



What's new in PostgreSQL 16

Speaker: 林宗禧 @ COSCUP 2023

Taiwan PostgreSQL User Group

2023.7.30

林宗禧 Tsung-Hsi Lin

- PostgreSQL愛好者(2012-)
- PostgreSQL推廣者(2017-)

How to use ?

- 以前: 開發FDW套件 (C, Python都有)
- 後來: 到處整合PG的應用
 - 推 Industry 4.0，讓業主不經意的導入 PG
 - 推 Smart City Solutions，拿PG做基礎
 - 前陣子研究碳排，最近在做運動數據





為何討論這個Topic?

- 今年度6月發布了PG 16 Beta2 (29 Jun.)
- PostgreSQL16 增加200多個新功能

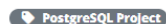
27th July 2023: [Setting the record straight: More updates on a trademark dispute](#)

Quick Links

- About
- Policies
- Feature Matrix
- Donate
- History
- Sponsors
 - Servers
- Latest News
- Upcoming Events
 - Past events
- Press
- Licence

PostgreSQL 16 Beta 2 Released!

Posted on **2023-06-29** by PostgreSQL Global Development Group



The PostgreSQL Global Development Group announces that the second beta release of PostgreSQL 16 is now **available for download**. This release contains previews of all features that will be available when PostgreSQL 16 is made generally available, though some details of the release can change during the beta period.

You can find information about all of the features and changes found in PostgreSQL 16 in the **release notes**:

<https://www.postgresql.org/docs/16/release-16.html>

In the spirit of the open source PostgreSQL community, we strongly encourage you to test the new features of PostgreSQL 16 on your systems to help us eliminate bugs or other issues that may exist. While we do not advise you to run PostgreSQL 16 Beta 2 in production environments, we encourage you to find ways to run your typical application workloads against this beta release.

Your testing and feedback will help the community ensure that the PostgreSQL 16 release upholds our standards of delivering a stable, reliable release of the world's most advanced open source relational database. Please read more about our **beta testing process** and how you can contribute:

<https://www.postgresql.org/developer/beta/>

Upgrading to PostgreSQL 16 Beta 2

To upgrade to PostgreSQL 16 Beta 2 from an earlier version of PostgreSQL, you will need to use a strategy similar to upgrading between major versions of PostgreSQL (e.g. `pg_upgrade` or `pg_dump` / `pg_restore`). For more information, please visit the documentation section on **upgrading**.

Changes Since Beta 1

Fixes and changes in PostgreSQL 16 Beta 2 include:

- The default collation provider selected by `initdb` is changed back to `libc`.
- The behavior of selecting the `c` locale with `libc` is deferred back to `libc`. On ICU 64 and higher, the `c` locale is obsolete, and ICU provides its own mechanism for selecting a locale or throwing an error.
- Several fixes related to join optimizations.
- Fix for B-tree code related to the changes introduced by logical decoding from standbys.
- Fix cache lookup hazards when looking up `MAINTAIN` privileges on partition ancestors.

Please see the **release notes** for a complete list of new and changed features:

<https://www.postgresql.org/docs/16/release-16.html>

Testing for Bugs & Compatibility



Agenda

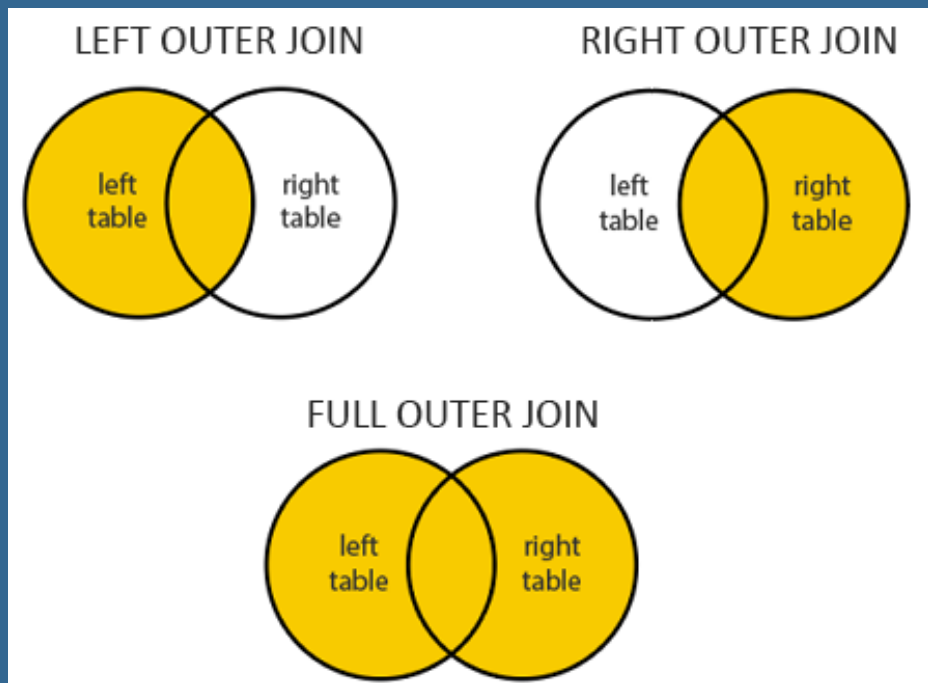
01. 適應大規模環境的精進
02. Monitoring
03. Client applications
04. Server 端管理與維運
05. Localization
06. 安全性
07. SQL functions and commands
08. Performance
09. Logical replication
10. 小結



01. 適應大規模環境的精進 (1/3)

■ 平行查詢改進 (Parallel Query Improvements)

平行查詢可以使用在 FULL OUTER JOIN 和 RIGHT OUTER JOIN 語法，string_agg 和 array_agg 函數



```
16=# EXPLAIN (costs off)
SELECT fare_conditions, array_agg(flight_id), string_agg(ticket_no, ',')
FROM ticket_flights
GROUP BY fare_conditions;
```

QUERY PLAN

Finalize GroupAggregate

Group Key: fare_conditions

-> Gather Merge

Workers Planned: 2

-> Sort

Sort Key: fare_conditions

-> Partial HashAggregate

Group Key: fare_conditions

-> Parallel Seq Scan on ticket_flights

```
15=# EXPLAIN (costs off)
SELECT fare_conditions, array_agg(fli
FROM ticket_flights
GROUP BY fare_conditions;
```

QUERY PLAN

HashAggregate

Group Key: fare_conditions

-> Seq Scan on ticket_flights

<https://www.dofactory.com/sql/outer-join>

https://postgrespro.com/blog/pgsql/5969981#commit_16fd03e9



01. 適應大規模環境的精進 (2/3)

■ 擴展統計 (Extended Statistics)

增加 pg_stat_io 的view，以獲取I/O統計資訊。

(pg_stat_all_tables 和 pg_stat_all_indexes 的 view添加了一列，顯示上次訪問時間)

```
-[ RECORD 1 ]+-----  
backend_type | client backend  
object       | relation  
context      | bulkread  
reads        | 427497  
read_time    | 752.489  
writes       | 77220  
write_time   | 215.93  
extends      |  
extend_time  |  
op_bytes     | 8192  
hits         | 38683  
evictions    | 94210  
reuses       | 330437  
fsyncs       |  
fsync_time   |  
stats_reset  | 2023-04-29 09:13:58.798952+03
```

```
SELECT backend_type, SUM(writes) blocks,  
       pg_size_pretty(SUM(writes*op_bytes)) size,  
       round(SUM(write_time)) "time, ms"  
FROM pg_stat_io  
WHERE writes > 0  
GROUP BY ROLLUP (backend_type)  
ORDER BY blocks;
```

backend_type	blocks	size	time, ms
background writer	17198	134 MB	187
checkpointer	30436	238 MB	139
background worker	76929	601 MB	213
autovacuum worker	88870	694 MB	528
client backend	369031	2883 MB	1055
	582464	4551 MB	2122

(6 rows)

1)跟踪系統範圍的緩衝區緩存命中（以允許計算準確的緩衝區緩存命中率）

2)累積的系統範圍 I/O 時間（不僅僅是當前存在的 I/O 計數pg_stat_io）

3)更好的累積 WAL 統計數據（即超越 pg_stat_wal 提供的）

4)表和索引的附加 I/O 跟踪



01. 適應大規模環境的精進 (3/3)

■ 精進批次處理效能 (Improved batch processing performance)

透過 postgres_fdw 可以使用 COPY 語法對 foreign table 批次插入多筆資料。

→ INSERT , COPY 支援 FDW 中的批次處理



02. Monitoring

- pg_buffercache：新的
pg_buffercache_usage_counts
函數
- EXPLAIN (generic_plan)：參數
化查詢的通用計劃

```
EXPLAIN SELECT count(*) FROM tickets WHERE book_ref = $1;
```

```
ERROR:  there is no parameter $1
```

```
LINE 1: EXPLAIN SELECT count(*) FROM tickets WHERE book_ref = $1;
```

```
^
```

```
SELECT * FROM pg_buffercache_usage_counts();
```

usage_count	buffers	dirty	pinned
0	15791	0	0
1	105	1	0
2	89	4	0
3	22	1	0
4	32	3	0
5	345	27	0

```
(6 rows)
```

```
EXPLAIN (generic_plan)
```

```
SELECT count(*) FROM tickets WHERE book_ref = $1;
```

```
QUERY PLAN
```

```
Aggregate (cost=65779.07..65779.08 rows=1 width=8)
```

```
-> Gather (cost=1000.00..65779.07 rows=2 width=0)
```

```
Workers Planned: 2
```

```
-> Parallel Seq Scan on tickets (cost=0.00..64778.87 rows=1 width=0)
```

```
Filter: (book_ref = $1)
```



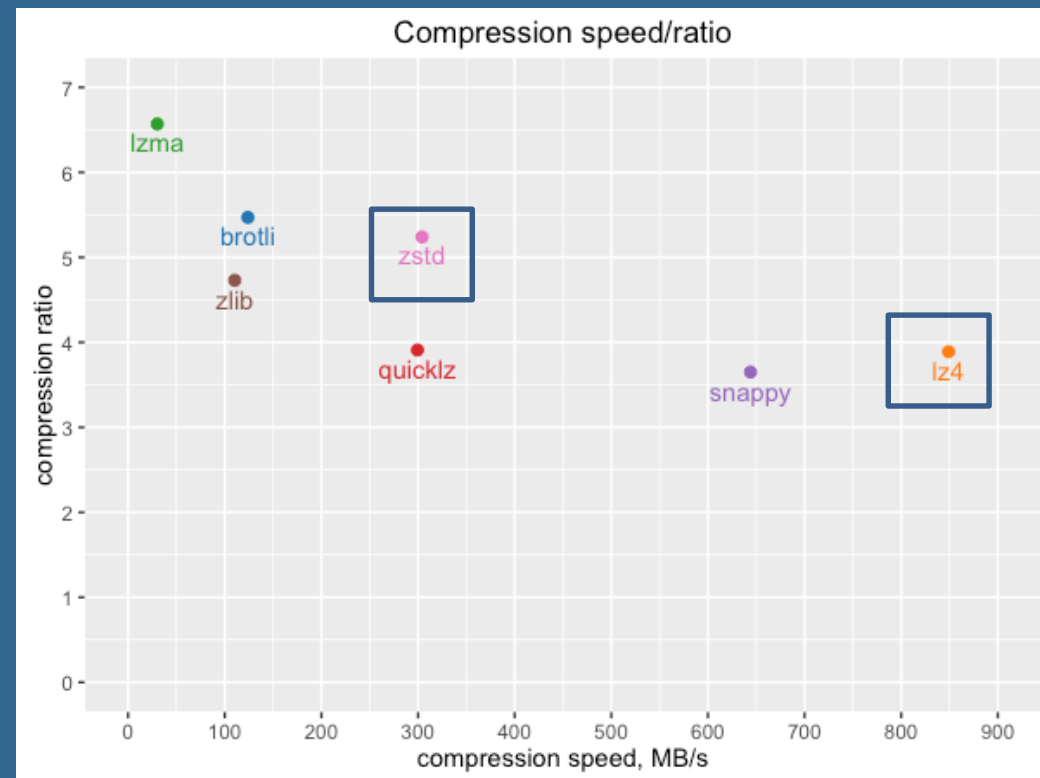

03. Client applications

■ pg_dump：支援 LZ4 和 zstd 壓縮方法

→ 在 PostgreSQL 16 之前，pg_dump 支援的唯一壓縮方法是 gzip。

→ 這次commit分別增加了 LZ4 和 zstd。

- LZ4、zstd皆是無失真資料壓縮演算法，著重於壓縮和解壓縮速度。
- Lz4 相較預設的pglz 壓縮結果相近，但 CPU 成本較低。
- Zstd 可以提供最高的壓縮率（比 lz4 高出 30%）。



<https://blog.gslin.org/archives/2021/11/10/10411/postgresql-14-%E6%94%AF%E6%8F%B4%E7%9A%84-lz4-%E5%A3%93%E7%B8%AE/>

■ pg_dump 與 partitioned tables

→ 增加新參數：--table-and-children、
--exclude-table-and-children和--exclude-table-data-
and-children



03. Client applications

■ libpq：平衡連接

為了均勻分佈副本之間的連接，可以在形成連接字符串時在應用端打亂副本列表。
或可以使用新的連接參數 `load_balance_hosts`：

```
psql "host=replica1,replica2,replica3"
```

面對大量連線，為了均勻分佈副本之間的連接，可以在連接字串的Client隨意書潤副本列表。
或可以使用新的連接參數 `load_balance_hosts`：

```
psql "host=replica1,replica2,replica3 load_balance_hosts=random"
```

`load_balance_hosts=random` 表示在嘗試連接之前將節點隨機分派。



04. Server 端管理與維運

■ initdb：設定初始化時的配置參數

16 的 initdb 輸入 -c（或--set）將覆蓋任何配置參數的值。

```
$ initdb -k -U postgres "-c wal_level=logical" "-c port=5402" -D ~/pg16/data2
$ pg_ctl start -D ~/pg16/data2 -l logfile2
$ psql -p 5402 -c "SHOW wal_level"

wal_level
-----
logical
```

初始化期間指定的參數存儲在postgresql.conf文件的末尾：

```
$ tail -2 ~/pg16/data2/postgresql.conf

wal_level = logical
port = 5402
```



04. Server 端管理與維運

■ Autovacuum：動態配置平衡 I/O

→ 當Autovacuum需要太長時間來vacuum大型Table時，
可以即時修改參數`autovacuum_vacuum_cost_limit`和/或
`autovacuum_vacuum_cost_delay`更新配置。

Ps. 在以前的 PostgreSQL 版本中，autovacuum 進程只有在處理完當前表後才能讀取配置文件。

■ 管理用於 VACUUM 和 ANALYZE 的共享記憶體大小

→ 新參數`Vacuum_buffer_usage_limit`，可用於控制環形緩衝區的大小。
→ 預設值維持 256 kB，但參數範圍可接受 12 kB 到 16 GB 的值（但不能使用超過 1/8 的緩衝區高速緩存）。



05. Localization

Internationalization and Unicode Conference (IUC)

■ ICU：UNICODE 排序規則

根據 SQL 標準，ICU 提供的
UNICODE 排序規則已在
PostgreSQL 16 中實作。

```
\d0 unicode
```

List of collations							
Schema	Name	Provider	Collate	Ctype	ICU Locale	ICU Rules	Deterministic?
pg_catalog	unicode	icu			und		yes

■ ICU：語言環境的規範化

ICU 提供商的區域設置名稱
將根據 BCP 47 規則轉換為
規範形式。並避免創建無效的排序規則。

```
CREATE COLLATION test_icu(provider=icu, locale='invalid_locale');
```

```
NOTICE: using standard form "invalid-locale" for locale "invalid_locale"
```

```
ERROR: ICU locale "invalid-locale" has unknown language "invalid"
```

```
HINT: To disable ICU locale validation, set parameter icu_validation_level to DISABLED.
```

■ ICU：自定義排序算法的自定義規則

舉例：排序以冬天為第一個季節進行

```
CREATE COLLATION seasons (  
    provider = icu, locale = 'en', rules = '& W < Sp < Su < A'  
);
```

```
WITH seasons(name) AS (  
    VALUES ('Summer'), ('Winter'), ('Spring'), ('Autumn')  
)
```

```
SELECT * FROM seasons ORDER BY name COLLATE seasons;
```



06. 安全性

■ libpq：新參數 require_auth

新的 libpq 參數 require_auth 允許指定 Service 應該使用的身份驗證方法。
可以列出多個方法允許服務器選擇使用。

```
SELECT rule_number rule, type, database, user_name, auth_method
FROM pg_hba_file_rules
WHERE database != '{replication}';
```

rule	type	database	user_name	auth_method
1	local	{all}	{all}	trust
2	host	{all}	{all}	scram-sha-256

```
$ psql 'host=localhost require_auth=md5,scram-sha-256'
-c 'SELECT system_user'
```

Password for user postgres:

system_user

scram-sha-256:postgres

(1 row)

- scram_iterations：使用 SCRAM-SHA-256 進行密碼加密的迭代計數器
- 新增 GSSAPI/Kerberos authentication



07. SQL functions and commands

- SQL/JSON 標準支援

- PostgreSQL 16 的增加對結構函數 (JSON_ARRAY, JSON_ARRAYAGG, JSON_OBJECT, JSON_OBJECTAGG) 和謂詞 (IS JSON [VALUE], IS JSON ARRAY, IS JSON OBJECT, IS JSON SCALAR) 的支援。

- 新函數 array_shuffle 和 array_sample

- array_shuffle : 對值進行混洗

- array_sample : 回傳指定長度的 Array

```
WITH a(digits) AS (  
    SELECT '{1,2,3,4,5,6,7,8,9,0}'::int[]  
)  
SELECT array_shuffle(a.digits),  
       array_sample(a.digits, 3)  
FROM a;
```

array_shuffle		array_sample
---------------	--	--------------

{1,0,9,8,2,3,5,7,4,6}		{9,1,4}
-----------------------	--	---------



07. SQL functions and commands

■ 新聚合函數any_value

SQL 標準聚合函數any_value為一組行返回任意非空值。
考慮一個沒有唯一鍵的表，其中一些行是重複的。

```
CREATE TABLE t (id int);  
INSERT INTO t  
VALUES(1),(1),(2),(3),(3) RETURNING *;
```

id
1
1
2
3
3

(5 rows)



```
DELETE FROM t  
WHERE ctid IN (SELECT any_value(ctid)  
               FROM t  
               GROUP BY id HAVING count(*) = 2  
);  
  
DELETE 2
```

```
SELECT * FROM t;
```

id
1
2
3

(3 rows)



07. SQL functions and commands

■ timestampz：加減時間間隔

```
SET timezone = 'UTC';

SELECT '2021-10-31 00:00:00+02'::timestampz + '1 day'::interval,
       date_add('2021-10-31 00:00:00+02'::timestampz, '1 day'::interval, 'Europe/Warsaw');
```

?column?		date_add
-----+-----		
2021-10-31 22:00:00+00		2021-10-31 23:00:00+00

■ XML：格式化值 XMLSERIALIZE函數提供 用於格式化輸出的參數 INDENT。

```
SELECT XMLSERIALIZE(
  DOCUMENT '<feature><name>Support [NO] INDENT option in XMLSERIALIZE()</name><author>Jim Jones</author><reviewers>Peter Smith and Tom Lane</reviewers><committer>Tom Lane</committer></feature>'
  INDENT);
```

xmlserialize
-----+
<feature>+
<name>Support [NO] INDENT option in XMLSERIALIZE()</name>+
<author>Jim Jones</author>+
<reviewers>Peter Smith and Tom Lane</reviewers>+
<committer>Tom Lane</committer>+
</feature>+



08. Performance

- ✓ 平行查詢改進 (Parallel Query Improvements)
- ✓ 精進批次處理效能 (Improved batch processing performance)
- **BRIN 索引不阻止熱更新**

There should be no (or minimal) UPDATE/DELETE on the table. UPDATE/DELETE cause dead tuples in the blocks and those tuples are released later with vacuuming process. Once vacuuming has happened, those blocks are part of the free list (In Postgres, there is no PCTUSED parameter like Oracle) and new rows will come into those block holes and hence changing min/max range for those blocks to fall in bigger range (assuming data is coming in sorted fashion as per mentioned in the 1st point).

```
CREATE INDEX flights_bi ON flights USING brin(actual_departure);
SELECT pg_stat_reset_single_table_counters('flights'::regclass);

UPDATE flights SET actual_departure = actual_departure + '5 min'::interval
WHERE flight_id = 1;

SELECT n_tup_hot_upd, n_tup_newpage_upd
FROM pg_stat_all_tables
WHERE relname = 'flights';
```

n_tup_hot_upd	n_tup_newpage_upd
1	0



09. Logical replication

■ 從物理副本進行邏輯複製

須將主節點和物理副本節點的wal_level參數設置為logical才能正常執行

```
replica=# SELECT pg_is_in_recovery(), current_setting('wal_level');
```

pg_is_in_recovery	current_setting
-----	-----
t	logical

```
primary=# CREATE TABLE t (id int primary key);
primary=# INSERT INTO t VALUES (42);
```

```
primary=# CREATE PUBLICATION pub FOR TABLE t;
```

在主服務器上創建發布：

```
primary=# CREATE TABLE t (id int primary key);
primary=# INSERT INTO t VALUES (42);
```

```
primary=# CREATE PUBLICATION pub FOR TABLE t;
```

```
subscriber=# CREATE TABLE t (id int primary key);
```

```
subscriber=# CREATE SUBSCRIPTION sub CONNECTION 'dbname=postgres port=5401' PUBLICATION pub;
```

```
primary=# SELECT pg_log_standby_snapshot();
```

```
replica=# SELECT slot_name, active FROM pg_replication_slots WHERE slot_name = 'sub';
```

slot_name	active
-----	-----
sub	t

(1 row)



10. 小結

■ 還有很多項目可以仔細研讀

- New function: random_normal
- Input formats for integer literals
- Goodbye, postmaster
- Parallel execution for string_agg and array_agg
- New parameter: enable_presorted_aggregate
- Planner support function for working with window functions
- Optimized grouping of repeating columns in GROUP BY and DISTINCT
- VACUUM parameters: SKIP_DATABASE_STATS and ONLY_DATABASE_STATS
- pg_dump: lock tables in batches
- PL/pgSQL: cursor variable initialization
- Roles with the CREATEROLE attribute
- Setting parameter values at the database and user level
- New parameter: reserved_connections
- postgres_fdw: analyzing foreign tables with TABLESAMPLE
- postgres_fdw: batch insert records during partition key updates
- pg_ident.conf: new ways to identify users in PostgreSQL
- Query jumbling for DDL and utility statements
- New function: bt_multi_page_stats
- New function: pg_split_walfile_name
- pg_walinspect, pg_waldump: collecting page images from WAL


- psql: \pset xheader_width
- vacuumdb --schema and --exclude-schema options
- New createuser features added
- Checkpoint and redo LSN added to LogCheckpointEnd log message
- pg_prepared_statements.result_types
- New parameter log_parameter_max_length for auto_explain
- Subquery alias in FROM clause became optional
- REINDEX: syntax improvements and more
- CREATE STATISTICS: statistic name is now optional
- CREATE TABLE: the STORAGE attribute
- The user created during cluster initialization now can't be stripped of superuser privileges
- TRUNCATE triggers on foreign tables are now supported
- New argument variation for pg_read_file/pg_read_binary_file
- Extensible WAL resource managers
- SYSTEM_USER function
- Frozen pages/tuples information in autovacuum's server log
- pg_stat_get_backend_idset returns the actual backend ID
- Improved performance of ORDER BY / DISTINCT aggregates
- Faster bulk-loading into partitioned tables
- Optimized lookups in snapshots
- Bidirectional logical replication
- pg_auth_members: role membership granting management
- pg_auth_members: role membership and privilege inheritance
- pg_receive_wal and pg_recvlogical can now handle SIGTERM



參考資料

- PostgresPro - Pavel Luzanov <https://postgrespro.com/blog/pgsql/5969929>
- HP Enterprise - Noriyoshi Shinoda



**Hewlett Packard
Enterprise**

May 28, 2023

PostgreSQL 16 New Features
With Examples (Beta 1)

Hewlett Packard Enterprise Japan G.K.
Noriyoshi Shinoda

© 2022-2023 Hewlett Packard Enterprise Japan G.K.

1

Index	
Index.....	2
1. About This Document.....	5
1.1. Purpose.....	5
1.2. Audience.....	5
1.3. Scope.....	5
1.4. Software Version.....	5
1.5. The Question, comment, and Responsibility.....	6
1.6. Notation.....	6
2. New Features: Summary.....	7
2.1. Improvements to adapt to large scale environments.....	7
2.2. Improved reliability.....	7
2.3. Improved maintainability.....	8
2.4. Improvements in Programming.....	8
2.5. Preparing for future new features.....	8
2.6. Incompatibility.....	9
2.6.1. End of support.....	9
2.6.2. Promotion.....	9
2.6.3. Bootstrap user attribute.....	9
2.6.4. WAL Archive.....	10
2.6.5. Extension.....	11
2.6.6. Datetime string format.....	11
2.6.7. ASCII string conversion.....	12
2.6.8. ALTER DEFAULT PRIVILEGES statement.....	12
2.6.9. ALTER TABLE statement.....	12
2.6.10. CREATE RULE statement.....	13
2.6.11. CREATE TABLE statement.....	13
2.6.12. CREATEROLE attribute.....	13
2.6.13. POWER function.....	14
2.6.14. RESET statement.....	15
2.6.15. Pg_stat_get_backend_idset function.....	15
2.6.16. Pg_walinspect extension.....	16
2.6.17. Postmaster.....	17
2.6.18. PsqI.....	17
2.6.19. Libpq.....	18
© 2022-2023 Hewlett Packard Enterprise Japan G.K.	
2	

Hewlett Packard Enterprise	
2.6.20. PL/Python.....	18
2.6.21. ECPG.....	18
3. New Feature Detail.....	19
3.1. Architecture.....	19
3.1.1. Modified System catalogs.....	19
3.1.2. Logical Replication Enhancements.....	23
3.1.3. Parallel Query Enhancements.....	27
3.1.4. Configuration Files.....	28
3.1.5. ICU Locale.....	29
3.1.6. Predefined Roles.....	33
3.1.7. Wait Events.....	33
3.1.8. Triggers.....	34
3.1.9. Log files.....	34
3.1.10. Kerberos Credential Delegation.....	35
3.1.11. Libpq.....	35
3.1.12. PL/pgSQL.....	38
3.1.13. MATERIALIZED VIEW.....	39
3.1.14. WAL sender process.....	39
3.1.15. GIN Index cost.....	39
3.1.16. Meson.....	39
3.1.17. UNICODE.....	39
3.1.18. LLVM.....	39
3.1.19. Extension.....	40
3.1.20. Reduction of WAL output.....	40
3.2. SQL Statement.....	41
3.2.1. Data Types.....	41
3.2.2. COPY.....	44
3.2.3. CREATE ROLE/USER.....	45
3.2.4. CREATE STATISTICS.....	46
3.2.5. CREATE TABLE.....	47
3.2.6. EXPLAIN.....	48
3.2.7. GRANT.....	49
3.2.8. JSON.....	51
3.2.9. REINDEX.....	53
3.2.10. VACUUM.....	54
3.2.11. Subquery.....	55
© 2022-2023 Hewlett Packard Enterprise Japan G.K.	
3	

more



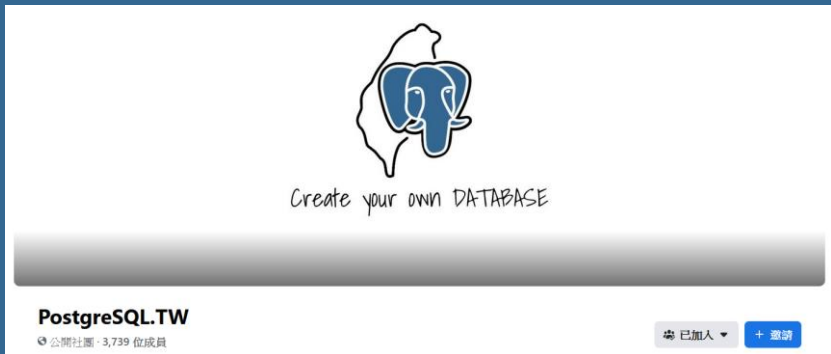
持續貢獻

DB-engine ranking

- <https://db-engines.com/en/ranking>

Rank			DBMS	Database Model	Score		
Jul 2023	Jun 2023	Jul 2022			Jul 2023	Jun 2023	Jul 2022
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1256.01	+24.54	-24.28
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1150.35	-13.59	-44.53
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	921.60	-8.47	-20.53
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	617.83	+5.01	+1.96
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	435.49	+10.13	-37.49
6.	6.	6.	Redis +	Key-value, Multi-model ⓘ	163.76	-3.59	-9.86
7.	7.	7.	IBM Db2	Relational, Multi-model ⓘ	139.81	-5.07	-21.40
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	139.59	-4.16	-14.74
9.	9.	9.	Microsoft Access	Relational	130.72	-3.73	-14.37
10.	10.	10.	SQLite +	Relational	130.20	-1.02	-6.48

- 加入FB：PostgreSQL.TW



DBMS of the Year 2022

DBMS of the Year: Snowflake



[Snowflake](#) is a cloud-based data platform, that gained popularity due to its scalability, flexibility and performance. It uses a custom SQL engine and columnar data storage, and offers a wide range of options to connect with external data sources and applications.

Snowflake climbed in our ranking to rank 11, up from rank 17 one year ago. We will see if it can enter the top 10 in 2023.

Runner-up: Google BigQuery

[BigQuery](#) is Google's cloud-based data warehouse platform. Besides the usual benefits that come with serverless computing, it comes with built-in machine learning and BI capabilities.

BigQuery stays at rank 21 in our ranking, and is making steady progress.

Third place: PostgreSQL

[PostgreSQL](#) is regular guest in our DBMS of the Year posts. It won three times and shows up again and again in the top three. PostgreSQL is known as open source DBMS with an advanced feature set. PostgreSQL 15, which was released in October 2022 brought many new features (e.g. support of the SQL MERGE statement, additional filter conditions for the logical replication of tables, structured server log output using JSON format) and performance improvements, particularly for in-memory and on-disc sorting.



Thank you.

Taiwan PostgreSQL User Group
林宗禧 linjose.tw@gmail.com

