the server in text format (hex encoded). Is there any way to get byte arrays to be sent in binary format in both directions? If so, how? If not, is it feasible?

👍 1

Assignees
No one assigned

Labels
None yet

Projects
None yet

go-pg

```
10  func main() {
11      opt, _ := pg.ParseURL( sURL: "postgres://postgres@localhost:5432/postgres?sslmode=disable")
12      db := pg.Connect(opt)
13      defer db.Close()
14
15      stmt, _ := db.Prepare( q: `SELECT $1::bytea, $2::timestamptz`)
16
17      v := make([]byte, 1000)
18      _, _ = rand.Read(v)
19
20      _, _ = stmt.Exec(v, time.Now())
21  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
11	0.525226	::1	::1	PGSQL	117	>
13	0.530239	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
15	0.530309	::1	::1	PGSQL	132	>P/D/S
17	0.532015	::1	::1	PGSQL	163	<1/t/T/Z
19	0.532197	::1	::1	PGSQL	2150	>B/E/S
21	0.533870	::1	::1	PGSQL	2147	<2/D/C/Z
23	0.533937	::1	::1	PGSQL	81	>X

```
▶ Frame 19: 2150 bytes on wire (17200 bits), 2150 bytes captured (17200 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 53154, Dst Port: 5432, Seq: 98, Ack: 405, Len: 2074
▼ PostgreSQL
    Type: Bind
    Length: 2058
    Portal:
    Statement: 1
    Parameter formats: 0
  ▼ Parameter values: 2
      Column length: 2002
      Data: 5c786662306364613631653666666439343396464313232…
      Column length: 35
      Data: 323032302d30362d32322031363a33343a30302e30373131…
    Result formats: 0
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
▼ PostgreSQL
    Type: Sync
    Length: 4
```

go-pg 完全不支援 Binary Format

Loopback: lo0: <live capture in progress>

Packets: 112 · Displayed: 7 (6.3%)

Profile: Default

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT	重複使用 Parse
lib/pq + Parepare	-	X	Bytea Only	2	
lib/pq + Parepare binary_parameters= yes	-	Bytea Only	Bytea Only	2	
gorm(lib/pq)	V	X	Bytea Only	2	
gorm(lib/pq) + binary_parameters= yes	V	Bytea Only	X	1	
go-pg + Parepare	-	X	X	2	
pgx	V				

pgx

```
12  func main() {
13      ctx := context.Background()
14      conn, _ := pgxpool.Connect(ctx, connString: "postgres://postgres@localhost:5432/postgres?sslmode=disable")
15      defer conn.Close()
16
17      v := &pgtype.Bytea{Bytes: make([]byte, 1000), Status: pgtype.Present}
18      _, _ = rand.Read(v.Bytes)
19
20      t := &pgtype.Timestamptz{}
21      _ = t.Set(time.Now())
22
23      _, _ = conn.Exec(ctx, sql: `SELECT $1::bytea, $2::timestamptz`, v, t)
24  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	4.636269	::1	::1	PGSQL	117	>
11	4.641779	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
13	4.641957	::1	::1	PGSQL	150	>P/D/S
15	4.643663	::1	::1	PGSQL	163	<1/t/T/Z
17	4.643740	::1	::1	PGSQL	1145	>B/D/E/S
19	4.646416	::1	::1	PGSQL	1185	<2/T/D/C/Z

```
▼ PostgreSQL
    Type: Bind
    Length: 1046
    Portal:
    Statement: lrupsc_1_0
  ▼ Parameter formats: 2
      Format: Binary (1)
      Format: Binary (1)
  ▼ Parameter values: 2
      Column length: 1000
      Data: e665d37d34be70546c8a83ce46a739c3c5cc0918876117a4…
      Column length: 8
      Data: 00024badbebf3d35
  ▼ Result formats: 2
      Format: Binary (1)
      Format: Binary (1)
▼ PostgreSQL
    Type: Describe
    Length: 6
    Portal:
▼ PostgreSQL
    Type: Execute
    Length: 9
    Portal:
    Returns: all rows
▼ PostgreSQL
    Type: Sync
    Length: 4
```

Internet Protocol Version 6 (ipv6), 40 bytes

Packets: 112 · Displayed: 6 (5.4%)

Profile: Default

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	4.636269	::1	::1	PGSQL	117	>
11	4.641779	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
13	4.641957	::1	::1	PGSQL	150	>P/D/S
15	4.643663	::1	::1	PGSQL	163	<1/t/T/Z
17	4.643740	::1	::1	PGSQL	1145	>B/D/E/S
19	4.646416	::1	::1	PGSQL	1185	<2/T/D/C/Z

```
    Column index: 0
    Type OID: 17
    Column length: -1
    Type modifier: -1
    Format: Binary (1)
▼ Column name: timestamptz
    Table OID: 0
    Column index: 0
    Type OID: 1184
    Column length: 8
    Type modifier: -1
    Format: Binary (1)
▼ PostgreSQL
    Type: Data row
    Length: 1022
  ▼ Field count: 2
      Column length: 1000
      Data: c665d37d34be70546c8a83ce46a739c3c5cc0918876117a4…
      Column length: 8
      Data: 00024badbebf3d35
▼ PostgreSQL
    Type: Command completion
    Length: 13
    Tag: SELECT 1
▼ PostgreSQL
    Type: Ready for query
    Length: 5
    Status: Idle (73)
```

The length of a parameter value, in bytes. -1 means NULL. (pgsql.val.length), 4 bytes

Packets: 172 · Displayed: 6 (3.5%)

Profile: Default

README.md

pgtype

pgtype implements Go types for over 70 PostgreSQL types. pgtype is the type system underlying the https://github.com/jackc/pgx PostgreSQL driver. These types support the binary format for enhanced performance with pgx. They also support the database/sql `Scan` and `Value` interfaces and can be used with https://github.com/lib/pq.

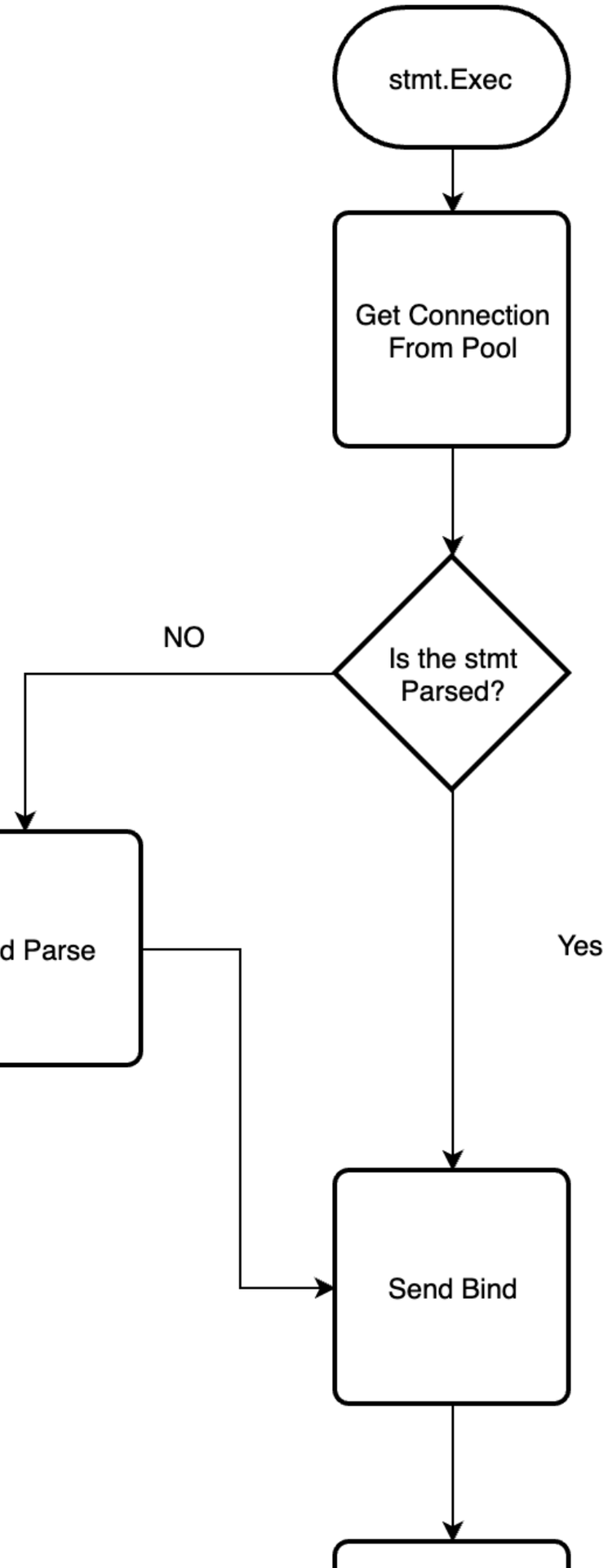
pgx 搭配 pgtype 即可支援多達 70 多種的 Binary Format

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT	重複使用 Parse
lib/pq + Parepare	-	X	Bytea Only	2	
lib/pq + Parepare binary_parameters= yes	-	Bytea Only	Bytea Only	2	
gorm(lib/pq)	V	X	Bytea Only	2	
gorm(lib/pq) + binary_parameters= yes	V	Bytea Only	X	1	
go-pg + Parepare	-	X	X	2	
pgx	V	70 types	70 types	2	

重複使用 Parse
減少 RTT 與解析 SQL 次數?

預期中的實作應該像右邊流程圖
go-pg 目前卻不是這樣



lib/pq

```go
func main() {
    db, _ := sql.Open(
        driverName: "postgres",
        dataSourceName: "postgres://postgres@localhost:5432/postgres?sslmode=disable",
    )
    defer db.Close()
    db.SetMaxOpenConns( n: 2)

    for i := 0; i < 6; i++ {
        go func() {
            db.Exec( query: `SELECT pg_sleep($1)`,  args...: 1)
        }()
    }
    time.Sleep(3 * time.Second)
}
```

將 Connection Pool 大小設定成 2
並送 6 個慢 query 觀察 lib/pq 如何使用 Extended Query Protocol

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	0.000203	::1	::1	PGSQL	178	>
10	0.000209	::1	::1	PGSQL	178	>
13	0.005286	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
15	0.005389	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
17	0.005446	::1	::1	PGSQL	116	>P/D/S
19	0.005470	::1	::1	PGSQL	116	>P/D/S
21	0.007083	::1	::1	PGSQL	132	<1/t/T/Z
23	0.007128	::1	::1	PGSQL	132	<1/t/T/Z
25	0.007149	::1	::1	PGSQL	109	>B/E/S
27	0.007162	::1	::1	PGSQL	109	>B/E/S
29	1.012329	::1	::1	PGSQL	112	<2/D/C/Z
30	1.012361	::1	::1	PGSQL	112	<2/D/C/Z
33	1.012500	::1	::1	PGSQL	116	>P/D/S
35	1.012533	::1	::1	PGSQL	116	>P/D/S
37	1.014807	::1	::1	PGSQL	132	<1/t/T/Z
38	1.014819	::1	::1	PGSQL	132	<1/t/T/Z
41	1.014861	::1	::1	PGSQL	109	>B/E/S
42	1.014875	::1	::1	PGSQL	109	>B/E/S
45	2.018276	::1	::1	PGSQL	112	<2/D/C/Z
47	2.018310	::1	::1	PGSQL	112	<2/D/C/Z
49	2.018398	::1	::1	PGSQL	116	>P/D/S
51	2.018430	::1	::1	PGSQL	116	>P/D/S
53	2.020199	::1	::1	PGSQL	132	<1/t/T/Z
55	2.020217	::1	::1	PGSQL	132	<1/t/T/Z
57	2.020247	::1	::1	PGSQL	109	>B/E/S
58	2.020261	::1	::1	PGSQL	109	>B/E/S
65	3.024882	::1	::1	PGSQL	112	<2/D/C/Z
66	3.024896	::1	::1	PGSQL	112	<2/D/C/Z

不顯性呼叫 Prepare 時
lib/pq 並不會重用 Parse
因此這邊可以看到 6 次 Parse

```
▶ Frame 9: 178 bytes on wire (1424 bits), 178 bytes captured (1424 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 54876, Dst Port: 5432, Seq: 1, Ack: 1, Len: 102
▼ PostgreSQL
    Type: Startup message
```

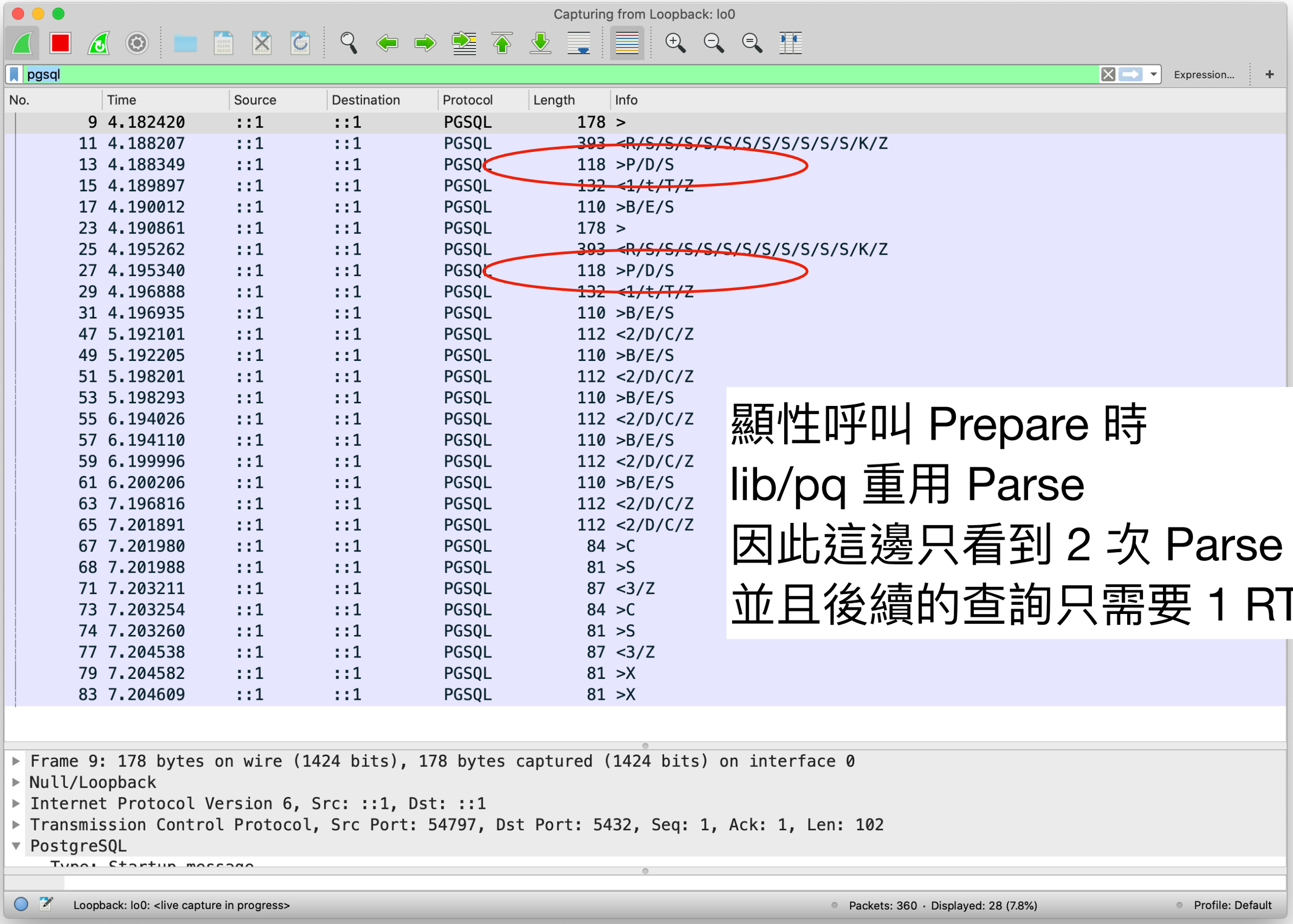
Loopback: lo0: <live capture in progress>

Packets: 162 · Displayed: 28 (17.3%)

Profile: Default

lib/pg + Prepare

```
10  func main() {
11      db, _ := sql.Open(
12          driverName: "postgres",
13          dataSourceName: "postgres://postgres@localhost:5432/postgres?sslmode=disable",
14      )
15      defer db.Close()
16      db.SetMaxOpenConns( n: 2)
17
18      stmt, _ := db.Prepare( query: `SELECT pg_sleep($1)`)
19      defer stmt.Close()
20
21      for i := 0; i < 6; i++ {
22          go func() {
23              stmt.Exec( args...: 1)
24          }()
25      }
26      time.Sleep(3 * time.Second)
27  }
```



顯性呼叫 Prepare 時
lib/pq 重用 Parse
因此這邊只看到 2 次 Parse
並且後續的查詢只需要 1 RTT

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT(重複使用)	重複使用 Parse
lib/pq + Parepare	-	X	Bytea	2(1)	V
lib/pq + Parepare binary_parameters= yes	-	Bytea	Bytea	2(1)	V
gorm(lib/pq)	V	X	Bytea	2	X
gorm(lib/pq) + binary_parameters= yes	V	Bytea	X	1	X
go-pg + Parepare	-	X	X	2	
pgx	V	70 types	70 types	2	

go-pg

```
32  func main() {
33      opt, _ := pg.ParseURL( sURL: "postgres://postgres@localhost:5432/postgres?sslmode=disable")
34      opt.PoolSize = 2
35      db := pg.Connect(opt)
36      defer db.Close()
37
38      stmt, _ := db.Prepare( q: `SELECT pg_sleep($1)`)
39      defer stmt.Close()
40
41      for i := 0; i < 6; i++ {
42          go func() {
43              stmt.Exec( params...: 1)
44          }()
45      }
46      time.Sleep(3 * time.Second)
47  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
11	3.057560	::1	::1	PGSQL	117	>
13	3.063407	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
15	3.063503	::1	::1	PGSQL	118	>P/D/S
17	3.065792	::1	::1	PGSQL	132	<1/t/T/Z
19	3.065906	::1	::1	PGSQL	110	>B/E/S
21	4.067901	::1	::1	PGSQL	112	<2/D/C/Z
23	4.068006	::1	::1	PGSQL	110	>B/E/S
25	5.070674	::1	::1	PGSQL	112	<2/D/C/Z
27	5.070747	::1	::1	PGSQL	110	>B/E/S
29	6.077383	::1	::1	PGSQL	112	<2/D/C/Z
31	6.077513	::1	::1	PGSQL	110	>B/E/S
33	7.079865	::1	::1	PGSQL	112	<2/D/C/Z
35	7.079946	::1	::1	PGSQL	110	>B/E/S
37	8.082907	::1	::1	PGSQL	112	<2/D/C/Z
39	8.082991	::1	::1	PGSQL	110	>B/E/S
49	9.087103	::1	::1	PGSQL	112	<2/D/C/Z
51	9.087247	::1	::1	PGSQL	89	>C/H
53	9.088973	::1	::1	PGSQL	81	<3
55	9.089029	::1	::1	PGSQL	81	>X

go-pg 在相同的測試中只看到 1 次 Parse
與預期中看到 2 次 Parse 不符。

```
▶ Frame 15: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 55001, Dst Port: 5432, Seq: 42, Ack: 318, Len: 42
▼ PostgreSQL
    Type: Parse
```

Loopback: lo0: <live capture in progress>

Packets: 228 · Displayed: 19 (8.3%)

Profile: Default

```
630  // Prepare creates a prepared statement for later queries or
631  // executions. Multiple queries or executions may be run concurrently
632  // from the returned statement.
633  func (db *baseDB) Prepare(q string) (*Stmt, error) {
634      return prepareStmt(db.withPool(pool.NewSingleConnPool(db.pool)), q)
635  }
```

這是因為 go-pg 在 prepare stmt 創建的的當下會立刻綁定一個 connection 給它。

這個 stmt 雖然可以 concurrently 呼叫，但它的 concurrency 被限制在只有 1，容易有效能問題。

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT(重複使用)	重複使用 Parse
lib/pq + Parepare	-	X	Bytea	2(1)	V
lib/pq + Parepare binary_parameters= yes	-	Bytea	Bytea	2(1)	V
gorm(lib/pq)	V	X	Bytea	2	X
gorm(lib/pq) + binary_parameters= yes	V	Bytea	X	1	X
go-pg + Parepare	-	X	X	2(1)	concurrency=1
pgx	V	70 types	70 types	2	

pgx

```
10  func main() {
11      ctx := context.Background()
12      conn, _ := pgxpool.Connect(
13          ctx,
            connString: "postgres://postgres@localhost:5432/postgres?sslmode=disable&pool_max_conns=2",
15      )
16      defer conn.Close()
17
18      for i := 0; i < 6; i++ {
19          go func() {
20              conn.Exec(ctx, sql: `SELECT pg_sleep($1)`, arguments...: 1)
21          }()
22      }
23      time.Sleep(3 * time.Second)
24  }
```

Capturing from Loopback: lo0

pgsql

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	1.784176	::1	::1	PGSQL	117	>
11	1.788804	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
13	1.788961	::1	::1	PGSQL	136	>P/D/S
19	1.790010	::1	::1	PGSQL	117	>
21	1.790645	::1	::1	PGSQL	132	<1/t/T/Z
23	1.790774	::1	::1	PGSQL	137	>B/D/E/S
25	1.794717	::1	::1	PGSQL	393	<R/S/S/S/S/S/S/S/S/S/S/S/K/Z
27	1.794819	::1	::1	PGSQL	136	>P/D/S
29	1.796316	::1	::1	PGSQL	132	<1/t/T/Z
31	1.796367	::1	::1	PGSQL	137	>B/D/E/S
33	2.794924	::1	::1	PGSQL	146	<2/T/D/C/Z
35	2.795059	::1	::1	PGSQL	137	>B/D/E/S
37	2.797301	::1	::1	PGSQL	146	<2/T/D/C/Z
39	2.797373	::1	::1	PGSQL	137	>B/D/E/S
41	3.798708	::1	::1	PGSQL	146	<2/T/D/C/Z
43	3.798813	::1	::1	PGSQL	137	>B/D/E/S
45	3.799362	::1	::1	PGSQL	146	<2/T/D/C/Z
47	3.799444	::1	::1	PGSQL	137	>B/D/E/S
49	4.801237	::1	::1	PGSQL	146	<2/T/D/C/Z
51	4.801334	::1	::1	PGSQL	81	>X
53	4.801857	::1	::1	PGSQL	146	<2/T/D/C/Z
55	4.801960	::1	::1	PGSQL	81	>X

```
▶ Frame 27: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 55101, Dst Port: 5432, Seq: 42, Ack: 318, Len: 60
▼ PostgreSQL
    Type: Parse
```

Loopback: lo0: <live capture in progress>

Packets: 180 · Displayed: 22 (12.2%)

Profile: Default

Extended Query Protocol 的支援程度

	自動使用 EQP	Binary Parameter	Binary Data Result	RTT(重複使用)	重複使用 Parse
lib/pq + Parepare	-	X	Bytea	2(1)	V
lib/pq + Parepare binary_parameters= yes	-	Bytea	Bytea	2(1)	V
gorm(lib/pq)	V	X	Bytea	2	X
gorm(lib/pq) + binary_parameters= yes	V	Bytea	X	1	X
go-pg + Parepare	-	X	X	2(1)	concurrency=1
pgx	V	70 types	70 types	2(1)	V

Q&A

https://join.dcard.today/

夥伴招募中，歡迎投履歷～